

libstdc++

Generated by Doxygen 1.7.1

Sun Oct 10 2010 16:37:01

Contents

1	Todo List	1
2	Module Documentation	11
2.1	Extensions	11
2.1.1	Detailed Description	12
2.2	SGI	12
2.2.1	Detailed Description	15
2.2.2	Function Documentation	16
2.2.2.1	__median	16
2.2.2.2	__median	17
2.2.2.3	_Find_first	17
2.2.2.4	_Find_next	17
2.2.2.5	_Unchecked_flip	18
2.2.2.6	_Unchecked_reset	18
2.2.2.7	_Unchecked_set	18
2.2.2.8	_Unchecked_set	18
2.2.2.9	_Unchecked_test	19
2.2.2.10	compose1	19
2.2.2.11	compose2	19
2.2.2.12	constant0	19
2.2.2.13	constant1	19
2.2.2.14	constant2	19
2.2.2.15	copy_n	20

2.2.2.16	distance	20
2.2.2.17	identity_element	21
2.2.2.18	identity_element	21
2.2.2.19	iota	21
2.2.2.20	is_heap	21
2.2.2.21	is_heap	21
2.2.2.22	is_sorted	22
2.2.2.23	is_sorted	22
2.2.2.24	lexicographical_compare_3way	22
2.2.2.25	power	23
2.2.2.26	power	23
2.2.2.27	random_sample	23
2.2.2.28	random_sample	24
2.2.2.29	random_sample_n	24
2.2.2.30	random_sample_n	24
2.2.2.31	uninitialized_copy_n	25
2.3	Containers	26
2.3.1	Detailed Description	26
2.4	Sequences	27
2.4.1	Detailed Description	28
2.5	Associative	29
2.5.1	Detailed Description	29
2.6	Unordered Associative	30
2.6.1	Detailed Description	30
2.7	Diagnostics	31
2.7.1	Detailed Description	31
2.8	Concurrency	32
2.8.1	Detailed Description	32
2.9	Exceptions	33
2.9.1	Detailed Description	35
2.9.2	Typedef Documentation	35

2.9.2.1	terminate_handler	35
2.9.2.2	unexpected_handler	36
2.9.3	Function Documentation	36
2.9.3.1	__verbose_terminate_handler	36
2.9.3.2	copy_exception	36
2.9.3.3	current_exception	36
2.9.3.4	make_exception_ptr	36
2.9.3.5	rethrow_exception	37
2.9.3.6	rethrow_if_nested	37
2.9.3.7	rethrow_if_nested	37
2.9.3.8	set_terminate	37
2.9.3.9	set_unexpected	37
2.9.3.10	terminate	37
2.9.3.11	throw_with_nested	37
2.9.3.12	uncaught_exception	38
2.9.3.13	unexpected	38
2.10	Time	38
2.10.1	Detailed Description	38
2.11	Complex Numbers	39
2.11.1	Detailed Description	44
2.11.2	Function Documentation	44
2.11.2.1	abs	44
2.11.2.2	acos	44
2.11.2.3	acosh	44
2.11.2.4	arg	44
2.11.2.5	arg	44
2.11.2.6	asin	45
2.11.2.7	asinh	45
2.11.2.8	atan	45
2.11.2.9	atanh	45
2.11.2.10	conj	45

2.11.2.11 cos	45
2.11.2.12 cosh	46
2.11.2.13 exp	46
2.11.2.14 fabs	46
2.11.2.15 log	46
2.11.2.16 log10	46
2.11.2.17 norm	47
2.11.2.18 operator!=	47
2.11.2.19 operator!=	47
2.11.2.20 operator!=	47
2.11.2.21 operator*	47
2.11.2.22 operator*	47
2.11.2.23 operator*	48
2.11.2.24 operator*=	48
2.11.2.25 operator*=	48
2.11.2.26 operator+	48
2.11.2.27 operator+	48
2.11.2.28 operator+	48
2.11.2.29 operator+	49
2.11.2.30 operator+=	49
2.11.2.31 operator-	49
2.11.2.32 operator-	49
2.11.2.33 operator-	49
2.11.2.34 operator-	49
2.11.2.35 operator-=	50
2.11.2.36 operator/	50
2.11.2.37 operator/	50
2.11.2.38 operator/	50
2.11.2.39 operator/=	50
2.11.2.40 operator/=	50
2.11.2.41 operator<<	51

2.11.2.42	operator=	51
2.11.2.43	operator=	51
2.11.2.44	operator==	51
2.11.2.45	operator==	51
2.11.2.46	operator==	52
2.11.2.47	operator>>	52
2.11.2.48	polar	52
2.11.2.49	pow	52
2.11.2.50	pow	52
2.11.2.51	pow	53
2.11.2.52	sin	53
2.11.2.53	sinh	53
2.11.2.54	sqrt	53
2.11.2.55	tan	53
2.11.2.56	tanh	53
2.12	Condition Variables	54
2.12.1	Detailed Description	54
2.12.2	Enumeration Type Documentation	54
2.12.2.1	cv_status	54
2.13	Futures	55
2.13.1	Detailed Description	57
2.13.2	Enumeration Type Documentation	57
2.13.2.1	future_errc	57
2.13.3	Variable Documentation	57
2.13.3.1	future_category	57
2.14	I/O	57
2.14.1	Detailed Description	60
2.14.2	Typedef Documentation	61
2.14.2.1	filebuf	61
2.14.2.2	fstream	61
2.14.2.3	ifstream	61

2.14.2.4	ios	61
2.14.2.5	iostream	61
2.14.2.6	istream	61
2.14.2.7	istringstream	61
2.14.2.8	ofstream	62
2.14.2.9	ostream	62
2.14.2.10	ostringstream	62
2.14.2.11	streambuf	62
2.14.2.12	stringbuf	62
2.14.2.13	stringstream	62
2.14.2.14	wfilebuf	62
2.14.2.15	wfstream	63
2.14.2.16	wfstream	63
2.14.2.17	wios	63
2.14.2.18	wiostream	63
2.14.2.19	wistream	63
2.14.2.20	wistringstream	63
2.14.2.21	wofstream	63
2.14.2.22	wostream	64
2.14.2.23	wostringstream	64
2.14.2.24	wstreambuf	64
2.14.2.25	wstringbuf	64
2.14.2.26	wstringstream	64
2.15	Memory	65
2.15.1	Detailed Description	65
2.16	Pointer Abstractions	65
2.16.1	Detailed Description	68
2.16.2	Function Documentation	68
2.16.2.1	allocate_shared	68
2.16.2.2	get_deleter	69
2.16.2.3	make_shared	69

2.16.2.4	operator<<	69
2.17	Mutexes	70
2.17.1	Detailed Description	71
2.17.2	Function Documentation	71
2.17.2.1	call_once	71
2.17.2.2	lock	72
2.17.2.3	try_lock	72
2.18	Numerics	73
2.18.1	Detailed Description	73
2.19	Rational Arithmetic	74
2.19.1	Detailed Description	75
2.20	Threads	75
2.20.1	Detailed Description	76
2.21	Utilities	77
2.21.1	Detailed Description	77
2.22	Numeric Arrays	78
2.22.1	Detailed Description	88
2.22.2	Function Documentation	88
2.22.2.1	gslice	88
2.22.2.2	gslice	88
2.22.2.3	gslice	88
2.22.2.4	gslice_array	89
2.22.2.5	indirect_array	89
2.22.2.6	mask_array	89
2.22.2.7	slice	89
2.22.2.8	slice	89
2.22.2.9	slice_array	89
2.22.2.10	valarray	90
2.22.2.11	valarray	90
2.22.2.12	valarray	90
2.22.2.13	valarray	90

2.22.2.14 valarray	90
2.22.2.15 valarray	90
2.22.2.16 valarray	91
2.22.2.17 valarray	91
2.22.2.18 valarray	91
2.22.2.19 ~gslice	91
2.22.2.20 apply	91
2.22.2.21 apply	92
2.22.2.22 begin	92
2.22.2.23 begin	92
2.22.2.24 cshift	92
2.22.2.25 end	93
2.22.2.26 end	93
2.22.2.27 max	94
2.22.2.28 min	94
2.22.2.29 operator!	94
2.22.2.30 operator%=	94
2.22.2.31 operator%=	94
2.22.2.32 operator%=	94
2.22.2.33 operator&=	95
2.22.2.34 operator&=	95
2.22.2.35 operator&=	95
2.22.2.36 operator*=	95
2.22.2.37 operator*=	95
2.22.2.38 operator*=	96
2.22.2.39 operator+	96
2.22.2.40 operator+=	96
2.22.2.41 operator+=	96
2.22.2.42 operator+=	96
2.22.2.43 operator-	96
2.22.2.44 operator-=	97

2.22.2.45 operator-=	97
2.22.2.46 operator-=	97
2.22.2.47 operator/=	97
2.22.2.48 operator/=	97
2.22.2.49 operator/=	97
2.22.2.50 operator<=<=	98
2.22.2.51 operator<=<=	98
2.22.2.52 operator<=<=	98
2.22.2.53 operator=	98
2.22.2.54 operator=	98
2.22.2.55 operator=	99
2.22.2.56 operator=	99
2.22.2.57 operator=	99
2.22.2.58 operator=	99
2.22.2.59 operator=	99
2.22.2.60 operator=	99
2.22.2.61 operator=	100
2.22.2.62 operator=	100
2.22.2.63 operator=	100
2.22.2.64 operator=	100
2.22.2.65 operator=	101
2.22.2.66 operator=	101
2.22.2.67 operator=	101
2.22.2.68 operator=	101
2.22.2.69 operator=	102
2.22.2.70 operator=	102
2.22.2.71 operator=	102
2.22.2.72 operator>>=	102
2.22.2.73 operator>>=	103
2.22.2.74 operator>>=	103
2.22.2.75 operator[]	103

2.22.2.76 operator[]	103
2.22.2.77 operator[]	104
2.22.2.78 operator[]	104
2.22.2.79 operator[]	105
2.22.2.80 operator[]	105
2.22.2.81 operator[]	106
2.22.2.82 operator[]	106
2.22.2.83 operator[]	106
2.22.2.84 operator^=	107
2.22.2.85 operator^=	107
2.22.2.86 operator^=	107
2.22.2.87 operator =	107
2.22.2.88 operator =	107
2.22.2.89 operator =	108
2.22.2.90 operator~	108
2.22.2.91 resize	108
2.22.2.92 shift	108
2.22.2.93 size	109
2.22.2.94 size	109
2.22.2.95 size	109
2.22.2.96 start	109
2.22.2.97 start	109
2.22.2.98 stride	110
2.22.2.99 stride	110
2.22.2.100sum	110
2.23 Mathematical Special Functions	110
2.23.1 Detailed Description	113
2.23.2 Function Documentation	113
2.23.2.1 assoc_laguerre	113
2.23.2.2 assoc_legendre	114
2.23.2.3 beta	114

2.23.2.4	<code>comp_ellint_1</code>	114
2.23.2.5	<code>comp_ellint_2</code>	114
2.23.2.6	<code>comp_ellint_3</code>	114
2.23.2.7	<code>conf_hyperg</code>	114
2.23.2.8	<code>cyl_bessel_i</code>	115
2.23.2.9	<code>cyl_bessel_j</code>	115
2.23.2.10	<code>cyl_bessel_k</code>	115
2.23.2.11	<code>cyl_neumann</code>	115
2.23.2.12	<code>ellint_1</code>	115
2.23.2.13	<code>ellint_2</code>	115
2.23.2.14	<code>ellint_3</code>	116
2.23.2.15	<code>expint</code>	116
2.23.2.16	<code>hermite</code>	116
2.23.2.17	<code>hyperg</code>	116
2.23.2.18	<code>laguerre</code>	116
2.23.2.19	<code>legendre</code>	116
2.23.2.20	<code>riemann_zeta</code>	117
2.23.2.21	<code>sph_bessel</code>	117
2.23.2.22	<code>sph_legendre</code>	117
2.23.2.23	<code>sph_neumann</code>	117
2.24	Type Traits	118
2.24.1	Detailed Description	123
2.24.2	Define Documentation	124
2.24.2.1	<code>_GLIBCXX_HAS_NESTED_TYPE</code>	124
2.24.3	Typedef Documentation	124
2.24.3.1	<code>false_type</code>	124
2.24.3.2	<code>true_type</code>	124
2.25	Decimal Floating-Point Arithmetic	124
2.25.1	Detailed Description	125
2.26	Binder Classes	125
2.26.1	Detailed Description	126

2.26.2	Function Documentation	127
2.26.2.1	bind	127
2.26.2.2	bind1st	127
2.26.2.3	bind2nd	127
2.26.3	Variable Documentation	127
2.26.3.1	<code>_GLIBCXX_DEPRECATED_ATTR</code>	127
2.27	Algorithms	128
2.27.1	Detailed Description	128
2.28	Mutating	129
2.28.1	Function Documentation	132
2.28.1.1	copy	132
2.28.1.2	copy_backward	132
2.28.1.3	copy_if	133
2.28.1.4	copy_n	133
2.28.1.5	fill	134
2.28.1.6	fill_n	134
2.28.1.7	generate	135
2.28.1.8	generate_n	135
2.28.1.9	is_partitioned	136
2.28.1.10	iter_swap	136
2.28.1.11	move	137
2.28.1.12	move	137
2.28.1.13	move_backward	138
2.28.1.14	partition	138
2.28.1.15	partition_copy	139
2.28.1.16	partition_point	139
2.28.1.17	random_shuffle	140
2.28.1.18	random_shuffle	140
2.28.1.19	remove	141
2.28.1.20	remove_copy	141
2.28.1.21	remove_copy_if	142

2.28.1.22	remove_if	142
2.28.1.23	replace	143
2.28.1.24	replace_copy_if	143
2.28.1.25	replace_if	144
2.28.1.26	reverse	144
2.28.1.27	reverse_copy	145
2.28.1.28	rotate	145
2.28.1.29	rotate_copy	146
2.28.1.30	shuffle	147
2.28.1.31	stable_partition	147
2.28.1.32	swap	148
2.28.1.33	swap_ranges	148
2.28.1.34	transform	149
2.28.1.35	transform	149
2.28.1.36	unique	150
2.28.1.37	unique	150
2.28.1.38	unique_copy	151
2.28.1.39	unique_copy	151
2.29	Non-Mutating	152
2.29.1	Function Documentation	154
2.29.1.1	adjacent_find	154
2.29.1.2	adjacent_find	154
2.29.1.3	all_of	155
2.29.1.4	any_of	155
2.29.1.5	count	156
2.29.1.6	count_if	156
2.29.1.7	equal	157
2.29.1.8	equal	157
2.29.1.9	find	158
2.29.1.10	find_end	158
2.29.1.11	find_end	159

2.29.1.12	find_first_of	160
2.29.1.13	find_first_of	160
2.29.1.14	find_if	161
2.29.1.15	find_if_not	161
2.29.1.16	for_each	162
2.29.1.17	mismatch	162
2.29.1.18	mismatch	163
2.29.1.19	none_of	163
2.29.1.20	search	164
2.29.1.21	search	164
2.29.1.22	search_n	165
2.29.1.23	search_n	166
2.30	Sorting	167
2.30.1	Function Documentation	170
2.30.1.1	inplace_merge	170
2.30.1.2	inplace_merge	170
2.30.1.3	is_sorted	171
2.30.1.4	is_sorted	172
2.30.1.5	is_sorted_until	172
2.30.1.6	is_sorted_until	172
2.30.1.7	lexicographical_compare	173
2.30.1.8	lexicographical_compare	173
2.30.1.9	max	174
2.30.1.10	max	174
2.30.1.11	max_element	175
2.30.1.12	max_element	175
2.30.1.13	merge	176
2.30.1.14	merge	176
2.30.1.15	min	177
2.30.1.16	min	177
2.30.1.17	min_element	178

2.30.1.18 min_element	178
2.30.1.19 minmax	179
2.30.1.20 minmax	179
2.30.1.21 minmax_element	180
2.30.1.22 minmax_element	180
2.30.1.23 next_permutation	181
2.30.1.24 next_permutation	181
2.30.1.25 nth_element	182
2.30.1.26 nth_element	182
2.30.1.27 partial_sort	183
2.30.1.28 partial_sort	183
2.30.1.29 partial_sort_copy	184
2.30.1.30 partial_sort_copy	185
2.30.1.31 prev_permutation	185
2.30.1.32 prev_permutation	186
2.30.1.33 sort	186
2.30.1.34 sort	187
2.30.1.35 stable_sort	187
2.30.1.36 stable_sort	188
2.31 Set Operation	189
2.31.1 Detailed Description	190
2.31.2 Function Documentation	190
2.31.2.1 includes	190
2.31.2.2 includes	191
2.31.2.3 set_difference	191
2.31.2.4 set_difference	192
2.31.2.5 set_intersection	193
2.31.2.6 set_intersection	193
2.31.2.7 set_symmetric_difference	194
2.31.2.8 set_symmetric_difference	194
2.31.2.9 set_union	195

2.31.2.10	set_union	196
2.32	Binary Search	196
2.32.1	Detailed Description	197
2.32.2	Function Documentation	198
2.32.2.1	binary_search	198
2.32.2.2	binary_search	198
2.32.2.3	equal_range	199
2.32.2.4	equal_range	199
2.32.2.5	lower_bound	200
2.32.2.6	lower_bound	201
2.32.2.7	upper_bound	201
2.32.2.8	upper_bound	202
2.33	Allocators	202
2.33.1	Detailed Description	204
2.34	Atomics	204
2.34.1	Detailed Description	210
2.34.2	Define Documentation	210
2.34.2.1	_GLIBCXX_ATOMIC_PROPERTY	210
2.34.3	Typedef Documentation	210
2.34.3.1	atomic_char	210
2.34.3.2	atomic_char16_t	210
2.34.3.3	atomic_char32_t	210
2.34.3.4	atomic_int	210
2.34.3.5	atomic_llong	211
2.34.3.6	atomic_long	211
2.34.3.7	atomic_schar	211
2.34.3.8	atomic_short	211
2.34.3.9	atomic_uchar	211
2.34.3.10	atomic_uint	211
2.34.3.11	atomic_ullong	211
2.34.3.12	atomic_ulong	212

2.34.3.13	atomic_ushort	212
2.34.3.14	atomic_wchar_t	212
2.34.3.15	memory_order	212
2.34.4	Enumeration Type Documentation	212
2.34.4.1	memory_order	212
2.34.5	Function Documentation	212
2.34.5.1	kill_dependency	212
2.35	Hashes	213
2.35.1	Detailed Description	213
2.36	Locales	213
2.36.1	Detailed Description	216
2.37	Random Number Generation	216
2.37.1	Detailed Description	217
2.37.2	Function Documentation	217
2.37.2.1	generate_canonical	217
2.38	Regular Expressions	217
2.38.1	Detailed Description	223
2.38.2	Typedef Documentation	223
2.38.2.1	cregex_token_iterator	223
2.38.2.2	csub_match	223
2.38.2.3	regex	224
2.38.2.4	sregex_token_iterator	224
2.38.2.5	ssub_match	224
2.38.2.6	wcregex_token_iterator	224
2.38.2.7	wsub_match	224
2.38.2.8	wregex	224
2.38.2.9	wsregex_token_iterator	224
2.38.2.10	wssub_match	225
2.38.3	Function Documentation	225
2.38.3.1	isctype	225
2.38.3.2	operator!=	225

2.38.3.3	operator!=	226
2.38.3.4	operator!=	226
2.38.3.5	operator!=	226
2.38.3.6	operator!=	227
2.38.3.7	operator!=	227
2.38.3.8	operator!=	228
2.38.3.9	operator!=	228
2.38.3.10	operator<	228
2.38.3.11	operator<	229
2.38.3.12	operator<	229
2.38.3.13	operator<	230
2.38.3.14	operator<	230
2.38.3.15	operator<	230
2.38.3.16	operator<	231
2.38.3.17	operator<<	231
2.38.3.18	operator<=	232
2.38.3.19	operator<=	232
2.38.3.20	operator<=	232
2.38.3.21	operator<=	233
2.38.3.22	operator<=	233
2.38.3.23	operator<=	233
2.38.3.24	operator<=	234
2.38.3.25	operator==	234
2.38.3.26	operator==	235
2.38.3.27	operator==	235
2.38.3.28	operator==	235
2.38.3.29	operator==	236
2.38.3.30	operator==	236
2.38.3.31	operator==	237
2.38.3.32	operator==	237
2.38.3.33	operator>	237

2.38.3.34 operator>	238
2.38.3.35 operator>	238
2.38.3.36 operator>	239
2.38.3.37 operator>	239
2.38.3.38 operator>	239
2.38.3.39 operator>	240
2.38.3.40 operator>=	240
2.38.3.41 operator>=	241
2.38.3.42 operator>=	241
2.38.3.43 operator>=	241
2.38.3.44 operator>=	242
2.38.3.45 operator>=	242
2.38.3.46 operator>=	243
2.38.3.47 regex_match	243
2.38.3.48 regex_match	244
2.38.3.49 regex_match	244
2.38.3.50 regex_match	245
2.38.3.51 regex_match	246
2.38.3.52 regex_match	246
2.38.3.53 regex_replace	247
2.38.3.54 regex_replace	248
2.38.3.55 regex_search	248
2.38.3.56 regex_search	249
2.38.3.57 regex_search	250
2.38.3.58 regex_search	250
2.38.3.59 regex_search	251
2.38.3.60 regex_search	252
2.38.3.61 swap	253
2.38.3.62 swap	253
2.38.3.63 value	253
2.39 Function Objects	254

2.39.1 Detailed Description	255
2.39.2 Function Documentation	256
2.39.2.1 mem_fn	256
2.40 Arithmetic Classes	256
2.40.1 Detailed Description	257
2.41 Comparison Classes	257
2.41.1 Detailed Description	258
2.42 Boolean Operations Classes	258
2.42.1 Detailed Description	258
2.43 Negators	259
2.43.1 Detailed Description	259
2.43.2 Function Documentation	260
2.43.2.1 not1	260
2.43.2.2 not2	260
2.44 Adaptors for pointers to functions	260
2.44.1 Detailed Description	261
2.44.2 Function Documentation	261
2.44.2.1 ptr_fun	261
2.44.2.2 ptr_fun	262
2.45 Adaptors for pointers to members	262
2.45.1 Detailed Description	263
2.46 Heap	263
2.46.1 Function Documentation	265
2.46.1.1 is_heap	265
2.46.1.2 is_heap	265
2.46.1.3 is_heap_until	265
2.46.1.4 is_heap_until	266
2.46.1.5 make_heap	266
2.46.1.6 make_heap	267
2.46.1.7 pop_heap	267
2.46.1.8 pop_heap	268

2.46.1.9	push_heap	268
2.46.1.10	push_heap	268
2.46.1.11	sort_heap	269
2.46.1.12	sort_heap	269
2.47	Iterators	270
2.47.1	Detailed Description	274
2.47.2	Function Documentation	274
2.47.2.1	__iterator_category	274
2.47.2.2	back_inserter	274
2.47.2.3	front_inserter	275
2.47.2.4	inserter	275
2.47.2.5	operator!=	275
2.47.2.6	operator==	276
2.47.2.7	operator==	276
2.48	Iterator Tags	276
2.48.1	Detailed Description	277
2.49	Strings	277
2.49.1	Typedef Documentation	278
2.49.1.1	string	278
2.49.1.2	u16string	278
2.49.1.3	u32string	278
2.49.1.4	wstring	278
2.50	Policy-Based Data Structures	279
2.50.1	Detailed Description	280
2.51	Random Number Generators	280
2.51.1	Detailed Description	282
2.51.2	Typedef Documentation	283
2.51.2.1	minstd_rand	283
2.51.2.2	minstd_rand0	283
2.51.2.3	mt19937	283
2.51.2.4	mt19937_64	283

2.51.3	Function Documentation	284
2.51.3.1	operator!=	284
2.51.3.2	operator!=	284
2.51.3.3	operator!=	285
2.51.3.4	operator!=	285
2.51.3.5	operator!=	286
2.51.3.6	operator!=	286
2.51.3.7	operator<<	287
2.52	Random Number Distributions	287
2.53	Uniform	288
2.53.1	Function Documentation	289
2.53.1.1	operator!=	289
2.53.1.2	operator!=	289
2.53.1.3	operator<<	289
2.53.1.4	operator<<	290
2.53.1.5	operator==	290
2.53.1.6	operator==	290
2.53.1.7	operator>>	291
2.53.1.8	operator>>	291
2.54	Normal	292
2.54.1	Function Documentation	293
2.54.1.1	operator!=	293
2.54.1.2	operator!=	293
2.54.1.3	operator!=	294
2.54.1.4	operator!=	294
2.54.1.5	operator!=	294
2.54.1.6	operator!=	294
2.54.1.7	operator!=	294
2.54.1.8	operator<<	295
2.54.1.9	operator==	295
2.54.1.10	operator>>	295

2.55 Bernoulli	296
2.55.1 Function Documentation	297
2.55.1.1 operator!=	297
2.55.1.2 operator!=	297
2.55.1.3 operator!=	298
2.55.1.4 operator!=	298
2.55.1.5 operator<<	298
2.55.1.6 operator<<	298
2.55.1.7 operator==	299
2.55.1.8 operator==	299
2.55.1.9 operator>>	299
2.55.1.10 operator>>	300
2.56 Poisson	300
2.56.1 Function Documentation	303
2.56.1.1 operator!=	303
2.56.1.2 operator!=	303
2.56.1.3 operator!=	303
2.56.1.4 operator!=	303
2.56.1.5 operator!=	303
2.56.1.6 operator!=	304
2.56.1.7 operator!=	304
2.56.1.8 operator<<	304
2.56.1.9 operator<<	304
2.56.1.10 operator<<	305
2.56.1.11 operator==	305
2.56.1.12 operator==	306
2.56.1.13 operator==	306
2.56.1.14 operator==	306
2.56.1.15 operator==	306
2.56.1.16 operator==	307
2.56.1.17 operator>>	307

2.56.1.18 operator>>	307
2.56.1.19 operator>>	308
2.57 Random Number Utilities	308
3 Directory Documentation	311
3.1 include/backward/ Directory Reference	312
3.2 include/x86_64-unknown-linux-gnu/bits/ Directory Reference	313
3.3 include/bits/ Directory Reference	315
3.4 include/debug/ Directory Reference	318
3.5 include/decimal/ Directory Reference	319
3.6 include/ext/pb_ds/detail/ Directory Reference	320
3.7 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/ Directory Reference	321
3.8 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/doxygen/ Directory Reference	322
3.9 include/ext/ Directory Reference	323
3.10 /mnt/share/src/gcc.svn-trunk/ Directory Reference	325
3.11 include/profile/impl/ Directory Reference	326
3.12 include/ Directory Reference	328
3.13 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/ Directory Reference	331
3.14 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/libsupc++/ Directory Refer- ence	332
3.15 include/parallel/ Directory Reference	333
3.16 include/ext/pb_ds/ Directory Reference	336
3.17 include/profile/ Directory Reference	338
3.18 /mnt/share/src/ Directory Reference	339
3.19 include/tr1/ Directory Reference	340
3.20 include/tr1_impl/ Directory Reference	341
3.21 include/x86_64-unknown-linux-gnu/ Directory Reference	342
4 Namespace Documentation	343
4.1 __gnu_cxx Namespace Reference	343
4.1.1 Detailed Description	365
4.1.2 Function Documentation	365

4.1.2.1	__static_pointer_cast	365
4.1.2.2	__static_pointer_cast	365
4.1.2.3	_Bit_scan_forward	366
4.1.2.4	operator!=	366
4.1.2.5	operator!=	366
4.1.2.6	operator!=	367
4.1.2.7	operator+	367
4.1.2.8	operator+	368
4.1.2.9	operator+	368
4.1.2.10	operator+	369
4.1.2.11	operator+	369
4.1.2.12	operator<	370
4.1.2.13	operator<	370
4.1.2.14	operator<	370
4.1.2.15	operator<=	371
4.1.2.16	operator<=	371
4.1.2.17	operator<=	372
4.1.2.18	operator==	372
4.1.2.19	operator==	372
4.1.2.20	operator==	373
4.1.2.21	operator==	373
4.1.2.22	operator>	374
4.1.2.23	operator>	374
4.1.2.24	operator>	375
4.1.2.25	operator>=	375
4.1.2.26	operator>=	376
4.1.2.27	operator>=	376
4.1.2.28	swap	377
4.2	__gnu_cxx::__detail Namespace Reference	377
4.2.1	Detailed Description	378
4.2.2	Function Documentation	378

4.2.2.1	__bit_allocate	378
4.2.2.2	__bit_free	378
4.2.2.3	__num_bitmaps	378
4.2.2.4	__num_blocks	379
4.3	__gnu_cxx::typelist Namespace Reference	379
4.3.1	Detailed Description	379
4.3.2	Function Documentation	379
4.3.2.1	apply_generator	379
4.4	__gnu_debug Namespace Reference	380
4.4.1	Detailed Description	386
4.4.2	Function Documentation	386
4.4.2.1	__base	386
4.4.2.2	__check_dereferenceable	387
4.4.2.3	__check_dereferenceable	387
4.4.2.4	__check_dereferenceable	387
4.4.2.5	__check_singular	387
4.4.2.6	__check_singular	387
4.4.2.7	__check_singular_aux	388
4.4.2.8	__check_string	388
4.4.2.9	__check_string	388
4.4.2.10	__valid_range	388
4.4.2.11	__valid_range	388
4.4.2.12	__valid_range_aux	389
4.4.2.13	__valid_range_aux	389
4.4.2.14	__valid_range_aux2	389
4.4.2.15	__valid_range_aux2	389
4.5	__gnu_internal Namespace Reference	390
4.5.1	Detailed Description	390
4.6	__gnu_parallel Namespace Reference	390
4.6.1	Detailed Description	406
4.6.2	Typedef Documentation	406

4.6.2.1	_BinIndex	406
4.6.2.2	_CASable	406
4.6.2.3	_SequenceIndex	407
4.6.2.4	_ThreadIndex	407
4.6.3	Enumeration Type Documentation	407
4.6.3.1	_AlgorithmStrategy	407
4.6.3.2	_FindAlgorithm	407
4.6.3.3	_MultiwayMergeAlgorithm	407
4.6.3.4	_Parallelism	407
4.6.3.5	_PartialSumAlgorithm	408
4.6.3.6	_SortAlgorithm	408
4.6.3.7	_SplittingAlgorithm	408
4.6.4	Function Documentation	408
4.6.4.1	__calc_borders	408
4.6.4.2	__compare_and_swap	409
4.6.4.3	__compare_and_swap_32	409
4.6.4.4	__compare_and_swap_64	409
4.6.4.5	__decode2	410
4.6.4.6	__determine_samples	410
4.6.4.7	__encode2	411
4.6.4.8	__fetch_and_add	411
4.6.4.9	__fetch_and_add_32	412
4.6.4.10	__fetch_and_add_64	412
4.6.4.11	__find_template	412
4.6.4.12	__find_template	413
4.6.4.13	__find_template	413
4.6.4.14	__find_template	414
4.6.4.15	__for_each_template_random_access	415
4.6.4.16	__for_each_template_random_access_ed	416
4.6.4.17	__for_each_template_random_access_omp_loop	416

4.6.4.18	__for_each_template_random_access_omp_loop_-static	417
4.6.4.19	__for_each_template_random_access_workstealing	418
4.6.4.20	__is_sorted	419
4.6.4.21	__median_of_three_iterators	419
4.6.4.22	__merge_advance	419
4.6.4.23	__merge_advance_movc	420
4.6.4.24	__merge_advance_usual	421
4.6.4.25	__parallel_merge_advance	421
4.6.4.26	__parallel_merge_advance	422
4.6.4.27	__parallel_nth_element	423
4.6.4.28	__parallel_partial_sort	423
4.6.4.29	__parallel_partial_sum	424
4.6.4.30	__parallel_partial_sum_basecase	424
4.6.4.31	__parallel_partial_sum_linear	425
4.6.4.32	__parallel_partition	425
4.6.4.33	__parallel_random_shuffle	426
4.6.4.34	__parallel_random_shuffle_drs	426
4.6.4.35	__parallel_random_shuffle_drs_pu	427
4.6.4.36	__parallel_sort	427
4.6.4.37	__parallel_sort	428
4.6.4.38	__parallel_sort	429
4.6.4.39	__parallel_sort	430
4.6.4.40	__parallel_sort	430
4.6.4.41	__parallel_sort	431
4.6.4.42	__parallel_sort	432
4.6.4.43	__parallel_sort_qs	432
4.6.4.44	__parallel_sort_qs_conquer	433
4.6.4.45	__parallel_sort_qs_divide	433
4.6.4.46	__parallel_sort_qsb	434
4.6.4.47	__parallel_unique_copy	434

4.6.4.48	<code>__parallel_unique_copy</code>	434
4.6.4.49	<code>__qsb_conquer</code>	435
4.6.4.50	<code>__qsb_divide</code>	436
4.6.4.51	<code>__qsb_local_sort_with_helping</code>	436
4.6.4.52	<code>__random_number_pow2</code>	437
4.6.4.53	<code>__rd_log2</code>	437
4.6.4.54	<code>__round_up_to_pow2</code>	437
4.6.4.55	<code>__search_template</code>	438
4.6.4.56	<code>__sequential_multiway_merge</code>	438
4.6.4.57	<code>__sequential_random_shuffle</code>	439
4.6.4.58	<code>__shrink</code>	439
4.6.4.59	<code>__shrink_and_double</code>	440
4.6.4.60	<code>__yield</code>	440
4.6.4.61	<code>equally_split</code>	440
4.6.4.62	<code>equally_split_point</code>	441
4.6.4.63	<code>list_partition</code>	441
4.6.4.64	<code>max</code>	442
4.6.4.65	<code>min</code>	442
4.6.4.66	<code>multiseq_partition</code>	443
4.6.4.67	<code>multiseq_selection</code>	443
4.6.4.68	<code>multiway_merge</code>	444
4.6.4.69	<code>multiway_merge_3_variant</code>	446
4.6.4.70	<code>multiway_merge_4_variant</code>	447
4.6.4.71	<code>multiway_merge_exact_splitting</code>	447
4.6.4.72	<code>multiway_merge_loser_tree</code>	448
4.6.4.73	<code>multiway_merge_loser_tree_sentinel</code>	449
4.6.4.74	<code>multiway_merge_loser_tree_unguarded</code>	449
4.6.4.75	<code>multiway_merge_sampling_splitting</code>	450
4.6.4.76	<code>multiway_merge_sentinels</code>	450
4.6.4.77	<code>parallel_multiway_merge</code>	452
4.6.4.78	<code>parallel_sort_mwms</code>	453

4.6.4.79	parallel_sort_mwms_pu	454
4.6.5	Variable Documentation	454
4.6.5.1	_CASable_bits	454
4.6.5.2	_CASable_mask	454
4.7	__gnu_pbds Namespace Reference	454
4.7.1	Detailed Description	457
4.8	__gnu_profile Namespace Reference	457
4.8.1	Detailed Description	462
4.8.2	Typedef Documentation	463
4.8.2.1	__env_t	463
4.8.3	Function Documentation	463
4.8.3.1	__get__global_lock	463
4.8.3.2	__profcxx_init	463
4.8.3.3	__report	463
4.9	__gnu_sequential Namespace Reference	464
4.9.1	Detailed Description	464
4.10	abi Namespace Reference	464
4.10.1	Detailed Description	464
4.11	std Namespace Reference	464
4.11.1	Detailed Description	604
4.11.2	Typedef Documentation	604
4.11.2.1	new_handler	604
4.11.2.2	streamoff	604
4.11.2.3	streampos	604
4.11.2.4	streamsize	605
4.11.2.5	u16streampos	605
4.11.2.6	u32streampos	605
4.11.2.7	wstreampos	605
4.11.3	Enumeration Type Documentation	605
4.11.3.1	"@52	605
4.11.3.2	float_denorm_style	605

4.11.3.3	<code>float_round_style</code>	606
4.11.4	Function Documentation	606
4.11.4.1	<code>__final_insertion_sort</code>	606
4.11.4.2	<code>__final_insertion_sort</code>	606
4.11.4.3	<code>__find</code>	607
4.11.4.4	<code>__find</code>	607
4.11.4.5	<code>__find_if</code>	607
4.11.4.6	<code>__find_if</code>	607
4.11.4.7	<code>__find_if_not</code>	607
4.11.4.8	<code>__find_if_not</code>	608
4.11.4.9	<code>__gcd</code>	608
4.11.4.10	<code>__heap_select</code>	608
4.11.4.11	<code>__heap_select</code>	608
4.11.4.12	<code>__inplace_stable_partition</code>	609
4.11.4.13	<code>__inplace_stable_sort</code>	609
4.11.4.14	<code>__inplace_stable_sort</code>	609
4.11.4.15	<code>__insertion_sort</code>	609
4.11.4.16	<code>__insertion_sort</code>	610
4.11.4.17	<code>__introsort_loop</code>	610
4.11.4.18	<code>__introsort_loop</code>	610
4.11.4.19	<code>__lg</code>	610
4.11.4.20	<code>__merge_adaptive</code>	611
4.11.4.21	<code>__merge_adaptive</code>	611
4.11.4.22	<code>__merge_backward</code>	611
4.11.4.23	<code>__merge_backward</code>	612
4.11.4.24	<code>__merge_without_buffer</code>	612
4.11.4.25	<code>__merge_without_buffer</code>	612
4.11.4.26	<code>__move_median_first</code>	612
4.11.4.27	<code>__move_median_first</code>	613
4.11.4.28	<code>__partition</code>	613
4.11.4.29	<code>__partition</code>	613

4.11.4.30 <code>__reverse</code>	613
4.11.4.31 <code>__reverse</code>	614
4.11.4.32 <code>__rotate</code>	614
4.11.4.33 <code>__rotate</code>	614
4.11.4.34 <code>__rotate</code>	614
4.11.4.35 <code>__rotate_adaptive</code>	615
4.11.4.36 <code>__search_n</code>	615
4.11.4.37 <code>__search_n</code>	615
4.11.4.38 <code>__search_n</code>	615
4.11.4.39 <code>__search_n</code>	616
4.11.4.40 <code>__stable_partition_adaptive</code>	616
4.11.4.41 <code>__unguarded_insertion_sort</code>	616
4.11.4.42 <code>__unguarded_insertion_sort</code>	616
4.11.4.43 <code>__unguarded_linear_insert</code>	617
4.11.4.44 <code>__unguarded_linear_insert</code>	617
4.11.4.45 <code>__unguarded_partition</code>	617
4.11.4.46 <code>__unguarded_partition</code>	617
4.11.4.47 <code>__unguarded_partition_pivot</code>	618
4.11.4.48 <code>__unguarded_partition_pivot</code>	618
4.11.4.49 <code>__unique_copy</code>	618
4.11.4.50 <code>__unique_copy</code>	618
4.11.4.51 <code>__unique_copy</code>	619
4.11.4.52 <code>__unique_copy</code>	619
4.11.4.53 <code>__unique_copy</code>	619
4.11.4.54 <code>__unique_copy</code>	619
4.11.4.55 <code>_Construct</code>	620
4.11.4.56 <code>_Destroy</code>	620
4.11.4.57 <code>_Destroy</code>	620
4.11.4.58 <code>accumulate</code>	620
4.11.4.59 <code>accumulate</code>	621
4.11.4.60 <code>addressof</code>	621

4.11.4.61 adjacent_difference	622
4.11.4.62 adjacent_difference	622
4.11.4.63 advance	623
4.11.4.64 begin	623
4.11.4.65 begin	623
4.11.4.66 begin	624
4.11.4.67 begin	624
4.11.4.68 bind	624
4.11.4.69 boolalpha	625
4.11.4.70 const_pointer_cast	625
4.11.4.71 cref	625
4.11.4.72 cref	625
4.11.4.73 dec	625
4.11.4.74 distance	626
4.11.4.75 dynamic_pointer_cast	626
4.11.4.76 end	626
4.11.4.77 end	627
4.11.4.78 end	627
4.11.4.79 end	627
4.11.4.80 endl	628
4.11.4.81 ends	628
4.11.4.82 fixed	628
4.11.4.83 flush	628
4.11.4.84 forward	629
4.11.4.85 forward	629
4.11.4.86 get_money	629
4.11.4.87 get_temporary_buffer	629
4.11.4.88 getline	630
4.11.4.89 getline	630
4.11.4.90 getline	631
4.11.4.91 getline	632

4.11.4.92 has_facet	632
4.11.4.93 hex	633
4.11.4.94 inner_product	633
4.11.4.95 inner_product	633
4.11.4.96 internal	634
4.11.4.97 iota	634
4.11.4.98 isalnum	634
4.11.4.99 isalpha	635
4.11.4.100 iscntrl	635
4.11.4.101 isdigit	635
4.11.4.102 isgraph	635
4.11.4.103 islower	635
4.11.4.104 isprint	635
4.11.4.105 ispunct	635
4.11.4.106 isspace	635
4.11.4.107 isupper	636
4.11.4.108 isxdigit	636
4.11.4.109 left	636
4.11.4.110 make_pair	636
4.11.4.111 inboolalpha	637
4.11.4.112 noshowbase	637
4.11.4.113 noshowpoint	637
4.11.4.114 noshowpos	637
4.11.4.115 noskipws	637
4.11.4.116 nounitbuf	637
4.11.4.117 nouppercase	638
4.11.4.118 oct	638
4.11.4.119 operator!=	638
4.11.4.120 operator!=	638
4.11.4.121 operator!=	639
4.11.4.122 operator!=	639

4.11.4.123operator!=	639
4.11.4.124operator!=	640
4.11.4.125operator!=	640
4.11.4.126operator!=	640
4.11.4.127operator!=	640
4.11.4.128operator!=	640
4.11.4.129operator!=	641
4.11.4.130operator!=	641
4.11.4.131operator!=	641
4.11.4.132operator!=	641
4.11.4.133operator!=	641
4.11.4.134operator!=	642
4.11.4.135operator!=	642
4.11.4.136operator&	642
4.11.4.137operator+	642
4.11.4.138operator+	643
4.11.4.139operator+	643
4.11.4.140operator+	644
4.11.4.141operator+	644
4.11.4.142operator<	644
4.11.4.143operator<	645
4.11.4.144operator<	645
4.11.4.145operator<	646
4.11.4.146operator<	646
4.11.4.147operator<	647
4.11.4.148operator<	647
4.11.4.149operator<	648
4.11.4.150operator<	648
4.11.4.151operator<	648
4.11.4.152operator<	649
4.11.4.153operator<	649

4.11.4.154operator<	650
4.11.4.155operator<	650
4.11.4.156operator<<	651
4.11.4.157operator<<	651
4.11.4.158operator<<	652
4.11.4.159operator<<	652
4.11.4.160operator<<	653
4.11.4.161operator<<	653
4.11.4.162operator<<	653
4.11.4.163operator<<	654
4.11.4.164operator<<	654
4.11.4.165operator<<	655
4.11.4.166operator<<	655
4.11.4.167operator<<	656
4.11.4.168operator<<	656
4.11.4.169operator<<	657
4.11.4.170operator<=	658
4.11.4.171operator<=	658
4.11.4.172operator<=	658
4.11.4.173operator<=	659
4.11.4.174operator<=	659
4.11.4.175operator<=	659
4.11.4.176operator<=	659
4.11.4.177operator<=	660
4.11.4.178operator<=	660
4.11.4.179operator<=	660
4.11.4.180operator<=	660
4.11.4.181operator<=	660
4.11.4.182operator<=	661
4.11.4.183operator<=	661
4.11.4.184operator==	661

4.11.4.185operator==	661
4.11.4.186operator==	662
4.11.4.187operator==	662
4.11.4.188operator==	663
4.11.4.189operator==	663
4.11.4.190operator==	663
4.11.4.191operator==	664
4.11.4.192operator==	664
4.11.4.193operator==	665
4.11.4.194operator==	665
4.11.4.195operator==	665
4.11.4.196operator==	666
4.11.4.197operator==	666
4.11.4.198operator==	667
4.11.4.199operator==	667
4.11.4.200operator==	667
4.11.4.201operator==	667
4.11.4.202operator>	668
4.11.4.203operator>	668
4.11.4.204operator>	668
4.11.4.205operator>	669
4.11.4.206operator>	669
4.11.4.207operator>	669
4.11.4.208operator>	669
4.11.4.209operator>	669
4.11.4.210operator>	670
4.11.4.211operator>	670
4.11.4.212operator>	670
4.11.4.213operator>	671
4.11.4.214operator>	671
4.11.4.215operator>	671

4.11.4.216operator>=	671
4.11.4.217operator>=	671
4.11.4.218operator>=	672
4.11.4.219operator>=	672
4.11.4.220operator>=	672
4.11.4.221operator>=	672
4.11.4.222operator>=	673
4.11.4.223operator>=	673
4.11.4.224operator>=	673
4.11.4.225operator>=	673
4.11.4.226operator>=	674
4.11.4.227operator>=	674
4.11.4.228operator>=	674
4.11.4.229operator>=	674
4.11.4.230operator>>	675
4.11.4.231operator>>	675
4.11.4.232operator>>	675
4.11.4.233operator>>	676
4.11.4.234operator>>	677
4.11.4.235operator>>	678
4.11.4.236operator>>	678
4.11.4.237operator>>	679
4.11.4.238operator>>	679
4.11.4.239operator>>	680
4.11.4.240operator>>	680
4.11.4.241operator^	681
4.11.4.242operator	682
4.11.4.243partial_sum	682
4.11.4.244partial_sum	683
4.11.4.245put_money	683
4.11.4.246ref	683

4.11.4.247ref	684
4.11.4.248replace_copy	684
4.11.4.249resetiosflags	684
4.11.4.250return_temporary_buffer	685
4.11.4.251right	685
4.11.4.252scientific	685
4.11.4.253set_new_handler	685
4.11.4.254setbase	685
4.11.4.255setfill	686
4.11.4.256setiosflags	686
4.11.4.257setprecision	686
4.11.4.258setw	687
4.11.4.259showbase	687
4.11.4.260showpoint	687
4.11.4.261showpos	687
4.11.4.262skipws	687
4.11.4.263static_pointer_cast	688
4.11.4.264swap	688
4.11.4.265swap	688
4.11.4.266swap	688
4.11.4.267swap	689
4.11.4.268swap	689
4.11.4.269swap	689
4.11.4.270swap	689
4.11.4.271lswap	690
4.11.4.272swap	690
4.11.4.273swap	690
4.11.4.274swap	690
4.11.4.275tolower	690
4.11.4.276toupper	691
4.11.4.277uninitialized_copy	691

4.11.4.278 uninitialized_copy_n	691
4.11.4.279 uninitialized_fill	692
4.11.4.280 uninitialized_fill_n	692
4.11.4.281 unitbuf	692
4.11.4.282 uppercase	693
4.11.4.283 use_facet	693
4.11.4.284 ws	693
4.11.5 Variable Documentation	694
4.11.5.1 __invoke	694
4.11.5.2 cerr	694
4.11.5.3 cin	694
4.11.5.4 clog	694
4.11.5.5 cout	694
4.11.5.6 wcerr	694
4.11.5.7 wcin	695
4.11.5.8 wclog	695
4.11.5.9 wcout	695
4.12 std::__debug Namespace Reference	695
4.12.1 Detailed Description	701
4.13 std::__detail Namespace Reference	701
4.13.1 Detailed Description	702
4.14 std::__parallel Namespace Reference	702
4.14.1 Detailed Description	727
4.15 std::__profile Namespace Reference	727
4.15.1 Detailed Description	734
4.16 std::chrono Namespace Reference	734
4.16.1 Detailed Description	737
4.16.2 Typedef Documentation	737
4.16.2.1 hours	737
4.16.2.2 microseconds	737
4.16.2.3 milliseconds	737

4.16.2.4	minutes	737
4.16.2.5	nanoseconds	737
4.16.2.6	seconds	738
4.16.3	Function Documentation	738
4.16.3.1	duration_cast	738
4.16.3.2	time_point_cast	738
4.17	std::decimal Namespace Reference	738
4.17.1	Detailed Description	749
4.17.2	Function Documentation	749
4.17.2.1	decimal32_to_long_long	749
4.18	std::placeholders Namespace Reference	749
4.18.1	Detailed Description	749
4.19	std::regex_constants Namespace Reference	750
4.19.1	Detailed Description	751
4.19.2	Typedef Documentation	751
4.19.2.1	match_flag_type	751
4.19.2.2	syntax_option_type	752
4.19.3	Enumeration Type Documentation	752
4.19.3.1	__match_flag	752
4.19.3.2	__syntax_option	752
4.19.3.3	error_type	752
4.19.4	Function Documentation	753
4.19.4.1	error_backref	753
4.19.4.2	error_badbrace	753
4.19.4.3	error_badrepeat	753
4.19.4.4	error_brace	753
4.19.4.5	error_brack	753
4.19.4.6	error_collate	753
4.19.4.7	error_complexity	753
4.19.4.8	error_ctype	754
4.19.4.9	error_escape	754

4.19.4.10 error_paren	754
4.19.4.11 error_range	754
4.19.4.12 error_space	754
4.19.4.13 error_stack	754
4.19.5 Variable Documentation	754
4.19.5.1 awk	754
4.19.5.2 basic	755
4.19.5.3 collate	755
4.19.5.4 ECMAScript	755
4.19.5.5 egrep	755
4.19.5.6 extended	755
4.19.5.7 format_default	756
4.19.5.8 format_first_only	756
4.19.5.9 format_no_copy	756
4.19.5.10 format_sed	757
4.19.5.11 grep	757
4.19.5.12 icase	757
4.19.5.13 match_any	757
4.19.5.14 match_continuous	757
4.19.5.15 match_default	757
4.19.5.16 match_not_bol	758
4.19.5.17 match_not_bow	758
4.19.5.18 match_not_eol	758
4.19.5.19 match_not_eow	758
4.19.5.20 match_not_null	758
4.19.5.21 match_prev_avail	758
4.19.5.22 nosubs	759
4.19.5.23 optimize	759
4.20 std::rel_ops Namespace Reference	759
4.20.1 Detailed Description	759
4.20.2 Function Documentation	760

4.20.2.1	operator!=	760
4.20.2.2	operator<=	760
4.20.2.3	operator>	760
4.20.2.4	operator>=	761
4.21	std::this_thread Namespace Reference	761
4.21.1	Detailed Description	761
4.21.2	Function Documentation	762
4.21.2.1	get_id	762
4.21.2.2	sleep_for	762
4.21.2.3	sleep_until	762
4.21.2.4	yield	762
4.22	std::tr1 Namespace Reference	762
4.22.1	Detailed Description	770
4.23	std::tr1::__detail Namespace Reference	770
4.23.1	Detailed Description	771
5	Class Documentation	773
5.1	__atomic0::atomic_address Struct Reference	773
5.1.1	Detailed Description	774
5.2	__atomic0::atomic_bool Struct Reference	774
5.2.1	Detailed Description	774
5.3	__atomic0::atomic_flag Struct Reference	775
5.3.1	Detailed Description	775
5.4	__atomic2::atomic_address Struct Reference	775
5.4.1	Detailed Description	776
5.5	__atomic2::atomic_bool Struct Reference	776
5.5.1	Detailed Description	777
5.6	__atomic2::atomic_flag Struct Reference	777
5.6.1	Detailed Description	777
5.7	__cxxabiv1::__forced_unwind Class Reference	777
5.7.1	Detailed Description	777

5.8	<code>__gnu_cxx::__common_pool_policy< _PoolTp, _Thread ></code> Struct Template Reference	778
5.8.1	Detailed Description	778
5.9	<code>__gnu_cxx::__detail::__mini_vector< _Tp ></code> Class Template Reference	778
5.9.1	Detailed Description	779
5.10	<code>__gnu_cxx::__detail::__Bitmap_counter< _Tp ></code> Class Template Reference	779
5.10.1	Detailed Description	780
5.11	<code>__gnu_cxx::__detail::__Ffit_finder< _Tp ></code> Class Template Reference	781
5.11.1	Detailed Description	782
5.11.2	Member Typedef Documentation	782
5.11.2.1	<code>argument_type</code>	782
5.11.2.2	<code>result_type</code>	782
5.12	<code>__gnu_cxx::__mt_alloc< _Tp, _Poolp ></code> Class Template Reference	782
5.12.1	Detailed Description	784
5.13	<code>__gnu_cxx::__mt_alloc_base< _Tp ></code> Class Template Reference	784
5.13.1	Detailed Description	785
5.14	<code>__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread ></code> Struct Template Reference	785
5.14.1	Detailed Description	786
5.15	<code>__gnu_cxx::__pool< false ></code> Class Template Reference	786
5.15.1	Detailed Description	787
5.16	<code>__gnu_cxx::__pool< true ></code> Class Template Reference	787
5.16.1	Detailed Description	789
5.17	<code>__gnu_cxx::__pool_alloc< _Tp ></code> Class Template Reference	789
5.17.1	Detailed Description	791
5.18	<code>__gnu_cxx::__pool_alloc_base</code> Class Reference	791
5.18.1	Detailed Description	792
5.19	<code>__gnu_cxx::__pool_base</code> Struct Reference	792
5.19.1	Detailed Description	793
5.20	<code>__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc ></code> Class Template Reference	794

5.20.1 Detailed Description	796
5.21 <code>__gnu_cxx::__scoped_lock</code> Class Reference	797
5.21.1 Detailed Description	797
5.22 <code>__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base></code> Class Template Reference	797
5.22.1 Detailed Description	802
5.22.2 Constructor & Destructor Documentation	802
5.22.2.1 <code>__versa_string</code>	802
5.22.2.2 <code>__versa_string</code>	802
5.22.2.3 <code>__versa_string</code>	802
5.22.2.4 <code>__versa_string</code>	803
5.22.2.5 <code>__versa_string</code>	803
5.22.2.6 <code>__versa_string</code>	803
5.22.2.7 <code>__versa_string</code>	804
5.22.2.8 <code>__versa_string</code>	804
5.22.2.9 <code>__versa_string</code>	805
5.22.2.10 <code>__versa_string</code>	805
5.22.2.11 <code>__versa_string</code>	805
5.22.2.12 <code>~__versa_string</code>	806
5.22.3 Member Function Documentation	806
5.22.3.1 <code>append</code>	806
5.22.3.2 <code>append</code>	806
5.22.3.3 <code>append</code>	807
5.22.3.4 <code>append</code>	807
5.22.3.5 <code>append</code>	808
5.22.3.6 <code>append</code>	808
5.22.3.7 <code>append</code>	809
5.22.3.8 <code>assign</code>	809
5.22.3.9 <code>assign</code>	809
5.22.3.10 <code>assign</code>	810
5.22.3.11 <code>assign</code>	810

5.22.3.12 assign	811
5.22.3.13 assign	811
5.22.3.14 assign	812
5.22.3.15 assign	812
5.22.3.16 at	813
5.22.3.17 at	813
5.22.3.18 back	814
5.22.3.19 back	814
5.22.3.20 begin	814
5.22.3.21 begin	814
5.22.3.22 c_str	815
5.22.3.23 capacity	815
5.22.3.24 cbegin	815
5.22.3.25 cend	815
5.22.3.26 clear	816
5.22.3.27 compare	816
5.22.3.28 compare	816
5.22.3.29 compare	817
5.22.3.30 compare	818
5.22.3.31 compare	819
5.22.3.32 compare	819
5.22.3.33 copy	820
5.22.3.34 crbegin	821
5.22.3.35 crend	821
5.22.3.36 data	821
5.22.3.37 empty	822
5.22.3.38 end	822
5.22.3.39 end	822
5.22.3.40 erase	822
5.22.3.41 erase	823
5.22.3.42 erase	823

5.22.3.43 find	824
5.22.3.44 find	824
5.22.3.45 find	825
5.22.3.46 find	825
5.22.3.47 find_first_not_of	826
5.22.3.48 find_first_not_of	827
5.22.3.49 find_first_not_of	827
5.22.3.50 find_first_not_of	828
5.22.3.51 find_first_of	828
5.22.3.52 find_first_of	829
5.22.3.53 find_first_of	829
5.22.3.54 find_first_of	830
5.22.3.55 find_last_not_of	830
5.22.3.56 find_last_not_of	831
5.22.3.57 find_last_not_of	832
5.22.3.58 find_last_not_of	832
5.22.3.59 find_last_of	833
5.22.3.60 find_last_of	833
5.22.3.61 find_last_of	834
5.22.3.62 find_last_of	834
5.22.3.63 front	835
5.22.3.64 front	835
5.22.3.65 get_allocator	835
5.22.3.66 insert	835
5.22.3.67 insert	836
5.22.3.68 insert	837
5.22.3.69 insert	837
5.22.3.70 insert	838
5.22.3.71 insert	838
5.22.3.72 insert	839
5.22.3.73 insert	839

5.22.3.74 insert	840
5.22.3.75 length	841
5.22.3.76 max_size	841
5.22.3.77 operator+=	841
5.22.3.78 operator+=	841
5.22.3.79 operator+=	842
5.22.3.80 operator+=	842
5.22.3.81 operator=	843
5.22.3.82 operator=	843
5.22.3.83 operator=	843
5.22.3.84 operator=	844
5.22.3.85 operator=	844
5.22.3.86 operator[]	844
5.22.3.87 operator[]	845
5.22.3.88 push_back	845
5.22.3.89 rbegin	845
5.22.3.90 rbegin	846
5.22.3.91 rend	846
5.22.3.92 rend	846
5.22.3.93 replace	846
5.22.3.94 replace	847
5.22.3.95 replace	848
5.22.3.96 replace	849
5.22.3.97 replace	849
5.22.3.98 replace	850
5.22.3.99 replace	851
5.22.3.100replace	851
5.22.3.101replace	852
5.22.3.102replace	853
5.22.3.103replace	853
5.22.3.104reserve	854

5.22.3.105	<code>resize</code>	855
5.22.3.106	<code>resize</code>	855
5.22.3.107	<code>rfind</code>	856
5.22.3.108	<code>rfind</code>	856
5.22.3.109	<code>rfind</code>	857
5.22.3.110	<code>rfind</code>	857
5.22.3.111	<code>shrink_to_fit</code>	858
5.22.3.112	<code>size</code>	858
5.22.3.113	<code>substr</code>	858
5.22.3.114	<code>swap</code>	859
5.22.4	Member Data Documentation	859
5.22.4.1	<code>npos</code>	859
5.23	<code>__gnu_cxx::_Caster<_ToType></code> Struct Template Reference	860
5.23.1	Detailed Description	860
5.24	<code>__gnu_cxx::_Char_types<_CharT></code> Struct Template Reference	860
5.24.1	Detailed Description	861
5.25	<code>__gnu_cxx::_ExtPtr_allocator<_Tp></code> Class Template Reference	861
5.25.1	Detailed Description	862
5.26	<code>__gnu_cxx::_Invalid_type</code> Struct Reference	863
5.26.1	Detailed Description	863
5.27	<code>__gnu_cxx::_Pointer_adapter<_Storage_policy></code> Class Template Reference	863
5.27.1	Detailed Description	865
5.28	<code>__gnu_cxx::_Relative_pointer_impl<_Tp></code> Class Template Reference	866
5.28.1	Detailed Description	866
5.29	<code>__gnu_cxx::_Relative_pointer_impl<const _Tp></code> Class Template Reference	867
5.29.1	Detailed Description	867
5.30	<code>__gnu_cxx::_Std_pointer_impl<_Tp></code> Class Template Reference	867
5.30.1	Detailed Description	868
5.31	<code>__gnu_cxx::_Unqualified_type<_Tp></code> Struct Template Reference	868
5.31.1	Detailed Description	868

5.32	<code>__gnu_cxx::annotate_base</code> Struct Reference	869
5.32.1	Detailed Description	869
5.33	<code>__gnu_cxx::array_allocator< _Tp, _Array ></code> Class Template Reference	870
5.33.1	Detailed Description	871
5.34	<code>__gnu_cxx::array_allocator_base< _Tp ></code> Class Template Reference	871
5.34.1	Detailed Description	872
5.35	<code>__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 ></code> Class Template Reference	872
5.35.1	Detailed Description	873
5.35.2	Member Typedef Documentation	874
5.35.2.1	<code>argument_type</code>	874
5.35.2.2	<code>result_type</code>	874
5.36	<code>__gnu_cxx::bitmap_allocator< _Tp ></code> Class Template Reference	874
5.36.1	Detailed Description	876
5.36.2	Member Function Documentation	876
5.36.2.1	<code>_M_allocate_single_object</code>	876
5.36.2.2	<code>_M_deallocate_single_object</code>	876
5.37	<code>__gnu_cxx::char_traits< _CharT ></code> Struct Template Reference	877
5.37.1	Detailed Description	878
5.38	<code>__gnu_cxx::character< V, I, S ></code> Struct Template Reference	878
5.38.1	Detailed Description	879
5.39	<code>__gnu_cxx::condition_base</code> Struct Reference	879
5.39.1	Detailed Description	880
5.40	<code>__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 ></code> Struct Template Reference	880
5.40.1	Detailed Description	881
5.41	<code>__gnu_cxx::constant_unary_fun< _Result, _Argument ></code> Struct Template Reference	881
5.41.1	Detailed Description	882
5.42	<code>__gnu_cxx::constant_void_fun< _Result ></code> Struct Template Reference	882
5.42.1	Detailed Description	882
5.43	<code>__gnu_cxx::debug_allocator< _Alloc ></code> Class Template Reference	883

5.43.1 Detailed Description	883
5.44 <code>__gnu_cxx::enc_filebuf<_CharT></code> Class Template Reference	884
5.44.1 Detailed Description	887
5.44.2 Member Typedef Documentation	887
5.44.2.1 <code>__streambuf_type</code>	887
5.44.2.2 <code>char_type</code>	888
5.44.2.3 <code>int_type</code>	888
5.44.2.4 <code>off_type</code>	888
5.44.2.5 <code>pos_type</code>	888
5.44.2.6 <code>traits_type</code>	889
5.44.3 Member Function Documentation	889
5.44.3.1 <code>_M_create_pback</code>	889
5.44.3.2 <code>_M_destroy_pback</code>	889
5.44.3.3 <code>_M_set_buffer</code>	889
5.44.3.4 <code>close</code>	890
5.44.3.5 <code>eback</code>	890
5.44.3.6 <code>egptr</code>	890
5.44.3.7 <code>epptr</code>	891
5.44.3.8 <code>gbump</code>	891
5.44.3.9 <code>getloc</code>	891
5.44.3.10 <code>gptr</code>	892
5.44.3.11 <code>imbue</code>	892
5.44.3.12 <code>in_avail</code>	893
5.44.3.13 <code>is_open</code>	893
5.44.3.14 <code>open</code>	893
5.44.3.15 <code>open</code>	894
5.44.3.16 <code>overflow</code>	894
5.44.3.17 <code>pbackfail</code>	895
5.44.3.18 <code>pbase</code>	895
5.44.3.19 <code>pbump</code>	895
5.44.3.20 <code>pptr</code>	896

5.44.3.21 pubimbue	896
5.44.3.22 pubseekoff	896
5.44.3.23 pubseekpos	897
5.44.3.24 pubsetbuf	897
5.44.3.25 pubsync	897
5.44.3.26 sbumpc	898
5.44.3.27 seekoff	898
5.44.3.28 seekpos	898
5.44.3.29 setbuf	899
5.44.3.30 setbuf	899
5.44.3.31 setg	900
5.44.3.32 setp	900
5.44.3.33 sgetc	900
5.44.3.34 sgetn	901
5.44.3.35 showmanyc	901
5.44.3.36 snextc	901
5.44.3.37 sputbackc	902
5.44.3.38 sputc	902
5.44.3.39 sputn	903
5.44.3.40 stossc	903
5.44.3.41 sungetc	903
5.44.3.42 sync	904
5.44.3.43 uflow	904
5.44.3.44 underflow	904
5.44.3.45 xsgetn	905
5.44.3.46 xsputn	905
5.44.4 Member Data Documentation	906
5.44.4.1 _M_buf	906
5.44.4.2 _M_buf_locale	906
5.44.4.3 _M_buf_size	906
5.44.4.4 _M_ext_buf	906

5.44.4.5	<code>_M_ext_buf_size</code>	907
5.44.4.6	<code>_M_ext_next</code>	907
5.44.4.7	<code>_M_in_beg</code>	907
5.44.4.8	<code>_M_in_cur</code>	907
5.44.4.9	<code>_M_in_end</code>	908
5.44.4.10	<code>_M_mode</code>	908
5.44.4.11	<code>_M_out_beg</code>	908
5.44.4.12	<code>_M_out_cur</code>	908
5.44.4.13	<code>_M_out_end</code>	909
5.44.4.14	<code>_M_pback</code>	909
5.44.4.15	<code>_M_pback_cur_save</code>	909
5.44.4.16	<code>_M_pback_end_save</code>	910
5.44.4.17	<code>_M_pback_init</code>	910
5.44.4.18	<code>_M_reading</code>	910
5.45	<code>__gnu_cxx::encoding_char_traits<_CharT></code> Struct Template Reference	910
5.45.1	Detailed Description	912
5.46	<code>__gnu_cxx::encoding_state</code> Class Reference	912
5.46.1	Detailed Description	913
5.47	<code>__gnu_cxx::forced_error</code> Struct Reference	913
5.47.1	Detailed Description	914
5.47.2	Member Function Documentation	914
5.47.2.1	<code>what</code>	914
5.48	<code>__gnu_cxx::free_list</code> Class Reference	915
5.48.1	Detailed Description	915
5.48.2	Member Function Documentation	916
5.48.2.1	<code>_M_clear</code>	916
5.48.2.2	<code>_M_get</code>	916
5.48.2.3	<code>_M_insert</code>	916
5.49	<code>__gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc></code> Class Template Reference	916
5.49.1	Detailed Description	918

5.50	<code>__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc ></code> Class Template Reference	919
5.50.1	Detailed Description	920
5.51	<code>__gnu_cxx::hash_multiset< _Value, _HashFn, _EqualKey, _Alloc ></code> Class Template Reference	921
5.51.1	Detailed Description	922
5.52	<code>__gnu_cxx::hash_set< _Value, _HashFn, _EqualKey, _Alloc ></code> Class Template Reference	923
5.52.1	Detailed Description	924
5.53	<code>__gnu_cxx::limit_condition</code> Struct Reference	925
5.53.1	Detailed Description	926
5.54	<code>__gnu_cxx::limit_condition::always_adjustor</code> Struct Reference	926
5.54.1	Detailed Description	926
5.55	<code>__gnu_cxx::limit_condition::limit_adjustor</code> Struct Reference	926
5.55.1	Detailed Description	926
5.56	<code>__gnu_cxx::limit_condition::never_adjustor</code> Struct Reference	927
5.56.1	Detailed Description	927
5.57	<code>__gnu_cxx::malloc_allocator< _Tp ></code> Class Template Reference	927
5.57.1	Detailed Description	928
5.58	<code>__gnu_cxx::new_allocator< _Tp ></code> Class Template Reference	928
5.58.1	Detailed Description	930
5.59	<code>__gnu_cxx::project1st< _Arg1, _Arg2 ></code> Struct Template Reference	930
5.59.1	Detailed Description	931
5.59.2	Member Typedef Documentation	931
5.59.2.1	<code>first_argument_type</code>	931
5.59.2.2	<code>result_type</code>	931
5.59.2.3	<code>second_argument_type</code>	931
5.60	<code>__gnu_cxx::project2nd< _Arg1, _Arg2 ></code> Struct Template Reference	931
5.60.1	Detailed Description	932
5.60.2	Member Typedef Documentation	932
5.60.2.1	<code>first_argument_type</code>	932
5.60.2.2	<code>result_type</code>	932

5.60.2.3	<code>second_argument_type</code>	932
5.61	<code>__gnu_cxx::random_condition</code> Struct Reference	933
5.61.1	Detailed Description	934
5.62	<code>__gnu_cxx::random_condition::always_adjustor</code> Struct Reference	934
5.62.1	Detailed Description	934
5.63	<code>__gnu_cxx::random_condition::group_adjustor</code> Struct Reference	934
5.63.1	Detailed Description	934
5.64	<code>__gnu_cxx::random_condition::never_adjustor</code> Struct Reference	935
5.64.1	Detailed Description	935
5.65	<code>__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc ></code> Struct Template Reference	935
5.65.1	Detailed Description	938
5.66	<code>__gnu_cxx::recursive_init_error</code> Class Reference	938
5.66.1	Detailed Description	939
5.66.2	Member Function Documentation	939
5.66.2.1	<code>what</code>	939
5.67	<code>__gnu_cxx::rope< _CharT, _Alloc ></code> Class Template Reference	940
5.67.1	Detailed Description	946
5.68	<code>__gnu_cxx::select1st< _Pair ></code> Struct Template Reference	946
5.68.1	Detailed Description	947
5.68.2	Member Typedef Documentation	947
5.68.2.1	<code>argument_type</code>	947
5.68.2.2	<code>result_type</code>	947
5.69	<code>__gnu_cxx::select2nd< _Pair ></code> Struct Template Reference	947
5.69.1	Detailed Description	948
5.69.2	Member Typedef Documentation	948
5.69.2.1	<code>argument_type</code>	948
5.69.2.2	<code>result_type</code>	948
5.70	<code>__gnu_cxx::slist< _Tp, _Alloc ></code> Class Template Reference	948
5.70.1	Detailed Description	951
5.71	<code>__gnu_cxx::stdio_filebuf< _CharT, _Traits ></code> Class Template Reference	951

5.71.1	Detailed Description	956
5.71.2	Member Typedef Documentation	956
5.71.2.1	__streambuf_type	956
5.71.2.2	char_type	956
5.71.2.3	int_type	956
5.71.2.4	off_type	957
5.71.2.5	pos_type	957
5.71.2.6	traits_type	957
5.71.3	Constructor & Destructor Documentation	957
5.71.3.1	stdio_filebuf	957
5.71.3.2	stdio_filebuf	958
5.71.3.3	stdio_filebuf	958
5.71.3.4	~stdio_filebuf	959
5.71.4	Member Function Documentation	959
5.71.4.1	_M_create_pback	959
5.71.4.2	_M_destroy_pback	959
5.71.4.3	_M_set_buffer	959
5.71.4.4	close	960
5.71.4.5	eback	960
5.71.4.6	egptr	960
5.71.4.7	epptr	961
5.71.4.8	fd	961
5.71.4.9	file	962
5.71.4.10	gbump	962
5.71.4.11	getloc	962
5.71.4.12	gptr	962
5.71.4.13	imbue	963
5.71.4.14	in_avail	964
5.71.4.15	is_open	964
5.71.4.16	open	964
5.71.4.17	open	965

5.71.4.18 overflow	965
5.71.4.19 pbackfail	966
5.71.4.20 pbase	967
5.71.4.21 pbump	967
5.71.4.22 pptr	967
5.71.4.23 pubimbue	968
5.71.4.24 pubseekoff	968
5.71.4.25 pubseekpos	969
5.71.4.26 pubsetbuf	969
5.71.4.27 pubsync	969
5.71.4.28 sbumpc	970
5.71.4.29 seekoff	970
5.71.4.30 seekpos	970
5.71.4.31 setbuf	971
5.71.4.32 setbuf	971
5.71.4.33 setg	972
5.71.4.34 setp	972
5.71.4.35 sgetc	972
5.71.4.36 sgetn	973
5.71.4.37 showmanyc	973
5.71.4.38 snextc	974
5.71.4.39 sputbackc	974
5.71.4.40 sputc	975
5.71.4.41 sputn	975
5.71.4.42 stosc	975
5.71.4.43 sungetc	976
5.71.4.44 sync	976
5.71.4.45 uflow	977
5.71.4.46 underflow	977
5.71.4.47 xsgetn	978
5.71.4.48 xspn	978

5.71.5	Member Data Documentation	979
5.71.5.1	_M_buf	979
5.71.5.2	_M_buf_locale	979
5.71.5.3	_M_buf_size	979
5.71.5.4	_M_ext_buf	980
5.71.5.5	_M_ext_buf_size	980
5.71.5.6	_M_ext_next	980
5.71.5.7	_M_in_beg	980
5.71.5.8	_M_in_cur	981
5.71.5.9	_M_in_end	981
5.71.5.10	_M_mode	981
5.71.5.11	_M_out_beg	982
5.71.5.12	_M_out_cur	982
5.71.5.13	_M_out_end	982
5.71.5.14	_M_pback	983
5.71.5.15	_M_pback_cur_save	983
5.71.5.16	_M_pback_end_save	983
5.71.5.17	_M_pback_init	984
5.71.5.18	_M_reading	984
5.72	__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits > Class Template Reference	984
5.72.1	Detailed Description	988
5.72.2	Member Typedef Documentation	988
5.72.2.1	__streambuf_type	988
5.72.2.2	char_type	989
5.72.2.3	int_type	989
5.72.2.4	off_type	989
5.72.2.5	pos_type	989
5.72.2.6	traits_type	990
5.72.3	Member Function Documentation	990
5.72.3.1	eback	990

5.72.3.2	egptr	990
5.72.3.3	eptr	991
5.72.3.4	file	991
5.72.3.5	gbump	991
5.72.3.6	getloc	992
5.72.3.7	gptr	992
5.72.3.8	imbue	992
5.72.3.9	in_avail	993
5.72.3.10	overflow	993
5.72.3.11	pbackfail	994
5.72.3.12	pbase	995
5.72.3.13	pbump	995
5.72.3.14	pptr	995
5.72.3.15	pubimbue	996
5.72.3.16	pubseekoff	996
5.72.3.17	pubseekpos	997
5.72.3.18	pubsetbuf	997
5.72.3.19	pubsync	997
5.72.3.20	sbumpc	998
5.72.3.21	seekoff	998
5.72.3.22	seekpos	998
5.72.3.23	setbuf	999
5.72.3.24	setg	999
5.72.3.25	setp	999
5.72.3.26	sgetc	1000
5.72.3.27	sgetn	1000
5.72.3.28	showmanyc	1001
5.72.3.29	snextc	1001
5.72.3.30	sputbackc	1002
5.72.3.31	sputc	1002
5.72.3.32	sputn	1003

5.72.3.33	stossc	1003
5.72.3.34	sungetc	1003
5.72.3.35	sync	1004
5.72.3.36	uflow	1004
5.72.3.37	underflow	1004
5.72.3.38	xsgetn	1005
5.72.3.39	xspn	1006
5.72.4	Member Data Documentation	1006
5.72.4.1	_M_buf_locale	1006
5.72.4.2	_M_in_beg	1006
5.72.4.3	_M_in_cur	1007
5.72.4.4	_M_in_end	1007
5.72.4.5	_M_out_beg	1007
5.72.4.6	_M_out_cur	1008
5.72.4.7	_M_out_end	1008
5.73	__gnu_cxx::subtractive_rng Class Reference	1009
5.73.1	Detailed Description	1009
5.73.2	Member Typedef Documentation	1010
5.73.2.1	argument_type	1010
5.73.2.2	result_type	1010
5.73.3	Constructor & Destructor Documentation	1010
5.73.3.1	subtractive_rng	1010
5.73.3.2	subtractive_rng	1010
5.73.4	Member Function Documentation	1010
5.73.4.1	operator()	1010
5.74	__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp> Struct Template Reference	1011
5.74.1	Detailed Description	1012
5.74.2	Constructor & Destructor Documentation	1012
5.74.2.1	temporary_buffer	1012
5.74.2.2	~temporary_buffer	1012

5.74.3	Member Function Documentation	1013
5.74.3.1	begin	1013
5.74.3.2	end	1013
5.74.3.3	requested_size	1013
5.74.3.4	size	1013
5.75	<code>__gnu_cxx::throw_allocator_base< _Tp, _Cond ></code> Class Template Reference	1014
5.75.1	Detailed Description	1015
5.76	<code>__gnu_cxx::throw_allocator_limit< _Tp ></code> Struct Template Reference	1015
5.76.1	Detailed Description	1016
5.77	<code>__gnu_cxx::throw_allocator_random< _Tp ></code> Struct Template Reference	1017
5.77.1	Detailed Description	1018
5.78	<code>__gnu_cxx::throw_value_base< _Cond ></code> Struct Template Reference	1018
5.78.1	Detailed Description	1019
5.79	<code>__gnu_cxx::throw_value_limit</code> Struct Reference	1019
5.79.1	Detailed Description	1021
5.80	<code>__gnu_cxx::throw_value_random</code> Struct Reference	1021
5.80.1	Detailed Description	1022
5.81	<code>__gnu_cxx::unary_compose< _Operation1, _Operation2 ></code> Class Template Reference	1022
5.81.1	Detailed Description	1023
5.81.2	Member Typedef Documentation	1024
5.81.2.1	argument_type	1024
5.81.2.2	result_type	1024
5.82	<code>__gnu_debug::__is_same< _Type1, _Type2 ></code> Struct Template Refer- ence	1024
5.82.1	Detailed Description	1024
5.83	<code>__gnu_debug::__After_nth_from< _Iterator ></code> Class Template Reference	1025
5.83.1	Detailed Description	1025
5.84	<code>__gnu_debug::__BeforeBeginHelper< _Sequence ></code> Struct Template Reference	1025
5.84.1	Detailed Description	1025

5.85	<code>__gnu_debug::_Not_equal_to<_Type></code> Class Template Reference	1026
5.85.1	Detailed Description	1026
5.86	<code>__gnu_debug::_Safe_iterator<_Iterator, _Sequence></code> Class Template Reference	1026
5.86.1	Detailed Description	1029
5.86.2	Constructor & Destructor Documentation	1029
5.86.2.1	<code>_Safe_iterator</code>	1029
5.86.2.2	<code>_Safe_iterator</code>	1029
5.86.2.3	<code>_Safe_iterator</code>	1030
5.86.2.4	<code>_Safe_iterator</code>	1030
5.86.3	Member Function Documentation	1030
5.86.3.1	<code>_M_attach</code>	1030
5.86.3.2	<code>_M_attach</code>	1031
5.86.3.3	<code>_M_attach_single</code>	1031
5.86.3.4	<code>_M_attach_single</code>	1031
5.86.3.5	<code>_M_attached_to</code>	1031
5.86.3.6	<code>_M_before_dereferenceable</code>	1031
5.86.3.7	<code>_M_can_compare</code>	1032
5.86.3.8	<code>_M_dereferenceable</code>	1032
5.86.3.9	<code>_M_detach</code>	1032
5.86.3.10	<code>_M_detach_single</code>	1032
5.86.3.11	<code>_M_get_distance</code>	1032
5.86.3.12	<code>_M_get_mutex</code>	1033
5.86.3.13	<code>_M_incrementable</code>	1033
5.86.3.14	<code>_M_invalidate</code>	1033
5.86.3.15	<code>_M_invalidate_single</code>	1033
5.86.3.16	<code>_M_is_before_begin</code>	1034
5.86.3.17	<code>_M_is_begin</code>	1034
5.86.3.18	<code>_M_is_end</code>	1034
5.86.3.19	<code>_M_singular</code>	1034
5.86.3.20	<code>base</code>	1035

5.86.3.21	<code>operator_iterator</code>	1035
5.86.3.22	<code>operator*</code>	1035
5.86.3.23	<code>operator++</code>	1035
5.86.3.24	<code>operator++</code>	1036
5.86.3.25	<code>operator--</code>	1036
5.86.3.26	<code>operator--</code>	1036
5.86.3.27	<code>operator-></code>	1037
5.86.3.28	<code>operator=</code>	1037
5.86.4	Member Data Documentation	1037
5.86.4.1	<code>_M_next</code>	1037
5.86.4.2	<code>_M_prior</code>	1038
5.86.4.3	<code>_M_sequence</code>	1038
5.86.4.4	<code>_M_version</code>	1038
5.87	<code>__gnu_debug::Safe_iterator_base</code> Class Reference	1038
5.87.1	Detailed Description	1040
5.87.2	Constructor & Destructor Documentation	1040
5.87.2.1	<code>_Safe_iterator_base</code>	1040
5.87.2.2	<code>_Safe_iterator_base</code>	1040
5.87.2.3	<code>_Safe_iterator_base</code>	1040
5.87.3	Member Function Documentation	1041
5.87.3.1	<code>_M_attach</code>	1041
5.87.3.2	<code>_M_attach_single</code>	1041
5.87.3.3	<code>_M_attached_to</code>	1041
5.87.3.4	<code>_M_can_compare</code>	1041
5.87.3.5	<code>_M_detach</code>	1041
5.87.3.6	<code>_M_detach_single</code>	1041
5.87.3.7	<code>_M_get_mutex</code>	1042
5.87.3.8	<code>_M_singular</code>	1042
5.87.4	Member Data Documentation	1042
5.87.4.1	<code>_M_next</code>	1042
5.87.4.2	<code>_M_prior</code>	1042

5.87.4.3	_M_sequence	1042
5.87.4.4	_M_version	1043
5.88	__gnu_debug::_Safe_sequence< _Sequence > Class Template Reference	1043
5.88.1	Detailed Description	1044
5.88.2	Member Function Documentation	1044
5.88.2.1	_M_detach_all	1044
5.88.2.2	_M_detach_singular	1045
5.88.2.3	_M_get_mutex	1045
5.88.2.4	_M_invalidate_all	1045
5.88.2.5	_M_invalidate_if	1045
5.88.2.6	_M_revalidate_singular	1045
5.88.2.7	_M_swap	1046
5.88.2.8	_M_transfer_iter	1046
5.88.3	Member Data Documentation	1046
5.88.3.1	_M_const_iterators	1046
5.88.3.2	_M_iterators	1046
5.88.3.3	_M_version	1047
5.89	__gnu_debug::_Safe_sequence_base Class Reference	1047
5.89.1	Detailed Description	1048
5.89.2	Constructor & Destructor Documentation	1048
5.89.2.1	~_Safe_sequence_base	1048
5.89.3	Member Function Documentation	1048
5.89.3.1	_M_detach_all	1048
5.89.3.2	_M_detach_singular	1049
5.89.3.3	_M_get_mutex	1049
5.89.3.4	_M_invalidate_all	1049
5.89.3.5	_M_revalidate_singular	1049
5.89.3.6	_M_swap	1049
5.89.4	Member Data Documentation	1049
5.89.4.1	_M_const_iterators	1049

5.89.4.2	_M_iterators	1050
5.89.4.3	_M_version	1050
5.90	__gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class	
	Template Reference	1050
5.90.1	Detailed Description	1058
5.90.2	Member Function Documentation	1058
5.90.2.1	_M_detach_all	1058
5.90.2.2	_M_detach_singular	1059
5.90.2.3	_M_get_mutex	1059
5.90.2.4	_M_invalidate_all	1059
5.90.2.5	_M_invalidate_if	1059
5.90.2.6	_M_revalidate_singular	1059
5.90.2.7	_M_swap	1060
5.90.2.8	_M_transfer_iter	1060
5.90.2.9	append	1060
5.90.2.10	append	1060
5.90.2.11	append	1061
5.90.2.12	append	1061
5.90.2.13	append	1061
5.90.2.14	append	1062
5.90.2.15	append	1062
5.90.2.16	assign	1063
5.90.2.17	assign	1063
5.90.2.18	assign	1063
5.90.2.19	assign	1064
5.90.2.20	assign	1064
5.90.2.21	assign	1065
5.90.2.22	assign	1065
5.90.2.23	assign	1065
5.90.2.24	at	1066
5.90.2.25	at	1066

5.90.2.26 back	1067
5.90.2.27 back	1067
5.90.2.28 begin	1067
5.90.2.29 begin	1067
5.90.2.30 c_str	1067
5.90.2.31 capacity	1067
5.90.2.32 cbegin	1068
5.90.2.33 cend	1068
5.90.2.34 clear	1068
5.90.2.35 compare	1068
5.90.2.36 compare	1069
5.90.2.37 compare	1069
5.90.2.38 compare	1070
5.90.2.39 compare	1070
5.90.2.40 compare	1071
5.90.2.41 copy	1071
5.90.2.42 crbegin	1072
5.90.2.43 crend	1072
5.90.2.44 data	1072
5.90.2.45 empty	1072
5.90.2.46 end	1073
5.90.2.47 end	1073
5.90.2.48 erase	1073
5.90.2.49 erase	1073
5.90.2.50 erase	1074
5.90.2.51 find	1074
5.90.2.52 find	1075
5.90.2.53 find	1075
5.90.2.54 find	1075
5.90.2.55 find_first_not_of	1076
5.90.2.56 find_first_not_of	1076

5.90.2.57 find_first_not_of	1077
5.90.2.58 find_first_not_of	1077
5.90.2.59 find_first_of	1077
5.90.2.60 find_first_of	1078
5.90.2.61 find_first_of	1078
5.90.2.62 find_first_of	1079
5.90.2.63 find_last_not_of	1079
5.90.2.64 find_last_not_of	1080
5.90.2.65 find_last_not_of	1080
5.90.2.66 find_last_not_of	1080
5.90.2.67 find_last_of	1081
5.90.2.68 find_last_of	1081
5.90.2.69 find_last_of	1082
5.90.2.70 find_last_of	1082
5.90.2.71 front	1083
5.90.2.72 front	1083
5.90.2.73 get_allocator	1083
5.90.2.74 insert	1083
5.90.2.75 insert	1084
5.90.2.76 insert	1084
5.90.2.77 insert	1085
5.90.2.78 insert	1085
5.90.2.79 insert	1086
5.90.2.80 insert	1086
5.90.2.81 insert	1087
5.90.2.82 insert	1087
5.90.2.83 length	1088
5.90.2.84 max_size	1088
5.90.2.85 operator+=	1088
5.90.2.86 operator+=	1088
5.90.2.87 operator+=	1089

5.90.2.88 operator+=	1089
5.90.2.89 operator[]	1089
5.90.2.90 operator[]	1090
5.90.2.91 push_back	1090
5.90.2.92 rbegin	1090
5.90.2.93 rbegin	1091
5.90.2.94 rend	1091
5.90.2.95 rend	1091
5.90.2.96 replace	1091
5.90.2.97 replace	1092
5.90.2.98 replace	1092
5.90.2.99 replace	1093
5.90.2.100replace	1093
5.90.2.101replace	1094
5.90.2.102replace	1095
5.90.2.103replace	1095
5.90.2.104replace	1096
5.90.2.105replace	1096
5.90.2.106replace	1097
5.90.2.107reserve	1098
5.90.2.108resize	1098
5.90.2.109resize	1098
5.90.2.110find	1099
5.90.2.111lrfind	1099
5.90.2.112rfind	1100
5.90.2.113rfind	1100
5.90.2.114shrink_to_fit	1100
5.90.2.115size	1101
5.90.2.116substr	1101
5.90.2.117swap	1101
5.90.3 Member Data Documentation	1102

5.90.3.1	_M_const_iterators	1102
5.90.3.2	_M_iterators	1102
5.90.3.3	_M_version	1102
5.90.3.4	npos	1102
5.91	__gnu_parallel::__accumulate_binop_reduct<_BinOp> Struct Template Reference	1103
5.91.1	Detailed Description	1103
5.92	__gnu_parallel::__accumulate_selector<_It> Struct Template Reference	1103
5.92.1	Detailed Description	1104
5.92.2	Member Function Documentation	1104
5.92.2.1	operator()	1104
5.92.3	Member Data Documentation	1105
5.92.3.1	_M_finish_iterator	1105
5.93	__gnu_parallel::__adjacent_difference_selector<_It> Struct Template Reference	1105
5.93.1	Detailed Description	1106
5.93.2	Member Data Documentation	1106
5.93.2.1	_M_finish_iterator	1106
5.94	__gnu_parallel::__adjacent_find_selector Struct Reference	1107
5.94.1	Detailed Description	1107
5.94.2	Member Function Documentation	1108
5.94.2.1	_M_sequential_algorithm	1108
5.94.2.2	operator()	1108
5.95	__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType> Class Template Reference	1108
5.95.1	Detailed Description	1109
5.95.2	Member Typedef Documentation	1110
5.95.2.1	argument_type	1110
5.95.2.2	result_type	1110
5.96	__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType> Class Template Reference	1110
5.96.1	Detailed Description	1111

5.96.2	Member Typedef Documentation	1112
5.96.2.1	argument_type	1112
5.96.2.2	result_type	1112
5.97	<code>__gnu_parallel::__count_if_selector< _It, _Diff ></code> Struct Template Reference	1112
5.97.1	Detailed Description	1113
5.97.2	Member Function Documentation	1113
5.97.2.1	operator()	1113
5.97.3	Member Data Documentation	1113
5.97.3.1	<code>_M_finish_iterator</code>	1113
5.98	<code>__gnu_parallel::__count_selector< _It, _Diff ></code> Struct Template Reference	1114
5.98.1	Detailed Description	1115
5.98.2	Member Function Documentation	1115
5.98.2.1	operator()	1115
5.98.3	Member Data Documentation	1115
5.98.3.1	<code>_M_finish_iterator</code>	1115
5.99	<code>__gnu_parallel::__fill_selector< _It ></code> Struct Template Reference	1116
5.99.1	Detailed Description	1116
5.99.2	Member Function Documentation	1117
5.99.2.1	operator()	1117
5.99.3	Member Data Documentation	1117
5.99.3.1	<code>_M_finish_iterator</code>	1117
5.100	<code>__gnu_parallel::__find_first_of_selector< _FIterator ></code> Struct Template Reference	1117
5.100.1	Detailed Description	1118
5.100.2	Member Function Documentation	1119
5.100.2.1	<code>_M_sequential_algorithm</code>	1119
5.100.2.2	operator()	1119
5.101	<code>__gnu_parallel::__find_if_selector</code> Struct Reference	1119
5.101.1	Detailed Description	1120
5.101.2	Member Function Documentation	1120

5.101.2.1 <code>_M_sequential_algorithm</code>	1120
5.101.2.2 <code>operator()</code>	1121
5.102 <code>__gnu_parallel::__for_each_selector<_It></code> Struct Template Reference	1121
5.102.1 Detailed Description	1122
5.102.2 Member Function Documentation	1122
5.102.2.1 <code>operator()</code>	1122
5.102.3 Member Data Documentation	1123
5.102.3.1 <code>_M_finish_iterator</code>	1123
5.103 <code>__gnu_parallel::__generate_selector<_It></code> Struct Template Reference	1123
5.103.1 Detailed Description	1124
5.103.2 Member Function Documentation	1124
5.103.2.1 <code>operator()</code>	1124
5.103.3 Member Data Documentation	1124
5.103.3.1 <code>_M_finish_iterator</code>	1124
5.104 <code>__gnu_parallel::__generic_find_selector</code> Struct Reference	1125
5.104.1 Detailed Description	1125
5.105 <code>__gnu_parallel::__generic_for_each_selector<_It></code> Struct Template Reference	1125
5.105.1 Detailed Description	1126
5.105.2 Member Data Documentation	1127
5.105.2.1 <code>_M_finish_iterator</code>	1127
5.106 <code>__gnu_parallel::__identity_selector<_It></code> Struct Template Reference	1127
5.106.1 Detailed Description	1128
5.106.2 Member Function Documentation	1128
5.106.2.1 <code>operator()</code>	1128
5.106.3 Member Data Documentation	1128
5.106.3.1 <code>_M_finish_iterator</code>	1128
5.107 <code>__gnu_parallel::__inner_product_selector<_It, _It2, _Tp></code> Struct Template Reference	1129
5.107.1 Detailed Description	1129
5.107.2 Constructor & Destructor Documentation	1130
5.107.2.1 <code>__inner_product_selector</code>	1130

5.107.3 Member Function Documentation	1130
5.107.3.1 operator()	1130
5.107.4 Member Data Documentation	1130
5.107.4.1 __begin1_iterator	1130
5.107.4.2 __begin2_iterator	1131
5.107.4.3 _M_finish_iterator	1131
5.108 __gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference	1131
5.108.1 Detailed Description	1132
5.109 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference	1132
5.109.1 Detailed Description	1132
5.110 __gnu_parallel::__mismatch_selector Struct Reference	1133
5.110.1 Detailed Description	1133
5.110.2 Member Function Documentation	1134
5.110.2.1 _M_sequential_algorithm	1134
5.110.2.2 operator()	1134
5.111 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __- sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	1134
5.111.1 Detailed Description	1135
5.112 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Tem- plate Reference	1135
5.112.1 Detailed Description	1135
5.113 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __- sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference	1136
5.113.1 Detailed Description	1136
5.114 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Tem- plate Reference	1136
5.114.1 Detailed Description	1137

5.115 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __-</code> <code>sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _</code> <code>Compare > Struct Template Reference</code>	1137
5.115.1 Detailed Description	1138
5.116 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch<</code> <code>false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare</code> <code>> Struct Template Reference</code>	1138
5.116.1 Detailed Description	1138
5.117 <code>__gnu_parallel::__replace_if_selector< _It, _Op, _Tp > Struct Tem-</code> <code>plate Reference</code>	1139
5.117.1 Detailed Description	1139
5.117.2 Constructor & Destructor Documentation	1140
5.117.2.1 <code>__replace_if_selector</code>	1140
5.117.3 Member Function Documentation	1140
5.117.3.1 <code>operator()</code>	1140
5.117.4 Member Data Documentation	1140
5.117.4.1 <code>__new_val</code>	1140
5.117.4.2 <code>_M_finish_iterator</code>	1141
5.118 <code>__gnu_parallel::__replace_selector< _It, _Tp > Struct Template Ref-</code> <code>erence</code>	1141
5.118.1 Detailed Description	1142
5.118.2 Constructor & Destructor Documentation	1142
5.118.2.1 <code>__replace_selector</code>	1142
5.118.3 Member Function Documentation	1142
5.118.3.1 <code>operator()</code>	1142
5.118.4 Member Data Documentation	1142
5.118.4.1 <code>__new_val</code>	1142
5.118.4.2 <code>_M_finish_iterator</code>	1143
5.119 <code>__gnu_parallel::__transform1_selector< _It > Struct Template Refer-</code> <code>ence</code>	1143
5.119.1 Detailed Description	1144
5.119.2 Member Function Documentation	1144
5.119.2.1 <code>operator()</code>	1144

5.119.3 Member Data Documentation	1144
5.119.3.1 _M_finish_iterator	1144
5.120 __gnu_parallel::__transform2_selector< _It > Struct Template Reference	1145
5.120.1 Detailed Description	1145
5.120.2 Member Function Documentation	1146
5.120.2.1 operator()	1146
5.120.3 Member Data Documentation	1146
5.120.3.1 _M_finish_iterator	1146
5.121 __gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference	1146
5.121.1 Detailed Description	1147
5.121.2 Member Typedef Documentation	1148
5.121.2.1 argument_type	1148
5.121.2.2 result_type	1148
5.122 __gnu_parallel::__DRandomShufflingGlobalData< _RAIter > Struct Template Reference	1148
5.122.1 Detailed Description	1149
5.122.2 Constructor & Destructor Documentation	1149
5.122.2.1 _DRandomShufflingGlobalData	1149
5.122.3 Member Data Documentation	1149
5.122.3.1 _M_bin_proc	1149
5.122.3.2 _M_dist	1150
5.122.3.3 _M_num_bins	1150
5.122.3.4 _M_num_bits	1150
5.122.3.5 _M_source	1150
5.122.3.6 _M_starts	1151
5.122.3.7 _M_temporaries	1151
5.123 __gnu_parallel::__DRSSorterPU< _RAIter, RandomNumberGenerator > Struct Template Reference	1151
5.123.1 Detailed Description	1152
5.123.2 Member Data Documentation	1152

5.123.2.1	__bins_end	1152
5.123.2.2	_M_bins_begin	1152
5.123.2.3	_M_num_threads	1152
5.123.2.4	_M_sd	1153
5.123.2.5	_M_seed	1153
5.124	__gnu_parallel::_DummyReduct Struct Reference	1153
5.124.1	Detailed Description	1153
5.125	__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference	1154
5.125.1	Detailed Description	1155
5.125.2	Member Typedef Documentation	1155
5.125.2.1	first_argument_type	1155
5.125.2.2	result_type	1155
5.125.2.3	second_argument_type	1155
5.126	__gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference	1155
5.126.1	Detailed Description	1156
5.126.2	Member Typedef Documentation	1157
5.126.2.1	first_argument_type	1157
5.126.2.2	result_type	1157
5.126.2.3	second_argument_type	1157
5.127	__gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference	1157
5.127.1	Detailed Description	1158
5.127.2	Constructor & Destructor Documentation	1158
5.127.2.1	_GuardedIterator	1158
5.127.3	Member Function Documentation	1158
5.127.3.1	operator _RAIter	1158
5.127.3.2	operator*	1159
5.127.3.3	operator++	1159
5.127.4	Friends And Related Function Documentation	1159
5.127.4.1	operator<	1159
5.127.4.2	operator<=	1160

5.128 <code>__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory ></code> Class Template Reference	1160
5.128.1 Detailed Description	1162
5.128.2 Member Typedef Documentation	1162
5.128.2.1 <code>first_type</code>	1162
5.128.2.2 <code>second_type</code>	1162
5.128.3 Member Data Documentation	1162
5.128.3.1 <code>first</code>	1162
5.128.3.2 <code>second</code>	1163
5.129 <code>__gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory ></code> Class Template Reference	1163
5.129.1 Detailed Description	1164
5.130 <code>__gnu_parallel::_Job< _DifferenceTp ></code> Struct Template Reference	1164
5.130.1 Detailed Description	1164
5.130.2 Member Data Documentation	1165
5.130.2.1 <code>_M_first</code>	1165
5.130.2.2 <code>_M_last</code>	1165
5.130.2.3 <code>_M_load</code>	1165
5.131 <code>__gnu_parallel::_Less< _T1, _T2 ></code> Struct Template Reference	1165
5.131.1 Detailed Description	1166
5.131.2 Member Typedef Documentation	1167
5.131.2.1 <code>first_argument_type</code>	1167
5.131.2.2 <code>result_type</code>	1167
5.131.2.3 <code>second_argument_type</code>	1167
5.132 <code>__gnu_parallel::_Lexicographic< _T1, _T2, _Compare ></code> Class Template Reference	1167
5.132.1 Detailed Description	1168
5.132.2 Member Typedef Documentation	1169
5.132.2.1 <code>first_argument_type</code>	1169
5.132.2.2 <code>result_type</code>	1169
5.132.2.3 <code>second_argument_type</code>	1169

5.133 <code>__gnu_parallel::LexicographicReverse< _T1, _T2, _Compare ></code> Class Template Reference	1169
5.133.1 Detailed Description	1170
5.133.2 Member Typedef Documentation	1171
5.133.2.1 <code>first_argument_type</code>	1171
5.133.2.2 <code>result_type</code>	1171
5.133.2.3 <code>second_argument_type</code>	1171
5.134 <code>__gnu_parallel::LoserTree< __stable, _Tp, _Compare ></code> Class Tem- plate Reference	1171
5.134.1 Detailed Description	1172
5.134.2 Member Function Documentation	1173
5.134.2.1 <code>__delete_min_insert</code>	1173
5.134.2.2 <code>__get_min_source</code>	1173
5.134.2.3 <code>__insert_start</code>	1173
5.134.3 Member Data Documentation	1174
5.134.3.1 <code>_M_comp</code>	1174
5.134.3.2 <code>_M_first_insert</code>	1174
5.134.3.3 <code>_M_log_k</code>	1174
5.134.3.4 <code>_M_losers</code>	1174
5.135 <code>__gnu_parallel::LoserTree< false, _Tp, _Compare ></code> Class Template Reference	1175
5.135.1 Detailed Description	1176
5.135.2 Member Function Documentation	1176
5.135.2.1 <code>__delete_min_insert</code>	1176
5.135.2.2 <code>__get_min_source</code>	1176
5.135.2.3 <code>__init_winner</code>	1177
5.135.2.4 <code>__insert_start</code>	1177
5.135.3 Member Data Documentation	1178
5.135.3.1 <code>_M_comp</code>	1178
5.135.3.2 <code>_M_first_insert</code>	1178
5.135.3.3 <code>_M_log_k</code>	1178
5.135.3.4 <code>_M_losers</code>	1178

5.136__gnu_parallel::_LoserTreeBase< _Tp, _Compare > Class Template Reference	1179
5.136.1 Detailed Description	1180
5.136.2 Constructor & Destructor Documentation	1180
5.136.2.1 _LoserTreeBase	1180
5.136.2.2 ~_LoserTreeBase	1180
5.136.3 Member Function Documentation	1181
5.136.3.1 __get_min_source	1181
5.136.3.2 __insert_start	1181
5.136.4 Member Data Documentation	1181
5.136.4.1 _M_comp	1181
5.136.4.2 _M_first_insert	1182
5.136.4.3 _M_log_k	1182
5.136.4.4 _M_losers	1182
5.137__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser Struct Reference	1182
5.137.1 Detailed Description	1183
5.137.2 Member Data Documentation	1183
5.137.2.1 _M_key	1183
5.137.2.2 _M_source	1183
5.137.2.3 _M_sup	1183
5.138__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare > Class Template Reference	1184
5.138.1 Detailed Description	1185
5.139__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare > Class Template Reference	1185
5.139.1 Detailed Description	1186
5.140__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference	1186
5.140.1 Detailed Description	1187
5.141__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference	1187
5.141.1 Detailed Description	1188

5.142__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _- Compare > Class Template Reference	1188
5.142.1 Detailed Description	1189
5.143__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference	1189
5.143.1 Detailed Description	1190
5.144__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference	1191
5.144.1 Detailed Description	1192
5.145__gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference	1192
5.145.1 Detailed Description	1192
5.145.2 Member Data Documentation	1193
5.145.2.1 _M_use_pointer	1193
5.146__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference	1193
5.146.1 Detailed Description	1194
5.147__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference	1195
5.147.1 Detailed Description	1196
5.148__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference	1196
5.148.1 Detailed Description	1197
5.149__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference	1197
5.149.1 Detailed Description	1198
5.149.2 Member Typedef Documentation	1199
5.149.2.1 first_argument_type	1199
5.149.2.2 result_type	1199
5.149.2.3 second_argument_type	1199
5.150__gnu_parallel::_Nothing Struct Reference	1199
5.150.1 Detailed Description	1199
5.150.2 Member Function Documentation	1200
5.150.2.1 operator()	1200

5.151 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference	1200
5.151.1 Detailed Description	1200
5.151.2 Member Data Documentation	1201
5.151.2.1 _M_begin	1201
5.151.2.2 _M_end	1201
5.152 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference	1201
5.152.1 Detailed Description	1202
5.152.2 Member Typedef Documentation	1203
5.152.2.1 first_argument_type	1203
5.152.2.2 result_type	1203
5.152.2.3 second_argument_type	1203
5.153 __gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference	1203
5.153.1 Detailed Description	1204
5.153.2 Member Data Documentation	1204
5.153.2.1 _M_num_threads	1204
5.153.2.2 _M_offsets	1204
5.153.2.3 _M_pieces	1204
5.153.2.4 _M_samples	1205
5.153.2.5 _M_source	1205
5.153.2.6 _M_starts	1205
5.153.2.7 _M_temporary	1205
5.154 __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference	1206
5.154.1 Detailed Description	1206
5.154.2 Constructor & Destructor Documentation	1206
5.154.2.1 _PseudoSequence	1206
5.154.3 Member Function Documentation	1207
5.154.3.1 begin	1207
5.154.3.2 end	1207

5.155__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference	1207
5.155.1 Detailed Description	1208
5.156__gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Ref- erence	1208
5.156.1 Detailed Description	1209
5.156.2 Member Typedef Documentation	1209
5.156.2.1 _Piece	1209
5.156.3 Constructor & Destructor Documentation	1209
5.156.3.1 _QSBThreadLocal	1209
5.156.4 Member Data Documentation	1210
5.156.4.1 _M_elements_leftover	1210
5.156.4.2 _M_global	1210
5.156.4.3 _M_initial	1210
5.156.4.4 _M_leftover_parts	1210
5.156.4.5 _M_num_threads	1210
5.157__gnu_parallel::_RandomNumber Class Reference	1211
5.157.1 Detailed Description	1211
5.157.2 Constructor & Destructor Documentation	1211
5.157.2.1 _RandomNumber	1211
5.157.2.2 _RandomNumber	1211
5.157.3 Member Function Documentation	1212
5.157.3.1 __genrand_bits	1212
5.157.3.2 operator()	1212
5.157.3.3 operator()	1212
5.158__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference	1212
5.158.1 Detailed Description	1213
5.158.2 Constructor & Destructor Documentation	1214
5.158.2.1 _RestrictedBoundedConcurrentQueue	1214
5.158.2.2 ~_RestrictedBoundedConcurrentQueue	1214
5.158.3 Member Function Documentation	1214

5.158.3.1 pop_back	1214
5.158.3.2 pop_front	1214
5.158.3.3 push_front	1215
5.159 __gnu_parallel::_SamplingSorter< __stable, __RAIter, __-	
StrictWeakOrdering > Struct Template Reference	1215
5.159.1 Detailed Description	1215
5.160 __gnu_parallel::_SamplingSorter< false, __RAIter, __-	
StrictWeakOrdering > Struct Template Reference	1216
5.160.1 Detailed Description	1216
5.161 __gnu_parallel::_Settings Struct Reference	1216
5.161.1 Detailed Description	1218
5.161.2 Member Function Documentation	1218
5.161.2.1 get	1218
5.161.2.2 set	1218
5.161.3 Member Data Documentation	1218
5.161.3.1 accumulate_minimal_n	1218
5.161.3.2 adjacent_difference_minimal_n	1219
5.161.3.3 cache_line_size	1219
5.161.3.4 count_minimal_n	1219
5.161.3.5 fill_minimal_n	1219
5.161.3.6 find_increasing_factor	1219
5.161.3.7 find_initial_block_size	1219
5.161.3.8 find_maximum_block_size	1219
5.161.3.9 find_scale_factor	1220
5.161.3.10 find_sequential_search_size	1220
5.161.3.11 for_each_minimal_n	1220
5.161.3.12 generate_minimal_n	1220
5.161.3.13 L1_cache_size	1220
5.161.3.14 L2_cache_size	1220
5.161.3.15 max_element_minimal_n	1221
5.161.3.16 merge_minimal_n	1221
5.161.3.17 merge_oversampling	1221

5.161.3.18	min_element_minimal_n	1221
5.161.3.19	multiway_merge_minimal_k	1221
5.161.3.20	multiway_merge_minimal_n	1221
5.161.3.21	multiway_merge_oversampling	1222
5.161.3.22	nth_element_minimal_n	1222
5.161.3.23	partial_sort_minimal_n	1222
5.161.3.24	partial_sum_dilation	1222
5.161.3.25	partial_sum_minimal_n	1222
5.161.3.26	partition_chunk_share	1222
5.161.3.27	partition_chunk_size	1223
5.161.3.28	partition_minimal_n	1223
5.161.3.29	qsb_steals	1223
5.161.3.30	random_shuffle_minimal_n	1223
5.161.3.31	replace_minimal_n	1223
5.161.3.32	search_minimal_n	1223
5.161.3.33	set_difference_minimal_n	1223
5.161.3.34	set_intersection_minimal_n	1224
5.161.3.35	set_symmetric_difference_minimal_n	1224
5.161.3.36	set_union_minimal_n	1224
5.161.3.37	sort_minimal_n	1224
5.161.3.38	sort_mwms_oversampling	1224
5.161.3.39	sort_qs_num_samples_preset	1224
5.161.3.40	sort_qsb_base_case_maximal_n	1225
5.161.3.41	TLB_size	1225
5.161.3.42	transform_minimal_n	1225
5.161.3.43	unique_copy_minimal_n	1225
5.162	__gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _- SortingPlacesIterator > Struct Template Reference	1225
5.162.1	Detailed Description	1226
5.163	__gnu_parallel::SplitConsistently< false, _RAIter, _Compare, _- SortingPlacesIterator > Struct Template Reference	1226
5.163.1	Detailed Description	1226

5.164__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _-	
SortingPlacesIterator > Struct Template Reference	1227
5.164.1 Detailed Description	1227
5.165__gnu_parallel::balanced_quicksort_tag Struct Reference	1227
5.165.1 Detailed Description	1228
5.165.2 Member Function Documentation	1228
5.165.2.1 __get_num_threads	1228
5.165.2.2 set_num_threads	1229
5.166__gnu_parallel::balanced_tag Struct Reference	1229
5.166.1 Detailed Description	1229
5.166.2 Member Function Documentation	1230
5.166.2.1 __get_num_threads	1230
5.166.2.2 set_num_threads	1230
5.167__gnu_parallel::constant_size_blocks_tag Struct Reference	1230
5.167.1 Detailed Description	1231
5.168__gnu_parallel::default_parallel_tag Struct Reference	1231
5.168.1 Detailed Description	1232
5.168.2 Member Function Documentation	1232
5.168.2.1 __get_num_threads	1232
5.168.2.2 set_num_threads	1233
5.169__gnu_parallel::equal_split_tag Struct Reference	1233
5.169.1 Detailed Description	1233
5.170__gnu_parallel::exact_tag Struct Reference	1234
5.170.1 Detailed Description	1234
5.170.2 Member Function Documentation	1235
5.170.2.1 __get_num_threads	1235
5.170.2.2 set_num_threads	1235
5.171__gnu_parallel::find_tag Struct Reference	1235
5.171.1 Detailed Description	1236
5.172__gnu_parallel::growing_blocks_tag Struct Reference	1236
5.172.1 Detailed Description	1237

5.173 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference . . .	1237
5.173.1 Detailed Description	1238
5.173.2 Member Function Documentation	1238
5.173.2.1 __get_num_threads	1238
5.173.2.2 set_num_threads	1239
5.174 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference .	1239
5.174.1 Detailed Description	1240
5.174.2 Member Function Documentation	1240
5.174.2.1 __get_num_threads	1240
5.174.2.2 set_num_threads	1240
5.175 __gnu_parallel::multiway_mergesort_tag Struct Reference	1240
5.175.1 Detailed Description	1241
5.175.2 Member Function Documentation	1241
5.175.2.1 __get_num_threads	1241
5.175.2.2 set_num_threads	1242
5.176 __gnu_parallel::omp_loop_static_tag Struct Reference	1242
5.176.1 Detailed Description	1243
5.176.2 Member Function Documentation	1243
5.176.2.1 __get_num_threads	1243
5.176.2.2 set_num_threads	1243
5.177 __gnu_parallel::omp_loop_tag Struct Reference	1243
5.177.1 Detailed Description	1244
5.177.2 Member Function Documentation	1244
5.177.2.1 __get_num_threads	1244
5.177.2.2 set_num_threads	1245
5.178 __gnu_parallel::parallel_tag Struct Reference	1245
5.178.1 Detailed Description	1247
5.178.2 Constructor & Destructor Documentation	1247
5.178.2.1 parallel_tag	1247
5.178.2.2 parallel_tag	1247
5.178.3 Member Function Documentation	1247

5.178.3.1	<code>__get_num_threads</code>	1247
5.178.3.2	<code>set_num_threads</code>	1247
5.179	<code>gnu_parallel::quicksort_tag</code> Struct Reference	1248
5.179.1	Detailed Description	1248
5.179.2	Member Function Documentation	1249
5.179.2.1	<code>__get_num_threads</code>	1249
5.179.2.2	<code>set_num_threads</code>	1249
5.180	<code>gnu_parallel::sampling_tag</code> Struct Reference	1249
5.180.1	Detailed Description	1250
5.180.2	Member Function Documentation	1250
5.180.2.1	<code>__get_num_threads</code>	1250
5.180.2.2	<code>set_num_threads</code>	1251
5.181	<code>gnu_parallel::sequential_tag</code> Struct Reference	1251
5.181.1	Detailed Description	1251
5.182	<code>gnu_parallel::unbalanced_tag</code> Struct Reference	1251
5.182.1	Detailed Description	1252
5.182.2	Member Function Documentation	1252
5.182.2.1	<code>__get_num_threads</code>	1252
5.182.2.2	<code>set_num_threads</code>	1253
5.183	<code>gnu_pbds::associative_container_tag</code> Struct Reference	1253
5.183.1	Detailed Description	1253
5.184	<code>gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator ></code> Class Template Reference	1254
5.184.1	Detailed Description	1255
5.185	<code>gnu_pbds::basic_hash_tag</code> Struct Reference	1255
5.185.1	Detailed Description	1256
5.186	<code>gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_TL, Allocator ></code> Class Template Reference	1256
5.186.1	Detailed Description	1257
5.187	<code>gnu_pbds::basic_tree_tag</code> Struct Reference	1257
5.187.1	Detailed Description	1258

5.188__gnu_pbds::binary_heap_tag Struct Reference	1258
5.188.1 Detailed Description	1258
5.189__gnu_pbds::binomial_heap_tag Struct Reference	1259
5.189.1 Detailed Description	1259
5.190__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference	1260
5.190.1 Detailed Description	1261
5.191__gnu_pbds::cc_hash_tag Struct Reference	1262
5.191.1 Detailed Description	1262
5.192__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Alloca- tor > Class Template Reference	1263
5.192.1 Detailed Description	1264
5.193__gnu_pbds::container_tag Struct Reference	1264
5.193.1 Detailed Description	1264
5.194__gnu_pbds::container_traits< Cntnr > Struct Template Reference . .	1265
5.194.1 Detailed Description	1265
5.195__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false > Struct Template Reference	1265
5.195.1 Detailed Description	1266
5.196__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true > Struct Template Reference	1266
5.196.1 Detailed Description	1267
5.197__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allo- cator, false > Struct Template Reference	1267
5.197.1 Detailed Description	1268
5.198__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allo- cator, true > Struct Template Reference	1268
5.198.1 Detailed Description	1269
5.199__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class Template Reference	1269
5.199.1 Detailed Description	1271
5.200__gnu_pbds::gp_hash_tag Struct Reference	1271

5.200.1 Detailed Description	1272
5.201 __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Al- locator > Class Template Reference	1272
5.201.1 Detailed Description	1274
5.202 __gnu_pbds::list_update_tag Struct Reference	1274
5.202.1 Detailed Description	1275
5.203 __gnu_pbds::null_mapped_type Struct Reference	1275
5.203.1 Detailed Description	1275
5.204 __gnu_pbds::ov_tree_tag Struct Reference	1276
5.204.1 Detailed Description	1276
5.205 __gnu_pbds::pairing_heap_tag Struct Reference	1277
5.205.1 Detailed Description	1277
5.206 __gnu_pbds::pat_trie_tag Struct Reference	1277
5.206.1 Detailed Description	1278
5.207 __gnu_pbds::priority_queue_tag Struct Reference	1279
5.207.1 Detailed Description	1279
5.208 __gnu_pbds::rb_tree_tag Struct Reference	1279
5.208.1 Detailed Description	1280
5.209 __gnu_pbds::rc_binomial_heap_tag Struct Reference	1281
5.209.1 Detailed Description	1281
5.210 __gnu_pbds::sequence_tag Struct Reference	1281
5.210.1 Detailed Description	1282
5.211 __gnu_pbds::splay_tree_tag Struct Reference	1282
5.211.1 Detailed Description	1283
5.212 __gnu_pbds::string_tag Struct Reference	1283
5.212.1 Detailed Description	1284
5.213 __gnu_pbds::thin_heap_tag Struct Reference	1284
5.213.1 Detailed Description	1285
5.214 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allo- cator > Class Template Reference	1285
5.214.1 Detailed Description	1287
5.215 __gnu_pbds::tree_tag Struct Reference	1287

5.215.1 Detailed Description	1288
5.216__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_-, Update, Allocator > Class Template Reference	1288
5.216.1 Detailed Description	1289
5.217__gnu_pbds::trie_tag Struct Reference	1290
5.217.1 Detailed Description	1290
5.218__gnu_profile::__container_size_info Class Reference	1291
5.218.1 Detailed Description	1292
5.219__gnu_profile::__container_size_stack_info Class Reference	1292
5.219.1 Detailed Description	1293
5.220__gnu_profile::__hashfunc_info Class Reference	1293
5.220.1 Detailed Description	1294
5.221__gnu_profile::__hashfunc_stack_info Class Reference	1295
5.221.1 Detailed Description	1296
5.222__gnu_profile::__list2vector_info Class Reference	1296
5.222.1 Detailed Description	1297
5.223__gnu_profile::__map2umap_info Class Reference	1297
5.223.1 Detailed Description	1299
5.224__gnu_profile::__map2umap_stack_info Class Reference	1299
5.224.1 Detailed Description	1300
5.225__gnu_profile::__object_info_base Class Reference	1300
5.225.1 Detailed Description	1301
5.226__gnu_profile::__reentrance_guard Struct Reference	1301
5.226.1 Detailed Description	1301
5.227__gnu_profile::__stack_hash Class Reference	1302
5.227.1 Detailed Description	1302
5.228__gnu_profile::__stack_info_base< __object_info > Class Template Reference	1302
5.228.1 Detailed Description	1302
5.229__gnu_profile::__trace_base< __object_info, __stack_info > Class Template Reference	1303
5.229.1 Detailed Description	1303

5.230	__gnu_profile::__trace_container_size Class Reference	1304
5.230.1	Detailed Description	1304
5.231	__gnu_profile::__trace_hash_func Class Reference	1304
5.231.1	Detailed Description	1305
5.232	__gnu_profile::__trace_hashtable_size Class Reference	1306
5.232.1	Detailed Description	1306
5.233	__gnu_profile::__trace_map2umap Class Reference	1306
5.233.1	Detailed Description	1307
5.234	__gnu_profile::__trace_vector_size Class Reference	1308
5.234.1	Detailed Description	1308
5.235	__gnu_profile::__trace_vector_to_list Class Reference	1308
5.235.1	Detailed Description	1310
5.236	__gnu_profile::__vector2list_info Class Reference	1310
5.236.1	Detailed Description	1311
5.237	__gnu_profile::__vector2list_stack_info Class Reference	1311
5.237.1	Detailed Description	1313
5.238	__gnu_profile::__warning_data Struct Reference	1313
5.238.1	Detailed Description	1313
5.239	std::__basic_future< _Res > Class Template Reference	1314
5.239.1	Detailed Description	1315
5.239.2	Member Function Documentation	1315
5.239.2.1	_M_get_result	1315
5.240	std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference	1315
5.240.1	Detailed Description	1317
5.240.2	Member Function Documentation	1317
5.240.2.1	do_out	1317
5.240.2.2	in	1318
5.240.2.3	out	1319
5.240.2.4	unshift	1320
5.241	std::__ctype_abstract_base< _CharT > Class Template Reference	1321

5.241.1 Detailed Description	1323
5.241.2 Member Typedef Documentation	1323
5.241.2.1 char_type	1323
5.241.3 Member Function Documentation	1324
5.241.3.1 do_is	1324
5.241.3.2 do_is	1324
5.241.3.3 do_narrow	1325
5.241.3.4 do_narrow	1325
5.241.3.5 do_scan_is	1326
5.241.3.6 do_scan_not	1326
5.241.3.7 do_tolower	1327
5.241.3.8 do_tolower	1327
5.241.3.9 do_toupper	1328
5.241.3.10do_toupper	1328
5.241.3.11do_widen	1329
5.241.3.12do_widen	1329
5.241.3.13is	1330
5.241.3.14is	1330
5.241.3.15narrow	1331
5.241.3.16narrow	1331
5.241.3.17scan_is	1332
5.241.3.18scan_not	1332
5.241.3.19olower	1333
5.241.3.20olower	1333
5.241.3.21toupper	1334
5.241.3.22toupper	1334
5.241.3.23widen	1334
5.241.3.24widen	1335
5.242std::__debug::bitset<_Nb> Class Template Reference	1336
5.242.1 Detailed Description	1338
5.242.2 Member Function Documentation	1338

5.242.2.1	_M_detach_all	1338
5.242.2.2	_M_detach_singular	1338
5.242.2.3	_M_get_mutex	1338
5.242.2.4	_M_invalidate_all	1339
5.242.2.5	_M_revalidate_singular	1339
5.242.2.6	_M_swap	1339
5.242.3	Member Data Documentation	1339
5.242.3.1	_M_const_iterators	1339
5.242.3.2	_M_iterators	1339
5.242.3.3	_M_version	1340
5.243	std::__debug::deque< _Tp, _Allocator > Class Template Reference	1340
5.243.1	Detailed Description	1343
5.243.2	Member Function Documentation	1343
5.243.2.1	_M_detach_all	1343
5.243.2.2	_M_detach_singular	1343
5.243.2.3	_M_get_mutex	1344
5.243.2.4	_M_invalidate_all	1344
5.243.2.5	_M_invalidate_if	1344
5.243.2.6	_M_revalidate_singular	1344
5.243.2.7	_M_swap	1344
5.243.2.8	_M_transfer_iter	1344
5.243.3	Member Data Documentation	1345
5.243.3.1	_M_const_iterators	1345
5.243.3.2	_M_iterators	1345
5.243.3.3	_M_version	1345
5.244	std::__debug::list< _Tp, _Allocator > Class Template Reference	1345
5.244.1	Detailed Description	1349
5.244.2	Member Function Documentation	1349
5.244.2.1	_M_detach_all	1349
5.244.2.2	_M_detach_singular	1349
5.244.2.3	_M_get_mutex	1349

5.244.2.4	<code>_M_invalidate_all</code>	1350
5.244.2.5	<code>_M_invalidate_if</code>	1350
5.244.2.6	<code>_M_revalidate_singular</code>	1350
5.244.2.7	<code>_M_swap</code>	1350
5.244.2.8	<code>_M_transfer_iter</code>	1350
5.244.3	Member Data Documentation	1350
5.244.3.1	<code>_M_const_iterators</code>	1350
5.244.3.2	<code>_M_iterators</code>	1351
5.244.3.3	<code>_M_version</code>	1351
5.245	<code>std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference</code>	1351
5.245.1	Detailed Description	1354
5.245.2	Member Function Documentation	1354
5.245.2.1	<code>_M_detach_all</code>	1354
5.245.2.2	<code>_M_detach_singular</code>	1355
5.245.2.3	<code>_M_get_mutex</code>	1355
5.245.2.4	<code>_M_invalidate_all</code>	1355
5.245.2.5	<code>_M_invalidate_if</code>	1355
5.245.2.6	<code>_M_revalidate_singular</code>	1355
5.245.2.7	<code>_M_swap</code>	1356
5.245.2.8	<code>_M_transfer_iter</code>	1356
5.245.3	Member Data Documentation	1356
5.245.3.1	<code>_M_const_iterators</code>	1356
5.245.3.2	<code>_M_iterators</code>	1356
5.245.3.3	<code>_M_version</code>	1356
5.246	<code>std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference</code>	1357
5.246.1	Detailed Description	1359
5.246.2	Member Function Documentation	1360
5.246.2.1	<code>_M_detach_all</code>	1360
5.246.2.2	<code>_M_detach_singular</code>	1360
5.246.2.3	<code>_M_get_mutex</code>	1360

5.246.2.4	_M_invalidate_all	1360
5.246.2.5	_M_invalidate_if	1360
5.246.2.6	_M_revalidate_singular	1361
5.246.2.7	_M_swap	1361
5.246.2.8	_M_transfer_iter	1361
5.246.3	Member Data Documentation	1361
5.246.3.1	_M_const_iterators	1361
5.246.3.2	_M_iterators	1361
5.246.3.3	_M_version	1362
5.247	std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference	1362
5.247.1	Detailed Description	1365
5.247.2	Member Function Documentation	1365
5.247.2.1	_M_detach_all	1365
5.247.2.2	_M_detach_singular	1365
5.247.2.3	_M_get_mutex	1365
5.247.2.4	_M_invalidate_all	1365
5.247.2.5	_M_invalidate_if	1366
5.247.2.6	_M_revalidate_singular	1366
5.247.2.7	_M_swap	1366
5.247.2.8	_M_transfer_iter	1366
5.247.3	Member Data Documentation	1366
5.247.3.1	_M_const_iterators	1366
5.247.3.2	_M_iterators	1367
5.247.3.3	_M_version	1367
5.248	std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference	1367
5.248.1	Detailed Description	1370
5.248.2	Member Function Documentation	1370
5.248.2.1	_M_detach_all	1370
5.248.2.2	_M_detach_singular	1371
5.248.2.3	_M_get_mutex	1371

5.248.2.4	<code>_M_invalidate_all</code>	1371
5.248.2.5	<code>_M_invalidate_if</code>	1371
5.248.2.6	<code>_M_revalidate_singular</code>	1371
5.248.2.7	<code>_M_swap</code>	1372
5.248.2.8	<code>_M_transfer_iter</code>	1372
5.248.3	Member Data Documentation	1372
5.248.3.1	<code>_M_const_iterators</code>	1372
5.248.3.2	<code>_M_iterators</code>	1372
5.248.3.3	<code>_M_version</code>	1372
5.249	<code>std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc></code>	
	Class Template Reference	1373
5.249.1	Detailed Description	1375
5.249.2	Member Function Documentation	1375
5.249.2.1	<code>_M_detach_all</code>	1375
5.249.2.2	<code>_M_detach_singular</code>	1375
5.249.2.3	<code>_M_get_mutex</code>	1376
5.249.2.4	<code>_M_invalidate_all</code>	1376
5.249.2.5	<code>_M_invalidate_if</code>	1376
5.249.2.6	<code>_M_revalidate_singular</code>	1376
5.249.2.7	<code>_M_swap</code>	1376
5.249.2.8	<code>_M_transfer_iter</code>	1377
5.249.3	Member Data Documentation	1377
5.249.3.1	<code>_M_const_iterators</code>	1377
5.249.3.2	<code>_M_iterators</code>	1377
5.249.3.3	<code>_M_version</code>	1377
5.250	<code>std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc></code>	
	Class Template Reference	1378
5.250.1	Detailed Description	1380
5.250.2	Member Function Documentation	1380
5.250.2.1	<code>_M_detach_all</code>	1380
5.250.2.2	<code>_M_detach_singular</code>	1380
5.250.2.3	<code>_M_get_mutex</code>	1380

5.250.2.4	_M_invalidate_all	1380
5.250.2.5	_M_invalidate_if	1381
5.250.2.6	_M_revalidate_singular	1381
5.250.2.7	_M_swap	1381
5.250.2.8	_M_transfer_iter	1381
5.250.3	Member Data Documentation	1381
5.250.3.1	_M_const_iterators	1381
5.250.3.2	_M_iterators	1382
5.250.3.3	_M_version	1382
5.251	std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >	
	Class Template Reference	1382
5.251.1	Detailed Description	1384
5.251.2	Member Function Documentation	1384
5.251.2.1	_M_detach_all	1384
5.251.2.2	_M_detach_singular	1384
5.251.2.3	_M_get_mutex	1385
5.251.2.4	_M_invalidate_all	1385
5.251.2.5	_M_invalidate_if	1385
5.251.2.6	_M_revalidate_singular	1385
5.251.2.7	_M_swap	1385
5.251.2.8	_M_transfer_iter	1386
5.251.3	Member Data Documentation	1386
5.251.3.1	_M_const_iterators	1386
5.251.3.2	_M_iterators	1386
5.251.3.3	_M_version	1386
5.252	std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class	
	Template Reference	1387
5.252.1	Detailed Description	1389
5.252.2	Member Function Documentation	1389
5.252.2.1	_M_detach_all	1389
5.252.2.2	_M_detach_singular	1389
5.252.2.3	_M_get_mutex	1390

5.252.2.4	_M_invalidate_all	1390
5.252.2.5	_M_invalidate_if	1390
5.252.2.6	_M_revalidate_singular	1390
5.252.2.7	_M_swap	1390
5.252.2.8	_M_transfer_iter	1391
5.252.3	Member Data Documentation	1391
5.252.3.1	_M_const_iterators	1391
5.252.3.2	_M_iterators	1391
5.252.3.3	_M_version	1391
5.253	std::__debug::vector< _Tp, _Allocator > Class Template Reference	1392
5.253.1	Detailed Description	1395
5.253.2	Constructor & Destructor Documentation	1395
5.253.2.1	vector	1395
5.253.3	Member Function Documentation	1395
5.253.3.1	_M_detach_all	1395
5.253.3.2	_M_detach_singular	1395
5.253.3.3	_M_get_mutex	1395
5.253.3.4	_M_invalidate_all	1396
5.253.3.5	_M_invalidate_if	1396
5.253.3.6	_M_revalidate_singular	1396
5.253.3.7	_M_swap	1396
5.253.3.8	_M_transfer_iter	1396
5.253.4	Member Data Documentation	1396
5.253.4.1	_M_const_iterators	1396
5.253.4.2	_M_iterators	1397
5.253.4.3	_M_version	1397
5.254	std::__declval_protector< _Tp > Struct Template Reference	1397
5.254.1	Detailed Description	1398
5.255	std::__exception_ptr::exception_ptr Class Reference	1398
5.255.1	Detailed Description	1398
5.256	std::__future_base Struct Reference	1399

5.256.1 Detailed Description	1400
5.257std::__future_base::__Ptr< _Res > Struct Template Reference	1400
5.257.1 Detailed Description	1401
5.258std::__future_base::__Result< _Res > Struct Template Reference	1401
5.258.1 Detailed Description	1402
5.259std::__future_base::__Result< _Res & > Struct Template Reference	1402
5.259.1 Detailed Description	1403
5.260std::__future_base::__Result< void > Struct Template Reference	1403
5.260.1 Detailed Description	1404
5.261std::__future_base::__Result_alloc< _Res, _Alloc > Struct Template Reference	1404
5.261.1 Detailed Description	1405
5.262std::__future_base::__Result_base Struct Reference	1405
5.262.1 Detailed Description	1406
5.263std::__future_base::__State Class Reference	1406
5.263.1 Detailed Description	1407
5.264std::__is_location_invariant< _Tp > Struct Template Reference	1407
5.264.1 Detailed Description	1407
5.265std::__iterator_traits< _Iterator, bool > Struct Template Reference	1408
5.265.1 Detailed Description	1408
5.266std::__numeric_limits_base Struct Reference	1408
5.266.1 Detailed Description	1410
5.266.2 Member Data Documentation	1410
5.266.2.1 digits	1410
5.266.2.2 digits10	1410
5.266.2.3 has_denorm	1410
5.266.2.4 has_denorm_loss	1410
5.266.2.5 has_infinity	1411
5.266.2.6 has_quiet_NaN	1411
5.266.2.7 has_signaling_NaN	1411
5.266.2.8 is_bounded	1411

5.266.2.9 is_exact	1411
5.266.2.10 is_iec559	1411
5.266.2.11 is_integer	1411
5.266.2.12 is_modulo	1412
5.266.2.13 is_signed	1412
5.266.2.14 is_specialized	1412
5.266.2.15 max_digits10	1412
5.266.2.16 max_exponent	1412
5.266.2.17 max_exponent10	1412
5.266.2.18 min_exponent	1413
5.266.2.19 min_exponent10	1413
5.266.2.20 radix	1413
5.266.2.21 round_style	1413
5.266.2.22 inyness_before	1413
5.266.2.23 traps	1413
5.267 std::__parallel::__CRandNumber< _MustBeInt > Struct Template Reference	1414
5.267.1 Detailed Description	1414
5.268 std::__profile::bitset< _Nb > Class Template Reference	1414
5.268.1 Detailed Description	1416
5.269 std::__profile::deque< _Tp, _Allocator > Class Template Reference	1416
5.269.1 Detailed Description	1418
5.270 std::__profile::list< _Tp, _Allocator > Class Template Reference	1418
5.270.1 Detailed Description	1421
5.271 std::__profile::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1421
5.271.1 Detailed Description	1423
5.272 std::__profile::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1423
5.272.1 Detailed Description	1425
5.273 std::__profile::multiset< _Key, _Compare, _Allocator > Class Template Reference	1425

5.273.1 Detailed Description	1427
5.274std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference	1427
5.274.1 Detailed Description	1429
5.275std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1429
5.275.1 Detailed Description	1431
5.276std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _- Alloc > Class Template Reference	1431
5.276.1 Detailed Description	1432
5.277std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1432
5.277.1 Detailed Description	1434
5.278std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference	1434
5.278.1 Detailed Description	1435
5.279std::_Base_bitset< _Nw > Struct Template Reference	1435
5.279.1 Detailed Description	1437
5.279.2 Member Data Documentation	1437
5.279.2.1 _M_w	1437
5.280std::_Base_bitset< 0 > Struct Template Reference	1437
5.280.1 Detailed Description	1438
5.281std::_Base_bitset< 1 > Struct Template Reference	1438
5.281.1 Detailed Description	1439
5.282std::_Build_index_tuple< _Num > Struct Template Reference	1440
5.282.1 Detailed Description	1440
5.283std::_Deque_base< _Tp, _Alloc > Class Template Reference	1440
5.283.1 Detailed Description	1442
5.283.2 Member Function Documentation	1442
5.283.2.1 _M_initialize_map	1442
5.284std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference .	1442
5.284.1 Detailed Description	1444
5.284.2 Member Function Documentation	1444

5.284.2.1 <code>_M_set_node</code>	1444
5.285 <code>std::_Derives_from_binary_function< _Tp ></code> Struct Template Reference	1445
5.285.1 Detailed Description	1445
5.286 <code>std::_Derives_from_unary_function< _Tp ></code> Struct Template Reference	1445
5.286.1 Detailed Description	1445
5.287 <code>std::_Function_base</code> Class Reference	1446
5.287.1 Detailed Description	1447
5.288 <code>std::_Function_to_function_pointer< _Tp, _IsFunctionType ></code> Struct Template Reference	1447
5.288.1 Detailed Description	1447
5.289 <code>std::_Fwd_list_base< _Tp, _Alloc ></code> Struct Template Reference . . .	1447
5.289.1 Detailed Description	1449
5.290 <code>std::_Fwd_list_const_iterator< _Tp ></code> Struct Template Reference . .	1449
5.290.1 Detailed Description	1450
5.291 <code>std::_Fwd_list_iterator< _Tp ></code> Struct Template Reference	1450
5.291.1 Detailed Description	1451
5.292 <code>std::_Fwd_list_node< _Tp ></code> Struct Template Reference	1451
5.292.1 Detailed Description	1452
5.293 <code>std::_Fwd_list_node_base</code> Struct Reference	1453
5.293.1 Detailed Description	1454
5.294 <code>std::_Index_tuple< _Indexes ></code> Struct Template Reference	1454
5.294.1 Detailed Description	1454
5.295 <code>std::_List_base< _Tp, _Alloc ></code> Class Template Reference	1454
5.295.1 Detailed Description	1456
5.296 <code>std::_List_const_iterator< _Tp ></code> Struct Template Reference	1456
5.296.1 Detailed Description	1457
5.297 <code>std::_List_iterator< _Tp ></code> Struct Template Reference	1457
5.297.1 Detailed Description	1458
5.298 <code>std::_List_node< _Tp ></code> Struct Template Reference	1458
5.298.1 Detailed Description	1459
5.298.2 Member Data Documentation	1460

5.298.2.1 <code>_M_data</code>	1460
5.299 <code>std::_List_node_base</code> Struct Reference	1460
5.299.1 Detailed Description	1461
5.300 <code>std::_Maybe_get_result_type< _Has_result_type, _Functor ></code> Struct Template Reference	1461
5.300.1 Detailed Description	1461
5.301 <code>std::_Maybe_unary_or_binary_function< _Res, _ArgTypes ></code> Struct Template Reference	1462
5.301.1 Detailed Description	1462
5.302 <code>std::_Maybe_unary_or_binary_function< _Res, _T1 ></code> Struct Tem- plate Reference	1462
5.302.1 Detailed Description	1463
5.302.2 Member Typedef Documentation	1463
5.302.2.1 <code>argument_type</code>	1463
5.302.2.2 <code>result_type</code>	1464
5.303 <code>std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 ></code> Struct Template Reference	1464
5.303.1 Detailed Description	1465
5.303.2 Member Typedef Documentation	1465
5.303.2.1 <code>first_argument_type</code>	1465
5.303.2.2 <code>result_type</code>	1465
5.303.2.3 <code>second_argument_type</code>	1465
5.304 <code>std::_Maybe_wrap_member_pointer< _Tp ></code> Struct Template Reference	1466
5.304.1 Detailed Description	1466
5.305 <code>std::_Maybe_wrap_member_pointer< _Tp _Class::* ></code> Struct Tem- plate Reference	1466
5.305.1 Detailed Description	1467
5.306 <code>std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const ></code> Class Tem- plate Reference	1467
5.306.1 Detailed Description	1468
5.307 <code>std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile ></code> Class Template Reference	1468
5.307.1 Detailed Description	1469

5.308std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile > Class Template Reference	1469
5.308.1 Detailed Description	1470
5.309std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) > Class Template Reference	1470
5.309.1 Detailed Description	1471
5.310std::_Mu< _Arg, false, false > Class Template Reference	1472
5.310.1 Detailed Description	1472
5.311std::_Mu< _Arg, false, true > Class Template Reference	1472
5.311.1 Detailed Description	1472
5.312std::_Mu< _Arg, true, false > Class Template Reference	1473
5.312.1 Detailed Description	1473
5.313std::_Mu< reference_wrapper< _Tp >, false, false > Class Template Reference	1473
5.313.1 Detailed Description	1474
5.314std::_Placeholder< _Num > Struct Template Reference	1474
5.314.1 Detailed Description	1474
5.315std::_Reference_wrapper_base< _Tp > Struct Template Reference	1475
5.315.1 Detailed Description	1475
5.316std::_Safe_tuple_element< __i, _Tuple > Struct Template Reference	1476
5.316.1 Detailed Description	1476
5.317std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe > Struct Template Reference	1476
5.317.1 Detailed Description	1477
5.318std::_Safe_tuple_element_impl< __i, _Tuple, false > Struct Template Reference	1477
5.318.1 Detailed Description	1477
5.319std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference	1478
5.319.1 Detailed Description	1479
5.319.2 Constructor & Destructor Documentation	1479
5.319.2.1 _Temporary_buffer	1479
5.319.3 Member Function Documentation	1479

5.319.3.1 begin	1479
5.319.3.2 end	1479
5.319.3.3 requested_size	1480
5.319.3.4 size	1480
5.320std::_Tuple_impl< _Idx > Struct Template Reference	1480
5.320.1 Detailed Description	1480
5.321std::_Tuple_impl< _Idx, _Head, _Tail...> Struct Template Reference	1481
5.321.1 Detailed Description	1482
5.322std::_Vector_base< _Tp, _Alloc > Struct Template Reference	1482
5.322.1 Detailed Description	1483
5.323std::_Weak_result_type< _Functor > Struct Template Reference	1483
5.323.1 Detailed Description	1483
5.324std::_Weak_result_type_impl< _Functor > Struct Template Reference	1484
5.324.1 Detailed Description	1484
5.325std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)> Struct Template Reference	1484
5.325.1 Detailed Description	1484
5.326std::_Weak_result_type_impl< _Res(*)(_ArgTypes...)> Struct Template Reference	1485
5.326.1 Detailed Description	1485
5.327std::_Weak_result_type_impl< _Res(_ArgTypes...)> Struct Template Reference	1485
5.327.1 Detailed Description	1486
5.328std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const > Struct Template Reference	1486
5.328.1 Detailed Description	1486
5.329std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile > Struct Template Reference	1486
5.329.1 Detailed Description	1487
5.330std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile > Struct Template Reference	1487
5.330.1 Detailed Description	1487

5.331std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)> Struct Template Reference	1488
5.331.1 Detailed Description	1488
5.332std::add_lvalue_reference< _Tp > Struct Template Reference	1488
5.332.1 Detailed Description	1488
5.333std::add_rvalue_reference< _Tp > Struct Template Reference	1489
5.333.1 Detailed Description	1489
5.334std::adopt_lock_t Struct Reference	1489
5.334.1 Detailed Description	1489
5.335std::aligned_storage< _Len, _Align > Struct Template Reference . .	1490
5.335.1 Detailed Description	1490
5.336std::allocator< _Tp > Class Template Reference	1490
5.336.1 Detailed Description	1492
5.337std::allocator< void > Class Template Reference	1492
5.337.1 Detailed Description	1492
5.338std::allocator_arg_t Struct Reference	1493
5.338.1 Detailed Description	1493
5.339std::atomic< _Tp > Struct Template Reference	1493
5.339.1 Detailed Description	1494
5.340std::atomic< _Tp * > Struct Template Reference	1494
5.340.1 Detailed Description	1495
5.341std::atomic< bool > Struct Template Reference	1495
5.341.1 Detailed Description	1495
5.342std::atomic< char > Struct Template Reference	1496
5.342.1 Detailed Description	1496
5.343std::atomic< char16_t > Struct Template Reference	1496
5.343.1 Detailed Description	1497
5.344std::atomic< char32_t > Struct Template Reference	1497
5.344.1 Detailed Description	1498
5.345std::atomic< int > Struct Template Reference	1498
5.345.1 Detailed Description	1499

5.346std::atomic< long > Struct Template Reference	1499
5.346.1 Detailed Description	1499
5.347std::atomic< long long > Struct Template Reference	1500
5.347.1 Detailed Description	1500
5.348std::atomic< short > Struct Template Reference	1500
5.348.1 Detailed Description	1501
5.349std::atomic< signed char > Struct Template Reference	1501
5.349.1 Detailed Description	1502
5.350std::atomic< unsigned char > Struct Template Reference	1502
5.350.1 Detailed Description	1503
5.351std::atomic< unsigned int > Struct Template Reference	1503
5.351.1 Detailed Description	1504
5.352std::atomic< unsigned long > Struct Template Reference	1504
5.352.1 Detailed Description	1504
5.353std::atomic< unsigned long long > Struct Template Reference	1505
5.353.1 Detailed Description	1505
5.354std::atomic< unsigned short > Struct Template Reference	1505
5.354.1 Detailed Description	1506
5.355std::atomic< void * > Struct Template Reference	1506
5.355.1 Detailed Description	1507
5.356std::atomic< wchar_t > Struct Template Reference	1507
5.356.1 Detailed Description	1508
5.357std::auto_ptr< _Tp > Class Template Reference	1508
5.357.1 Detailed Description	1509
5.357.2 Member Typedef Documentation	1509
5.357.2.1 element_type	1509
5.357.3 Constructor & Destructor Documentation	1509
5.357.3.1 auto_ptr	1509
5.357.3.2 auto_ptr	1510
5.357.3.3 auto_ptr	1510
5.357.3.4 ~auto_ptr	1510

5.357.3.5 auto_ptr	1511
5.357.4 Member Function Documentation	1511
5.357.4.1 get	1511
5.357.4.2 operator*	1511
5.357.4.3 operator->	1512
5.357.4.4 operator=	1512
5.357.4.5 operator=	1512
5.357.4.6 release	1512
5.357.4.7 reset	1513
5.358std::auto_ptr_ref< _Tp1 > Struct Template Reference	1513
5.358.1 Detailed Description	1514
5.359std::back_insert_iterator< _Container > Class Template Reference . .	1514
5.359.1 Detailed Description	1515
5.359.2 Member Typedef Documentation	1515
5.359.2.1 container_type	1515
5.359.2.2 difference_type	1516
5.359.2.3 iterator_category	1516
5.359.2.4 pointer	1516
5.359.2.5 reference	1516
5.359.2.6 value_type	1516
5.359.3 Constructor & Destructor Documentation	1516
5.359.3.1 back_insert_iterator	1516
5.359.4 Member Function Documentation	1517
5.359.4.1 operator*	1517
5.359.4.2 operator++	1517
5.359.4.3 operator++	1517
5.359.4.4 operator=	1517
5.360std::bad_alloc Class Reference	1518
5.360.1 Detailed Description	1518
5.360.2 Member Function Documentation	1518
5.360.2.1 what	1518

5.361std::bad_cast Class Reference	1519
5.361.1 Detailed Description	1519
5.361.2 Member Function Documentation	1520
5.361.2.1 what	1520
5.362std::bad_exception Class Reference	1520
5.362.1 Detailed Description	1520
5.362.2 Member Function Documentation	1521
5.362.2.1 what	1521
5.363std::bad_function_call Class Reference	1521
5.363.1 Detailed Description	1521
5.363.2 Member Function Documentation	1522
5.363.2.1 what	1522
5.364std::bad_typeid Class Reference	1522
5.364.1 Detailed Description	1523
5.364.2 Member Function Documentation	1523
5.364.2.1 what	1523
5.365std::basic_filebuf< _CharT, _Traits > Class Template Reference . . .	1523
5.365.1 Detailed Description	1527
5.365.2 Member Typedef Documentation	1527
5.365.2.1 __streambuf_type	1527
5.365.2.2 char_type	1527
5.365.2.3 int_type	1528
5.365.2.4 off_type	1528
5.365.2.5 pos_type	1528
5.365.2.6 traits_type	1529
5.365.3 Constructor & Destructor Documentation	1529
5.365.3.1 basic_filebuf	1529
5.365.3.2 ~basic_filebuf	1529
5.365.4 Member Function Documentation	1529
5.365.4.1 _M_create_pback	1529
5.365.4.2 _M_destroy_pback	1530

5.365.4.3 _M_set_buffer	1530
5.365.4.4 close	1530
5.365.4.5 eback	1531
5.365.4.6 egptr	1531
5.365.4.7 epptr	1531
5.365.4.8 gbump	1532
5.365.4.9 getloc	1532
5.365.4.10gptr	1532
5.365.4.11imbue	1533
5.365.4.12n_avail	1533
5.365.4.13is_open	1534
5.365.4.14open	1534
5.365.4.15open	1535
5.365.4.16overflow	1535
5.365.4.17pbackfail	1536
5.365.4.18pbase	1536
5.365.4.19pbump	1537
5.365.4.20pptr	1537
5.365.4.21pubimbue	1538
5.365.4.22pubseekoff	1538
5.365.4.23pubseekpos	1538
5.365.4.24pubsetbuf	1539
5.365.4.25pubsync	1539
5.365.4.26sbumpc	1539
5.365.4.27seekoff	1540
5.365.4.28seekpos	1540
5.365.4.29setbuf	1540
5.365.4.30setbuf	1541
5.365.4.31setg	1541
5.365.4.32setp	1542
5.365.4.33setc	1542

5.365.4.34	getn	1543
5.365.4.35	showmanyc	1543
5.365.4.36	nextc	1544
5.365.4.37	sputbackc	1544
5.365.4.38	putc	1544
5.365.4.39	sputn	1545
5.365.4.40	stoss	1545
5.365.4.41	sungetc	1546
5.365.4.42	sync	1546
5.365.4.43	uflow	1546
5.365.4.44	underflow	1547
5.365.4.45	xsgetn	1548
5.365.4.46	xspn	1548
5.365.5	Member Data Documentation	1549
5.365.5.1	_M_buf	1549
5.365.5.2	_M_buf_locale	1549
5.365.5.3	_M_buf_size	1549
5.365.5.4	_M_ext_buf	1550
5.365.5.5	_M_ext_buf_size	1550
5.365.5.6	_M_ext_next	1550
5.365.5.7	_M_in_beg	1550
5.365.5.8	_M_in_cur	1551
5.365.5.9	_M_in_end	1551
5.365.5.10	_M_mode	1551
5.365.5.11	_M_out_beg	1552
5.365.5.12	_M_out_cur	1552
5.365.5.13	_M_out_end	1552
5.365.5.14	_M_pback	1553
5.365.5.15	_M_pback_cur_save	1553
5.365.5.16	_M_pback_end_save	1553
5.365.5.17	_M_pback_init	1553

5.365.5.18_M_reading	1554
5.366std::basic_fstream< _CharT, _Traits > Class Template Reference . .	1554
5.366.1 Detailed Description	1562
5.366.2 Member Typedef Documentation	1563
5.366.2.1 __ctype_type	1563
5.366.2.2 __ctype_type	1563
5.366.2.3 __num_get_type	1563
5.366.2.4 __num_put_type	1563
5.366.2.5 __num_put_type	1564
5.366.2.6 char_type	1564
5.366.2.7 char_type	1564
5.366.2.8 event_callback	1564
5.366.2.9 fmtflags	1565
5.366.2.10int_type	1566
5.366.2.11int_type	1566
5.366.2.12istate	1566
5.366.2.13off_type	1567
5.366.2.14off_type	1567
5.366.2.15openmode	1567
5.366.2.16pos_type	1568
5.366.2.17pos_type	1568
5.366.2.18seekdir	1568
5.366.2.19traits_type	1568
5.366.2.20traits_type	1569
5.366.3 Member Enumeration Documentation	1569
5.366.3.1 event	1569
5.366.4 Constructor & Destructor Documentation	1569
5.366.4.1 basic_fstream	1569
5.366.4.2 basic_fstream	1570
5.366.4.3 basic_fstream	1570
5.366.4.4 ~basic_fstream	1570

5.366.5 Member Function Documentation	1571
5.366.5.1 _M_getloc	1571
5.366.5.2 _M_write	1571
5.366.5.3 bad	1571
5.366.5.4 clear	1572
5.366.5.5 close	1572
5.366.5.6 copyfmt	1572
5.366.5.7 eof	1573
5.366.5.8 exceptions	1573
5.366.5.9 exceptions	1574
5.366.5.10fail	1574
5.366.5.11fill	1575
5.366.5.12fill	1575
5.366.5.13flags	1575
5.366.5.14flags	1576
5.366.5.15flush	1576
5.366.5.16gcount	1577
5.366.5.17get	1577
5.366.5.18get	1577
5.366.5.19get	1578
5.366.5.20get	1579
5.366.5.21get	1579
5.366.5.22get	1580
5.366.5.23getline	1580
5.366.5.24getline	1581
5.366.5.25getloc	1581
5.366.5.26good	1582
5.366.5.27ignore	1582
5.366.5.28ignore	1583
5.366.5.29ignore	1583
5.366.5.30mbue	1584

5.366.5.3	limit	1584
5.366.5.32	is_open	1585
5.366.5.33	word	1585
5.366.5.34	narrow	1585
5.366.5.35	open	1586
5.366.5.36	open	1586
5.366.5.37	operator void *	1587
5.366.5.38	operator!	1587
5.366.5.39	operator<<	1587
5.366.5.40	operator<<	1587
5.366.5.41	operator<<	1588
5.366.5.42	operator<<	1588
5.366.5.43	operator<<	1589
5.366.5.44	operator<<	1589
5.366.5.45	operator<<	1589
5.366.5.46	operator<<	1589
5.366.5.47	operator<<	1590
5.366.5.48	operator<<	1590
5.366.5.49	operator<<	1591
5.366.5.50	operator<<	1591
5.366.5.51	operator<<	1591
5.366.5.52	operator<<	1592
5.366.5.53	operator<<	1592
5.366.5.54	operator<<	1593
5.366.5.55	operator<<	1593
5.366.5.56	operator>>	1594
5.366.5.57	operator>>	1594
5.366.5.58	operator>>	1594
5.366.5.59	operator>>	1595
5.366.5.60	operator>>	1595
5.366.5.61	operator>>	1595

5.366.5.62operator>>	1596
5.366.5.63operator>>	1596
5.366.5.64operator>>	1596
5.366.5.65operator>>	1597
5.366.5.66operator>>	1597
5.366.5.67operator>>	1598
5.366.5.68operator>>	1598
5.366.5.69operator>>	1599
5.366.5.70operator>>	1599
5.366.5.71operator>>	1599
5.366.5.72operator>>	1600
5.366.5.73peek	1600
5.366.5.74precision	1601
5.366.5.75precision	1601
5.366.5.76put	1601
5.366.5.77putback	1602
5.366.5.78pword	1602
5.366.5.79rdbuf	1603
5.366.5.80rdbuf	1603
5.366.5.81rdstate	1604
5.366.5.82read	1604
5.366.5.83readsome	1605
5.366.5.84register_callback	1605
5.366.5.85seekg	1606
5.366.5.86seekg	1606
5.366.5.87seekp	1607
5.366.5.88seekp	1607
5.366.5.89setf	1608
5.366.5.90setf	1608
5.366.5.91setstate	1609
5.366.5.92sync	1609

5.366.5.93	sync_with_stdio	1610
5.366.5.94	tellg	1610
5.366.5.95	tellp	1611
5.366.5.96	tie	1611
5.366.5.97	tie	1612
5.366.5.98	unget	1612
5.366.5.99	unsetf	1612
5.366.5.100	widen	1613
5.366.5.101	width	1613
5.366.5.102	width	1614
5.366.5.103	write	1614
5.366.5.104	xalloc	1615
5.366.6	Member Data Documentation	1615
5.366.6.1	_M_gcount	1615
5.366.6.2	adjustfield	1615
5.366.6.3	app	1615
5.366.6.4	ate	1616
5.366.6.5	badbit	1616
5.366.6.6	basefield	1616
5.366.6.7	beg	1616
5.366.6.8	binary	1616
5.366.6.9	boolalpha	1617
5.366.6.10	cur	1617
5.366.6.11	ldec	1617
5.366.6.12	end	1617
5.366.6.13	eofbit	1617
5.366.6.14	failbit	1618
5.366.6.15	fixed	1618
5.366.6.16	floatfield	1618
5.366.6.17	goodbit	1618
5.366.6.18	hex	1619

5.366.6.19in	1619
5.366.6.20internal	1619
5.366.6.21left	1619
5.366.6.22oct	1620
5.366.6.23out	1620
5.366.6.24right	1620
5.366.6.25scientific	1620
5.366.6.26showbase	1620
5.366.6.27showpoint	1620
5.366.6.28showpos	1620
5.366.6.29skipws	1621
5.366.6.30trunc	1621
5.366.6.31unitbuf	1621
5.366.6.32uppercase	1621
5.367std::basic_ifstream< _CharT, _Traits > Class Template Reference . .	1621
5.367.1 Detailed Description	1628
5.367.2 Member Typedef Documentation	1628
5.367.2.1 __ctype_type	1628
5.367.2.2 __num_get_type	1628
5.367.2.3 __num_put_type	1629
5.367.2.4 char_type	1629
5.367.2.5 event_callback	1629
5.367.2.6 fmtflags	1629
5.367.2.7 int_type	1630
5.367.2.8 iostate	1631
5.367.2.9 off_type	1631
5.367.2.10openmode	1631
5.367.2.11pos_type	1632
5.367.2.12seekdir	1632
5.367.2.13traits_type	1632
5.367.3 Member Enumeration Documentation	1632

5.367.3.1 event	1632
5.367.4 Constructor & Destructor Documentation	1633
5.367.4.1 basic_ifstream	1633
5.367.4.2 basic_ifstream	1633
5.367.4.3 basic_ifstream	1633
5.367.4.4 ~basic_ifstream	1634
5.367.5 Member Function Documentation	1634
5.367.5.1 _M_getloc	1634
5.367.5.2 bad	1634
5.367.5.3 clear	1635
5.367.5.4 close	1635
5.367.5.5 copyfmt	1635
5.367.5.6 eof	1636
5.367.5.7 exceptions	1636
5.367.5.8 exceptions	1636
5.367.5.9 fail	1637
5.367.5.10 fill	1637
5.367.5.11 fill	1638
5.367.5.12 flags	1638
5.367.5.13 flags	1639
5.367.5.14 gcount	1639
5.367.5.15 get	1639
5.367.5.16 get	1640
5.367.5.17 get	1640
5.367.5.18 get	1641
5.367.5.19 get	1642
5.367.5.20 get	1642
5.367.5.21 getline	1643
5.367.5.22 getline	1643
5.367.5.23 getloc	1644
5.367.5.24 good	1644

5.367.5.25ignore	1645
5.367.5.26ignore	1645
5.367.5.27ignore	1646
5.367.5.28mbue	1646
5.367.5.29nit	1647
5.367.5.30s_open	1647
5.367.5.31iword	1647
5.367.5.32narrow	1648
5.367.5.33open	1648
5.367.5.34open	1649
5.367.5.35operator void *	1649
5.367.5.36operator!	1649
5.367.5.37operator>>	1649
5.367.5.38operator>>	1650
5.367.5.39operator>>	1650
5.367.5.40operator>>	1650
5.367.5.41operator>>	1651
5.367.5.42operator>>	1651
5.367.5.43operator>>	1651
5.367.5.44operator>>	1652
5.367.5.45operator>>	1652
5.367.5.46operator>>	1653
5.367.5.47operator>>	1653
5.367.5.48operator>>	1654
5.367.5.49operator>>	1654
5.367.5.50operator>>	1655
5.367.5.51operator>>	1655
5.367.5.52operator>>	1655
5.367.5.53operator>>	1656
5.367.5.54peek	1656
5.367.5.55precision	1656

5.367.5.56precision	1657
5.367.5.57putback	1657
5.367.5.58pword	1658
5.367.5.59rdbuf	1658
5.367.5.60rdbuf	1659
5.367.5.61rdstate	1659
5.367.5.62read	1659
5.367.5.63readsome	1660
5.367.5.64register_callback	1661
5.367.5.65seekg	1661
5.367.5.66seekg	1662
5.367.5.67setf	1662
5.367.5.68setf	1663
5.367.5.69setstate	1663
5.367.5.70sync	1664
5.367.5.71sync_with_stdio	1664
5.367.5.72tellg	1665
5.367.5.73tie	1665
5.367.5.74tie	1666
5.367.5.75unset	1666
5.367.5.76unsetf	1666
5.367.5.77widen	1667
5.367.5.78width	1667
5.367.5.79width	1668
5.367.5.80xalloc	1668
5.367.6 Member Data Documentation	1668
5.367.6.1 _M_gcount	1668
5.367.6.2 adjustfield	1669
5.367.6.3 app	1669
5.367.6.4 ate	1669
5.367.6.5 badbit	1669

5.367.6.6 basefield	1669
5.367.6.7 beg	1670
5.367.6.8 binary	1670
5.367.6.9 boolalpha	1670
5.367.6.10cur	1670
5.367.6.11dec	1670
5.367.6.12end	1670
5.367.6.13eofbit	1671
5.367.6.14failbit	1671
5.367.6.15fixed	1671
5.367.6.16floatfield	1671
5.367.6.17goodbit	1672
5.367.6.18hex	1672
5.367.6.19n	1672
5.367.6.20internal	1672
5.367.6.21left	1673
5.367.6.22oct	1673
5.367.6.23out	1673
5.367.6.24right	1673
5.367.6.25scientific	1673
5.367.6.26showbase	1673
5.367.6.27showpoint	1674
5.367.6.28showpos	1674
5.367.6.29skipws	1674
5.367.6.30trunc	1674
5.367.6.31unitbuf	1674
5.367.6.32uppercase	1674
5.368std::basic_ios<_CharT, _Traits> Class Template Reference	1675
5.368.1 Detailed Description	1678
5.368.2 Member Typedef Documentation	1679
5.368.2.1 __ctype_type	1679

5.368.2.2	__num_get_type	1679
5.368.2.3	__num_put_type	1679
5.368.2.4	char_type	1679
5.368.2.5	event_callback	1680
5.368.2.6	fmtflags	1680
5.368.2.7	int_type	1681
5.368.2.8	iostate	1681
5.368.2.9	off_type	1682
5.368.2.10	openmode	1682
5.368.2.11	lpos_type	1683
5.368.2.12	seekdir	1683
5.368.2.13	traits_type	1683
5.368.3	Member Enumeration Documentation	1684
5.368.3.1	event	1684
5.368.4	Constructor & Destructor Documentation	1684
5.368.4.1	basic_ios	1684
5.368.4.2	~basic_ios	1684
5.368.4.3	basic_ios	1684
5.368.5	Member Function Documentation	1685
5.368.5.1	_M_getloc	1685
5.368.5.2	bad	1685
5.368.5.3	clear	1685
5.368.5.4	copyfmt	1686
5.368.5.5	eof	1686
5.368.5.6	exceptions	1686
5.368.5.7	exceptions	1687
5.368.5.8	fail	1687
5.368.5.9	fill	1688
5.368.5.10	fill	1688
5.368.5.11	iflags	1689
5.368.5.12	iflags	1689

5.368.5.13	getloc	1689
5.368.5.14	good	1690
5.368.5.15	imbue	1690
5.368.5.16	init	1690
5.368.5.17	word	1691
5.368.5.18	narrow	1691
5.368.5.19	operator void *	1692
5.368.5.20	operator!	1692
5.368.5.21	precision	1692
5.368.5.22	precision	1692
5.368.5.23	pword	1693
5.368.5.24	rdbuf	1693
5.368.5.25	rdbuf	1694
5.368.5.26	rdstate	1694
5.368.5.27	register_callback	1695
5.368.5.28	setf	1695
5.368.5.29	setf	1695
5.368.5.30	setstate	1696
5.368.5.31	sync_with_stdio	1696
5.368.5.32	tie	1697
5.368.5.33	tie	1697
5.368.5.34	unsetf	1697
5.368.5.35	widen	1698
5.368.5.36	width	1698
5.368.5.37	width	1699
5.368.5.38	xalloc	1699
5.368.6	Member Data Documentation	1699
5.368.6.1	adjustfield	1699
5.368.6.2	app	1699
5.368.6.3	ate	1700
5.368.6.4	badbit	1700

5.368.6.5 basefield	1700
5.368.6.6 beg	1700
5.368.6.7 binary	1700
5.368.6.8 boolalpha	1701
5.368.6.9 cur	1701
5.368.6.10dec	1701
5.368.6.11lend	1701
5.368.6.12eofbit	1701
5.368.6.13failbit	1702
5.368.6.14fixed	1702
5.368.6.15floatfield	1702
5.368.6.16goodbit	1702
5.368.6.17hex	1703
5.368.6.18in	1703
5.368.6.19internal	1703
5.368.6.20left	1703
5.368.6.21loct	1704
5.368.6.22out	1704
5.368.6.23right	1704
5.368.6.24scientific	1704
5.368.6.25showbase	1704
5.368.6.26showpoint	1704
5.368.6.27showpos	1704
5.368.6.28skipws	1705
5.368.6.29trunc	1705
5.368.6.30unitbuf	1705
5.368.6.31uppercase	1705
5.369std::basic_istream< _CharT, _Traits > Class Template Reference . .	1705
5.369.1 Detailed Description	1713
5.369.2 Member Typedef Documentation	1713
5.369.2.1 __ctype_type	1713

5.369.2.2	__ctype_type	1713
5.369.2.3	__num_get_type	1713
5.369.2.4	__num_put_type	1714
5.369.2.5	__num_put_type	1714
5.369.2.6	char_type	1714
5.369.2.7	char_type	1714
5.369.2.8	event_callback	1715
5.369.2.9	fmtflags	1715
5.369.2.10	int_type	1716
5.369.2.11	lint_type	1716
5.369.2.12	iostate	1716
5.369.2.13	off_type	1717
5.369.2.14	off_type	1717
5.369.2.15	openmode	1717
5.369.2.16	pos_type	1718
5.369.2.17	pos_type	1718
5.369.2.18	seekdir	1718
5.369.2.19	traits_type	1719
5.369.2.20	traits_type	1719
5.369.3	Member Enumeration Documentation	1719
5.369.3.1	event	1719
5.369.4	Constructor & Destructor Documentation	1720
5.369.4.1	basic_iostream	1720
5.369.4.2	~basic_iostream	1720
5.369.5	Member Function Documentation	1720
5.369.5.1	_M_getloc	1720
5.369.5.2	_M_write	1721
5.369.5.3	bad	1721
5.369.5.4	clear	1721
5.369.5.5	copyfmt	1722
5.369.5.6	eof	1722

5.369.5.7 exceptions	1723
5.369.5.8 exceptions	1723
5.369.5.9 fail	1724
5.369.5.10fill	1724
5.369.5.11fill	1724
5.369.5.12flags	1725
5.369.5.13flags	1725
5.369.5.14flush	1725
5.369.5.15gcount	1726
5.369.5.16get	1726
5.369.5.17get	1727
5.369.5.18get	1727
5.369.5.19get	1728
5.369.5.20get	1728
5.369.5.21get	1729
5.369.5.22getline	1730
5.369.5.23getline	1730
5.369.5.24getloc	1731
5.369.5.25good	1731
5.369.5.26ignore	1732
5.369.5.27ignore	1732
5.369.5.28ignore	1733
5.369.5.29imbue	1733
5.369.5.30init	1734
5.369.5.31iword	1734
5.369.5.32narrow	1734
5.369.5.33operator void *	1735
5.369.5.34operator!	1735
5.369.5.35operator<<	1735
5.369.5.36operator<<	1736
5.369.5.37operator<<	1736

5.369.5.38operator<<	1737
5.369.5.39operator<<	1737
5.369.5.40operator<<	1737
5.369.5.41operator<<	1737
5.369.5.42operator<<	1738
5.369.5.43operator<<	1738
5.369.5.44operator<<	1739
5.369.5.45operator<<	1739
5.369.5.46operator<<	1739
5.369.5.47operator<<	1740
5.369.5.48operator<<	1740
5.369.5.49operator<<	1741
5.369.5.50operator<<	1741
5.369.5.51operator<<	1742
5.369.5.52operator>>	1742
5.369.5.53operator>>	1742
5.369.5.54operator>>	1743
5.369.5.55operator>>	1743
5.369.5.56operator>>	1744
5.369.5.57operator>>	1744
5.369.5.58operator>>	1744
5.369.5.59operator>>	1745
5.369.5.60operator>>	1745
5.369.5.61operator>>	1746
5.369.5.62operator>>	1746
5.369.5.63operator>>	1746
5.369.5.64operator>>	1747
5.369.5.65operator>>	1747
5.369.5.66operator>>	1747
5.369.5.67operator>>	1748
5.369.5.68operator>>	1748

5.369.5.69peek	1749
5.369.5.70precision	1749
5.369.5.71precision	1749
5.369.5.72put	1750
5.369.5.73putback	1750
5.369.5.74pword	1751
5.369.5.75rdbuf	1751
5.369.5.76rdbuf	1752
5.369.5.77rdstate	1753
5.369.5.78read	1753
5.369.5.79readsome	1754
5.369.5.80register_callback	1754
5.369.5.81seekg	1755
5.369.5.82seekg	1755
5.369.5.83seekp	1756
5.369.5.84seekp	1756
5.369.5.85setf	1757
5.369.5.86setf	1757
5.369.5.87setstate	1758
5.369.5.88sync	1758
5.369.5.89sync_with_stdio	1759
5.369.5.90tellg	1759
5.369.5.91tellp	1760
5.369.5.92tie	1760
5.369.5.93tie	1760
5.369.5.94unget	1761
5.369.5.95unsetf	1761
5.369.5.96widen	1762
5.369.5.97width	1762
5.369.5.98width	1762
5.369.5.99write	1763

5.369.5.10	alloc	1763
5.369.6	Member Data Documentation	1764
5.369.6.1	_M_gcount	1764
5.369.6.2	adjustfield	1764
5.369.6.3	app	1764
5.369.6.4	ate	1764
5.369.6.5	badbit	1764
5.369.6.6	basefield	1765
5.369.6.7	beg	1765
5.369.6.8	binary	1765
5.369.6.9	boolalpha	1765
5.369.6.10	cur	1766
5.369.6.11	dec	1766
5.369.6.12	end	1766
5.369.6.13	eofbit	1766
5.369.6.14	failbit	1766
5.369.6.15	fixed	1767
5.369.6.16	floatfield	1767
5.369.6.17	goodbit	1767
5.369.6.18	hex	1767
5.369.6.19	in	1768
5.369.6.20	internal	1768
5.369.6.21	left	1768
5.369.6.22	oct	1768
5.369.6.23	out	1768
5.369.6.24	right	1769
5.369.6.25	scientific	1769
5.369.6.26	showbase	1769
5.369.6.27	showpoint	1769
5.369.6.28	showpos	1769
5.369.6.29	skipws	1769

5.369.6.30trunc	1769
5.369.6.3lunitbuf	1770
5.369.6.32uppercase	1770
5.370std::basic_istream< _CharT, _Traits > Class Template Reference . .	1770
5.370.1 Detailed Description	1776
5.370.2 Member Typedef Documentation	1776
5.370.2.1 __ctype_type	1776
5.370.2.2 __num_get_type	1777
5.370.2.3 __num_put_type	1777
5.370.2.4 char_type	1777
5.370.2.5 event_callback	1777
5.370.2.6 fmtflags	1778
5.370.2.7 int_type	1779
5.370.2.8 iostate	1779
5.370.2.9 off_type	1779
5.370.2.10openmode	1779
5.370.2.11pos_type	1780
5.370.2.12seekdir	1780
5.370.2.13traits_type	1781
5.370.3 Member Enumeration Documentation	1781
5.370.3.1 event	1781
5.370.4 Constructor & Destructor Documentation	1781
5.370.4.1 basic_istream	1781
5.370.4.2 ~basic_istream	1781
5.370.5 Member Function Documentation	1782
5.370.5.1 _M_getloc	1782
5.370.5.2 bad	1782
5.370.5.3 clear	1782
5.370.5.4 copyfmt	1783
5.370.5.5 eof	1783
5.370.5.6 exceptions	1784

5.370.5.7 exceptions	1784
5.370.5.8 fail	1785
5.370.5.9 fill	1785
5.370.5.10fill	1785
5.370.5.11flags	1786
5.370.5.12flags	1786
5.370.5.13gcount	1786
5.370.5.14get	1787
5.370.5.15get	1787
5.370.5.16get	1788
5.370.5.17get	1788
5.370.5.18get	1789
5.370.5.19get	1790
5.370.5.20getline	1790
5.370.5.21getline	1791
5.370.5.22getloc	1791
5.370.5.23good	1792
5.370.5.24ignore	1792
5.370.5.25ignore	1793
5.370.5.26ignore	1793
5.370.5.27imbue	1793
5.370.5.28init	1794
5.370.5.29word	1794
5.370.5.30narrow	1795
5.370.5.31operator void *	1795
5.370.5.32operator!	1795
5.370.5.33operator>>	1796
5.370.5.34operator>>	1796
5.370.5.35operator>>	1797
5.370.5.36operator>>	1797
5.370.5.37operator>>	1797

5.370.5.38operator>>	1798
5.370.5.39operator>>	1798
5.370.5.40operator>>	1798
5.370.5.41operator>>	1799
5.370.5.42operator>>	1799
5.370.5.43operator>>	1799
5.370.5.44operator>>	1800
5.370.5.45operator>>	1800
5.370.5.46operator>>	1800
5.370.5.47operator>>	1801
5.370.5.48operator>>	1801
5.370.5.49operator>>	1802
5.370.5.50peek	1802
5.370.5.51precision	1802
5.370.5.52precision	1803
5.370.5.53putback	1803
5.370.5.54pword	1804
5.370.5.55rdbuf	1804
5.370.5.56rdbuf	1805
5.370.5.57rdstate	1806
5.370.5.58read	1806
5.370.5.59readsome	1807
5.370.5.60register_callback	1807
5.370.5.61seekg	1808
5.370.5.62seekg	1808
5.370.5.63setf	1809
5.370.5.64setf	1809
5.370.5.65setstate	1810
5.370.5.66sync	1810
5.370.5.67sync_with_stdio	1811
5.370.5.68tellg	1811

5.370.5.69ie	1812
5.370.5.70ie	1812
5.370.5.71unget	1812
5.370.5.72unsetf	1813
5.370.5.73widen	1813
5.370.5.74width	1814
5.370.5.75width	1814
5.370.5.76xalloc	1814
5.370.6 Member Data Documentation	1815
5.370.6.1 _M_gcount	1815
5.370.6.2 adjustfield	1815
5.370.6.3 app	1815
5.370.6.4 ate	1815
5.370.6.5 badbit	1816
5.370.6.6 basefield	1816
5.370.6.7 beg	1816
5.370.6.8 binary	1816
5.370.6.9 boolalpha	1817
5.370.6.10cur	1817
5.370.6.11dec	1817
5.370.6.12end	1817
5.370.6.13eofbit	1817
5.370.6.14failbit	1818
5.370.6.15fixed	1818
5.370.6.16floatfield	1818
5.370.6.17goodbit	1818
5.370.6.18hex	1819
5.370.6.19in	1819
5.370.6.20internal	1819
5.370.6.21left	1819
5.370.6.22oct	1819

5.370.6.23out	1819
5.370.6.24right	1820
5.370.6.25scientific	1820
5.370.6.26showbase	1820
5.370.6.27showpoint	1820
5.370.6.28showpos	1820
5.370.6.29skipws	1820
5.370.6.30trunc	1821
5.370.6.31unitbuf	1821
5.370.6.32uppercase	1821
5.371std::basic_istream< _CharT, _Traits >::sentry Class Reference	1821
5.371.1 Detailed Description	1822
5.371.2 Member Typedef Documentation	1822
5.371.2.1 traits_type	1822
5.371.3 Constructor & Destructor Documentation	1822
5.371.3.1 sentry	1822
5.371.4 Member Function Documentation	1823
5.371.4.1 operator bool	1823
5.372std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference	1823
5.372.1 Detailed Description	1830
5.372.2 Member Typedef Documentation	1830
5.372.2.1 __ctype_type	1830
5.372.2.2 __num_get_type	1830
5.372.2.3 __num_put_type	1830
5.372.2.4 char_type	1831
5.372.2.5 event_callback	1831
5.372.2.6 fmtflags	1831
5.372.2.7 int_type	1832
5.372.2.8 iostate	1832
5.372.2.9 off_type	1833

5.372.2.10openmode	1833
5.372.2.11pos_type	1834
5.372.2.12seekdir	1834
5.372.2.13traits_type	1834
5.372.3 Member Enumeration Documentation	1834
5.372.3.1 event	1834
5.372.4 Constructor & Destructor Documentation	1835
5.372.4.1 basic_istream	1835
5.372.4.2 basic_istream	1835
5.372.4.3 ~basic_istream	1836
5.372.5 Member Function Documentation	1836
5.372.5.1 _M_getloc	1836
5.372.5.2 bad	1836
5.372.5.3 clear	1837
5.372.5.4 copyfmt	1837
5.372.5.5 eof	1837
5.372.5.6 exceptions	1838
5.372.5.7 exceptions	1838
5.372.5.8 fail	1839
5.372.5.9 fill	1839
5.372.5.10fill	1840
5.372.5.11flags	1840
5.372.5.12flags	1840
5.372.5.13gcount	1841
5.372.5.14get	1841
5.372.5.15get	1841
5.372.5.16get	1842
5.372.5.17get	1842
5.372.5.18get	1843
5.372.5.19get	1844
5.372.5.20getline	1844

5.372.5.21getline	1845
5.372.5.22getloc	1846
5.372.5.23good	1846
5.372.5.24ignore	1846
5.372.5.25ignore	1847
5.372.5.26ignore	1848
5.372.5.27mbue	1848
5.372.5.28nit	1849
5.372.5.29word	1849
5.372.5.30narrow	1849
5.372.5.31operator void *	1850
5.372.5.32operator!	1850
5.372.5.33operator>>	1850
5.372.5.34operator>>	1851
5.372.5.35operator>>	1851
5.372.5.36operator>>	1851
5.372.5.37operator>>	1852
5.372.5.38operator>>	1852
5.372.5.39operator>>	1853
5.372.5.40operator>>	1853
5.372.5.41operator>>	1854
5.372.5.42operator>>	1854
5.372.5.43operator>>	1854
5.372.5.44operator>>	1855
5.372.5.45operator>>	1855
5.372.5.46operator>>	1855
5.372.5.47operator>>	1856
5.372.5.48operator>>	1856
5.372.5.49operator>>	1857
5.372.5.50peek	1857
5.372.5.51precision	1857

5.372.5.52precision	1858
5.372.5.53putback	1858
5.372.5.54pword	1859
5.372.5.55rdbuf	1859
5.372.5.56rdbuf	1859
5.372.5.57rdstate	1860
5.372.5.58read	1860
5.372.5.59readsome	1861
5.372.5.60register_callback	1862
5.372.5.61seekg	1862
5.372.5.62seekg	1863
5.372.5.63setf	1863
5.372.5.64setf	1864
5.372.5.65setstate	1864
5.372.5.66str	1865
5.372.5.67str	1865
5.372.5.68sync	1865
5.372.5.69sync_with_stdio	1866
5.372.5.70tellg	1866
5.372.5.71tie	1867
5.372.5.72tie	1867
5.372.5.73unget	1867
5.372.5.74unsetf	1868
5.372.5.75widen	1868
5.372.5.76width	1869
5.372.5.77width	1869
5.372.5.78xalloc	1869
5.372.6 Member Data Documentation	1870
5.372.6.1 _M_gcount	1870
5.372.6.2 adjustfield	1870
5.372.6.3 app	1870

5.372.6.4	ate	1870
5.372.6.5	badbit	1871
5.372.6.6	basefield	1871
5.372.6.7	beg	1871
5.372.6.8	binary	1871
5.372.6.9	boolalpha	1872
5.372.6.10	cur	1872
5.372.6.11	ldec	1872
5.372.6.12	end	1872
5.372.6.13	eofbit	1872
5.372.6.14	failbit	1873
5.372.6.15	fixed	1873
5.372.6.16	floatfield	1873
5.372.6.17	goodbit	1873
5.372.6.18	hex	1874
5.372.6.19	in	1874
5.372.6.20	internal	1874
5.372.6.21	lleft	1874
5.372.6.22	oct	1874
5.372.6.23	out	1875
5.372.6.24	right	1875
5.372.6.25	scientific	1875
5.372.6.26	showbase	1875
5.372.6.27	showpoint	1875
5.372.6.28	showpos	1875
5.372.6.29	skipws	1876
5.372.6.30	trunc	1876
5.372.6.31	unitbuf	1876
5.372.6.32	uppercase	1876
5.373	std::basic_ofstream<_CharT, _Traits> Class Template Reference	1876
5.373.1	Detailed Description	1882

5.373.2 Member Typedef Documentation	1883
5.373.2.1 __ctype_type	1883
5.373.2.2 __num_get_type	1883
5.373.2.3 __num_put_type	1883
5.373.2.4 char_type	1883
5.373.2.5 event_callback	1884
5.373.2.6 fmtflags	1884
5.373.2.7 int_type	1885
5.373.2.8 iostate	1885
5.373.2.9 off_type	1885
5.373.2.10 openmode	1886
5.373.2.11 pos_type	1886
5.373.2.12 seekdir	1886
5.373.2.13 traits_type	1887
5.373.3 Member Enumeration Documentation	1887
5.373.3.1 event	1887
5.373.4 Constructor & Destructor Documentation	1887
5.373.4.1 basic_ofstream	1887
5.373.4.2 basic_ofstream	1887
5.373.4.3 basic_ofstream	1888
5.373.4.4 ~basic_ofstream	1888
5.373.5 Member Function Documentation	1888
5.373.5.1 _M_getloc	1888
5.373.5.2 _M_write	1889
5.373.5.3 bad	1889
5.373.5.4 clear	1890
5.373.5.5 close	1890
5.373.5.6 copyfmt	1890
5.373.5.7 eof	1891
5.373.5.8 exceptions	1891
5.373.5.9 exceptions	1891

5.373.5.10fail	1892
5.373.5.11fill	1892
5.373.5.12fill	1893
5.373.5.13flags	1893
5.373.5.14flags	1893
5.373.5.15flush	1894
5.373.5.16getloc	1894
5.373.5.17good	1895
5.373.5.18mbue	1895
5.373.5.19nit	1895
5.373.5.20is_open	1896
5.373.5.21word	1896
5.373.5.22narrow	1896
5.373.5.23open	1897
5.373.5.24open	1897
5.373.5.25operator void *	1898
5.373.5.26operator!	1898
5.373.5.27operator<<	1898
5.373.5.28operator<<	1898
5.373.5.29operator<<	1899
5.373.5.30operator<<	1899
5.373.5.31operator<<	1900
5.373.5.32operator<<	1900
5.373.5.33operator<<	1900
5.373.5.34operator<<	1901
5.373.5.35operator<<	1901
5.373.5.36operator<<	1902
5.373.5.37operator<<	1902
5.373.5.38operator<<	1902
5.373.5.39operator<<	1903
5.373.5.40operator<<	1903

5.373.5.41operator<<	1904
5.373.5.42operator<<	1904
5.373.5.43operator<<	1904
5.373.5.44precision	1905
5.373.5.45precision	1905
5.373.5.46put	1905
5.373.5.47pword	1906
5.373.5.48rdbuf	1906
5.373.5.49rdbuf	1907
5.373.5.50rdstate	1907
5.373.5.51register_callback	1907
5.373.5.52seekp	1908
5.373.5.53seekp	1908
5.373.5.54setf	1909
5.373.5.55setf	1909
5.373.5.56setstate	1910
5.373.5.57sync_with_stdio	1910
5.373.5.58tellp	1911
5.373.5.59tie	1911
5.373.5.60tie	1911
5.373.5.61unsetf	1912
5.373.5.62widen	1912
5.373.5.63width	1913
5.373.5.64width	1913
5.373.5.65write	1913
5.373.5.66xalloc	1914
5.373.6 Member Data Documentation	1914
5.373.6.1 adjustfield	1914
5.373.6.2 app	1914
5.373.6.3 ate	1915
5.373.6.4 badbit	1915

5.373.6.5 basefield	1915
5.373.6.6 beg	1915
5.373.6.7 binary	1915
5.373.6.8 boolalpha	1916
5.373.6.9 cur	1916
5.373.6.10dec	1916
5.373.6.11lend	1916
5.373.6.12eofbit	1916
5.373.6.13failbit	1917
5.373.6.14fixed	1917
5.373.6.15floatfield	1917
5.373.6.16goodbit	1917
5.373.6.17hex	1918
5.373.6.18in	1918
5.373.6.19internal	1918
5.373.6.20left	1918
5.373.6.21loct	1919
5.373.6.22out	1919
5.373.6.23right	1919
5.373.6.24scientific	1919
5.373.6.25showbase	1919
5.373.6.26showpoint	1919
5.373.6.27showpos	1919
5.373.6.28skipws	1920
5.373.6.29trunc	1920
5.373.6.30unitbuf	1920
5.373.6.31uppercase	1920
5.374std::basic_ostream< _CharT, _Traits > Class Template Reference . .	1920
5.374.1 Detailed Description	1926
5.374.2 Member Typedef Documentation	1926
5.374.2.1 __ctype_type	1926

5.374.2.2	__num_get_type	1926
5.374.2.3	__num_put_type	1926
5.374.2.4	char_type	1927
5.374.2.5	event_callback	1927
5.374.2.6	fmtflags	1927
5.374.2.7	int_type	1928
5.374.2.8	iostate	1929
5.374.2.9	off_type	1929
5.374.2.10	openmode	1929
5.374.2.11	lpos_type	1930
5.374.2.12	seekdir	1930
5.374.2.13	traits_type	1930
5.374.3	Member Enumeration Documentation	1931
5.374.3.1	event	1931
5.374.4	Constructor & Destructor Documentation	1931
5.374.4.1	basic_ostream	1931
5.374.4.2	~basic_ostream	1931
5.374.5	Member Function Documentation	1931
5.374.5.1	_M_getloc	1931
5.374.5.2	_M_write	1932
5.374.5.3	bad	1932
5.374.5.4	clear	1933
5.374.5.5	copyfmt	1933
5.374.5.6	eof	1933
5.374.5.7	exceptions	1934
5.374.5.8	exceptions	1934
5.374.5.9	fail	1935
5.374.5.10	fill	1935
5.374.5.11	ifill	1936
5.374.5.12	flags	1936
5.374.5.13	flags	1936

5.374.5.14flush	1937
5.374.5.15getloc	1937
5.374.5.16good	1938
5.374.5.17mbue	1938
5.374.5.18nit	1938
5.374.5.19word	1939
5.374.5.20narrow	1939
5.374.5.21operator void *	1940
5.374.5.22operator!	1940
5.374.5.23operator<<	1940
5.374.5.24operator<<	1940
5.374.5.25operator<<	1941
5.374.5.26operator<<	1941
5.374.5.27operator<<	1942
5.374.5.28operator<<	1942
5.374.5.29operator<<	1942
5.374.5.30operator<<	1943
5.374.5.31operator<<	1943
5.374.5.32operator<<	1944
5.374.5.33operator<<	1944
5.374.5.34operator<<	1944
5.374.5.35operator<<	1945
5.374.5.36operator<<	1945
5.374.5.37operator<<	1945
5.374.5.38operator<<	1946
5.374.5.39operator<<	1946
5.374.5.40precision	1947
5.374.5.41precision	1947
5.374.5.42put	1947
5.374.5.43pword	1948
5.374.5.44rdbuf	1948

5.374.5.45	rdbuf	1949
5.374.5.46	rdstate	1949
5.374.5.47	register_callback	1950
5.374.5.48	seekp	1950
5.374.5.49	seekp	1951
5.374.5.50	setf	1951
5.374.5.51	lsetf	1951
5.374.5.52	setstate	1952
5.374.5.53	sync_with_stdio	1952
5.374.5.54	tellp	1953
5.374.5.55	tie	1953
5.374.5.56	tie	1954
5.374.5.57	unsetf	1954
5.374.5.58	widen	1954
5.374.5.59	width	1955
5.374.5.60	width	1955
5.374.5.61	lwrite	1955
5.374.5.62	xalloc	1956
5.374.6	Member Data Documentation	1956
5.374.6.1	adjustfield	1956
5.374.6.2	app	1957
5.374.6.3	ate	1957
5.374.6.4	badbit	1957
5.374.6.5	basefield	1957
5.374.6.6	beg	1957
5.374.6.7	binary	1958
5.374.6.8	boolalpha	1958
5.374.6.9	cur	1958
5.374.6.10	dec	1958
5.374.6.11	lend	1958
5.374.6.12	eofbit	1959

5.374.6.13failbit	1959
5.374.6.14fixed	1959
5.374.6.15floatfield	1959
5.374.6.16goodbit	1960
5.374.6.17hex	1960
5.374.6.18in	1960
5.374.6.19internal	1960
5.374.6.20left	1961
5.374.6.21loct	1961
5.374.6.22out	1961
5.374.6.23right	1961
5.374.6.24scientific	1961
5.374.6.25showbase	1961
5.374.6.26showpoint	1962
5.374.6.27showpos	1962
5.374.6.28skipws	1962
5.374.6.29trunc	1962
5.374.6.30unitbuf	1962
5.374.6.31uppercase	1962
5.375std::basic_ostream< _CharT, _Traits >::sentry Class Reference . . .	1963
5.375.1 Detailed Description	1963
5.375.2 Constructor & Destructor Documentation	1963
5.375.2.1 sentry	1963
5.375.2.2 ~sentry	1964
5.375.3 Member Function Documentation	1964
5.375.3.1 operator bool	1964
5.376std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference	1964
5.376.1 Detailed Description	1970
5.376.2 Member Typedef Documentation	1971
5.376.2.1 __ctype_type	1971

5.376.2.2	__num_get_type	1971
5.376.2.3	__num_put_type	1971
5.376.2.4	char_type	1971
5.376.2.5	event_callback	1972
5.376.2.6	fmtflags	1972
5.376.2.7	int_type	1973
5.376.2.8	iostate	1973
5.376.2.9	off_type	1973
5.376.2.10	openmode	1974
5.376.2.11	lpos_type	1974
5.376.2.12	seekdir	1974
5.376.2.13	traits_type	1975
5.376.3	Member Enumeration Documentation	1975
5.376.3.1	event	1975
5.376.4	Constructor & Destructor Documentation	1975
5.376.4.1	basic_ostringstream	1975
5.376.4.2	basic_ostringstream	1976
5.376.4.3	~basic_ostringstream	1976
5.376.5	Member Function Documentation	1976
5.376.5.1	_M_getloc	1976
5.376.5.2	_M_write	1977
5.376.5.3	bad	1977
5.376.5.4	clear	1977
5.376.5.5	copyfmt	1978
5.376.5.6	eof	1978
5.376.5.7	exceptions	1979
5.376.5.8	exceptions	1979
5.376.5.9	fail	1980
5.376.5.10	fill	1980
5.376.5.11	ifill	1981
5.376.5.12	flags	1981

5.376.5.13	flags	1981
5.376.5.14	flush	1982
5.376.5.15	getloc	1982
5.376.5.16	good	1982
5.376.5.17	mbue	1983
5.376.5.18	nit	1983
5.376.5.19	word	1983
5.376.5.20	narrow	1984
5.376.5.21	operator void *	1984
5.376.5.22	operator!	1985
5.376.5.23	operator<<	1985
5.376.5.24	operator<<	1985
5.376.5.25	operator<<	1986
5.376.5.26	operator<<	1986
5.376.5.27	operator<<	1986
5.376.5.28	operator<<	1987
5.376.5.29	operator<<	1987
5.376.5.30	operator<<	1987
5.376.5.31	operator<<	1988
5.376.5.32	operator<<	1988
5.376.5.33	operator<<	1989
5.376.5.34	operator<<	1989
5.376.5.35	operator<<	1989
5.376.5.36	operator<<	1990
5.376.5.37	operator<<	1990
5.376.5.38	operator<<	1991
5.376.5.39	operator<<	1991
5.376.5.40	precision	1991
5.376.5.41	precision	1992
5.376.5.42	put	1992
5.376.5.43	pwd	1993

5.376.5.44	rdbuf	1993
5.376.5.45	rdbuf	1994
5.376.5.46	rdstate	1994
5.376.5.47	register_callback	1994
5.376.5.48	seekp	1995
5.376.5.49	seekp	1995
5.376.5.50	setf	1996
5.376.5.51	setf	1996
5.376.5.52	setstate	1996
5.376.5.53	str	1997
5.376.5.54	str	1997
5.376.5.55	sync_with_stdio	1998
5.376.5.56	tellp	1998
5.376.5.57	tie	1998
5.376.5.58	tie	1999
5.376.5.59	unsetf	1999
5.376.5.60	widen	1999
5.376.5.61	width	2000
5.376.5.62	width	2000
5.376.5.63	write	2001
5.376.5.64	xalloc	2001
5.376.6	Member Data Documentation	2002
5.376.6.1	adjustfield	2002
5.376.6.2	app	2002
5.376.6.3	ate	2002
5.376.6.4	badbit	2002
5.376.6.5	basefield	2003
5.376.6.6	beg	2003
5.376.6.7	binary	2003
5.376.6.8	boolalpha	2003
5.376.6.9	cur	2003

5.376.6.10dec	2003
5.376.6.11lend	2004
5.376.6.12eofbit	2004
5.376.6.13failbit	2004
5.376.6.14fixed	2004
5.376.6.15floatfield	2005
5.376.6.16goodbit	2005
5.376.6.17hex	2005
5.376.6.18in	2005
5.376.6.19internal	2006
5.376.6.20left	2006
5.376.6.21loct	2006
5.376.6.22out	2006
5.376.6.23right	2006
5.376.6.24scientific	2006
5.376.6.25showbase	2007
5.376.6.26showpoint	2007
5.376.6.27showpos	2007
5.376.6.28skipws	2007
5.376.6.29trunc	2007
5.376.6.30unitbuf	2007
5.376.6.31uppercase	2007
5.377std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference	2008
5.377.1 Detailed Description	2010
5.377.2 Constructor & Destructor Documentation	2010
5.377.2.1 basic_regex	2010
5.377.2.2 basic_regex	2010
5.377.2.3 basic_regex	2011
5.377.2.4 basic_regex	2011
5.377.2.5 basic_regex	2011
5.377.2.6 basic_regex	2012

5.377.2.7 basic_regex	2012
5.377.2.8 basic_regex	2013
5.377.2.9 ~basic_regex	2013
5.377.3 Member Function Documentation	2013
5.377.3.1 assign	2013
5.377.3.2 assign	2014
5.377.3.3 assign	2014
5.377.3.4 assign	2015
5.377.3.5 assign	2015
5.377.3.6 assign	2016
5.377.3.7 assign	2016
5.377.3.8 flags	2016
5.377.3.9 getloc	2017
5.377.3.10 imbue	2017
5.377.3.11 mark_count	2017
5.377.3.12 operator=	2017
5.377.3.13 operator=	2017
5.377.3.14 operator=	2018
5.377.3.15 operator=	2018
5.377.3.16 swap	2018
5.378 std::basic_streambuf<_CharT, _Traits> Class Template Reference	2019
5.378.1 Detailed Description	2022
5.378.2 Member Typedef Documentation	2023
5.378.2.1 __streambuf_type	2023
5.378.2.2 char_type	2023
5.378.2.3 int_type	2024
5.378.2.4 off_type	2024
5.378.2.5 pos_type	2024
5.378.2.6 traits_type	2025
5.378.3 Constructor & Destructor Documentation	2025
5.378.3.1 ~basic_streambuf	2025

5.378.3.2 basic_streambuf	2025
5.378.4 Member Function Documentation	2026
5.378.4.1 eback	2026
5.378.4.2 egptr	2026
5.378.4.3 epptr	2026
5.378.4.4 gbump	2027
5.378.4.5 getloc	2027
5.378.4.6 gptr	2027
5.378.4.7 imbue	2028
5.378.4.8 in_avail	2028
5.378.4.9 overflow	2029
5.378.4.10 pbackfail	2029
5.378.4.11 pbase	2030
5.378.4.12 pbump	2030
5.378.4.13 pptr	2031
5.378.4.14 pubimbue	2031
5.378.4.15 pubseekoff	2031
5.378.4.16 pubseekpos	2032
5.378.4.17 pubsetbuf	2032
5.378.4.18 pubsync	2032
5.378.4.19 bumpc	2033
5.378.4.20 seekoff	2033
5.378.4.21 seekpos	2034
5.378.4.22 setbuf	2034
5.378.4.23 setg	2034
5.378.4.24 setp	2035
5.378.4.25 sgetc	2035
5.378.4.26 sgetn	2035
5.378.4.27 showmanyc	2036
5.378.4.28 nextc	2036
5.378.4.29 putbackc	2037

5.378.4.30	putc	2037
5.378.4.31	lputc	2038
5.378.4.32	stossc	2038
5.378.4.33	sungetc	2038
5.378.4.34	sync	2039
5.378.4.35	uflow	2039
5.378.4.36	underflow	2040
5.378.4.37	xsgetn	2040
5.378.4.38	xspn	2041
5.378.5	Member Data Documentation	2041
5.378.5.1	_M_buf_locale	2041
5.378.5.2	_M_in_beg	2042
5.378.5.3	_M_in_cur	2042
5.378.5.4	_M_in_end	2042
5.378.5.5	_M_out_beg	2043
5.378.5.6	_M_out_cur	2043
5.378.5.7	_M_out_end	2043
5.379	std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference	2044
5.379.1	Detailed Description	2049
5.379.2	Constructor & Destructor Documentation	2050
5.379.2.1	basic_string	2050
5.379.2.2	basic_string	2050
5.379.2.3	basic_string	2050
5.379.2.4	basic_string	2050
5.379.2.5	basic_string	2051
5.379.2.6	basic_string	2051
5.379.2.7	basic_string	2051
5.379.2.8	basic_string	2052
5.379.2.9	basic_string	2052
5.379.2.10	basic_string	2052
5.379.2.11	lbasic_string	2053

5.379.2.12~basic_string	2053
5.379.3 Member Function Documentation	2053
5.379.3.1 append	2053
5.379.3.2 append	2054
5.379.3.3 append	2054
5.379.3.4 append	2055
5.379.3.5 append	2055
5.379.3.6 append	2055
5.379.3.7 append	2056
5.379.3.8 assign	2056
5.379.3.9 assign	2057
5.379.3.10assign	2057
5.379.3.11assign	2058
5.379.3.12assign	2058
5.379.3.13assign	2059
5.379.3.14assign	2059
5.379.3.15assign	2059
5.379.3.16at	2060
5.379.3.17at	2060
5.379.3.18back	2061
5.379.3.19back	2061
5.379.3.20begin	2061
5.379.3.21begin	2061
5.379.3.22c_str	2062
5.379.3.23capacity	2062
5.379.3.24cbegin	2062
5.379.3.25end	2062
5.379.3.26clear	2063
5.379.3.27compare	2063
5.379.3.28compare	2063
5.379.3.29compare	2064

5.379.3.30compare	2065
5.379.3.31compare	2065
5.379.3.32compare	2066
5.379.3.33copy	2067
5.379.3.34crbegin	2067
5.379.3.35crend	2067
5.379.3.36data	2068
5.379.3.37empty	2068
5.379.3.38end	2068
5.379.3.39end	2068
5.379.3.40erase	2069
5.379.3.41erase	2069
5.379.3.42erase	2070
5.379.3.43find	2070
5.379.3.44find	2071
5.379.3.45find	2071
5.379.3.46find	2071
5.379.3.47find_first_not_of	2072
5.379.3.48find_first_not_of	2072
5.379.3.49find_first_not_of	2073
5.379.3.50find_first_not_of	2073
5.379.3.51find_first_of	2074
5.379.3.52find_first_of	2074
5.379.3.53find_first_of	2075
5.379.3.54find_first_of	2075
5.379.3.55find_last_not_of	2076
5.379.3.56find_last_not_of	2076
5.379.3.57find_last_not_of	2077
5.379.3.58find_last_not_of	2077
5.379.3.59find_last_of	2078
5.379.3.60find_last_of	2078

5.379.3.61find_last_of	2079
5.379.3.62find_last_of	2079
5.379.3.63front	2080
5.379.3.64front	2080
5.379.3.65get_allocator	2080
5.379.3.66insert	2080
5.379.3.67insert	2081
5.379.3.68insert	2081
5.379.3.69insert	2082
5.379.3.70insert	2082
5.379.3.71insert	2083
5.379.3.72insert	2083
5.379.3.73insert	2084
5.379.3.74insert	2085
5.379.3.75length	2085
5.379.3.76max_size	2085
5.379.3.77operator+=	2086
5.379.3.78operator+=	2086
5.379.3.79operator+=	2086
5.379.3.80operator+=	2087
5.379.3.81operator=	2087
5.379.3.82operator=	2087
5.379.3.83operator=	2088
5.379.3.84operator=	2088
5.379.3.85operator=	2088
5.379.3.86operator[]	2088
5.379.3.87operator[]	2089
5.379.3.88push_back	2089
5.379.3.89rbegin	2090
5.379.3.90rbegin	2090
5.379.3.91rend	2090

5.379.3.92	rend	2090
5.379.3.93	replace	2090
5.379.3.94	replace	2091
5.379.3.95	replace	2092
5.379.3.96	replace	2092
5.379.3.97	replace	2093
5.379.3.98	replace	2093
5.379.3.99	replace	2094
5.379.3.100	replace	2095
5.379.3.101	replace	2095
5.379.3.102	replace	2096
5.379.3.103	replace	2096
5.379.3.104	reserve	2097
5.379.3.105	resize	2098
5.379.3.106	resize	2098
5.379.3.107	find	2099
5.379.3.108	find	2099
5.379.3.109	find	2100
5.379.3.110	find	2100
5.379.3.111	shrink_to_fit	2101
5.379.3.112	size	2101
5.379.3.113	substr	2101
5.379.3.114	swap	2102
5.379.4	Member Data Documentation	2102
5.379.4.1	npos	2102
5.380	std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference	2102
5.380.1	Detailed Description	2106
5.380.2	Member Typedef Documentation	2106
5.380.2.1	__streambuf_type	2106
5.380.2.2	char_type	2106

5.380.2.3 int_type	2107
5.380.2.4 off_type	2107
5.380.2.5 pos_type	2107
5.380.2.6 traits_type	2107
5.380.3 Constructor & Destructor Documentation	2108
5.380.3.1 basic_stringbuf	2108
5.380.3.2 basic_stringbuf	2108
5.380.4 Member Function Documentation	2108
5.380.4.1 eback	2108
5.380.4.2 egptr	2109
5.380.4.3 epptr	2109
5.380.4.4 gbump	2110
5.380.4.5 getloc	2110
5.380.4.6 gptr	2110
5.380.4.7 imbue	2111
5.380.4.8 in_avail	2111
5.380.4.9 overflow	2111
5.380.4.10 pbackfail	2112
5.380.4.11 pbase	2113
5.380.4.12 pbump	2113
5.380.4.13 pptr	2113
5.380.4.14 pubimbue	2114
5.380.4.15 pubseekoff	2114
5.380.4.16 pubseekpos	2115
5.380.4.17 pubsetbuf	2115
5.380.4.18 pubsync	2115
5.380.4.19 sbumpc	2116
5.380.4.20 seekoff	2116
5.380.4.21 seekpos	2116
5.380.4.22 setbuf	2117
5.380.4.23 setbuf	2117

5.380.4.24	setg	2118
5.380.4.25	setp	2118
5.380.4.26	sgetc	2118
5.380.4.27	sgetn	2119
5.380.4.28	showmanyc	2119
5.380.4.29	nextc	2120
5.380.4.30	putbackc	2120
5.380.4.31	putc	2120
5.380.4.32	putn	2121
5.380.4.33	tossc	2121
5.380.4.34	str	2122
5.380.4.35	str	2122
5.380.4.36	ungetc	2122
5.380.4.37	sync	2123
5.380.4.38	uflow	2123
5.380.4.39	underflow	2124
5.380.4.40	xgetn	2124
5.380.4.41	xputn	2125
5.380.5	Member Data Documentation	2126
5.380.5.1	_M_buf_locale	2126
5.380.5.2	_M_in_beg	2126
5.380.5.3	_M_in_cur	2126
5.380.5.4	_M_in_end	2126
5.380.5.5	_M_mode	2127
5.380.5.6	_M_out_beg	2127
5.380.5.7	_M_out_cur	2127
5.380.5.8	_M_out_end	2128
5.381	std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference	2128
5.381.1	Detailed Description	2136
5.381.2	Member Typedef Documentation	2137

5.381.2.1	__ctype_type	2137
5.381.2.2	__ctype_type	2137
5.381.2.3	__num_get_type	2137
5.381.2.4	__num_put_type	2137
5.381.2.5	__num_put_type	2138
5.381.2.6	char_type	2138
5.381.2.7	char_type	2138
5.381.2.8	event_callback	2138
5.381.2.9	fmtflags	2139
5.381.2.10	int_type	2140
5.381.2.11	int_type	2140
5.381.2.12	iostate	2140
5.381.2.13	off_type	2141
5.381.2.14	off_type	2141
5.381.2.15	openmode	2141
5.381.2.16	pos_type	2142
5.381.2.17	pos_type	2142
5.381.2.18	seekdir	2142
5.381.2.19	traits_type	2142
5.381.2.20	traits_type	2143
5.381.3	Member Enumeration Documentation	2143
5.381.3.1	event	2143
5.381.4	Constructor & Destructor Documentation	2143
5.381.4.1	basic_stringstream	2143
5.381.4.2	basic_stringstream	2144
5.381.4.3	~basic_stringstream	2144
5.381.5	Member Function Documentation	2144
5.381.5.1	_M_getloc	2144
5.381.5.2	_M_write	2145
5.381.5.3	bad	2145
5.381.5.4	clear	2146

5.381.5.5 copyfmt	2146
5.381.5.6 eof	2146
5.381.5.7 exceptions	2147
5.381.5.8 exceptions	2147
5.381.5.9 fail	2148
5.381.5.10fill	2148
5.381.5.11fill	2149
5.381.5.12flags	2149
5.381.5.13flags	2149
5.381.5.14flush	2150
5.381.5.15gcount	2150
5.381.5.16get	2150
5.381.5.17get	2151
5.381.5.18get	2151
5.381.5.19get	2152
5.381.5.20get	2153
5.381.5.21get	2153
5.381.5.22getline	2154
5.381.5.23getline	2155
5.381.5.24getloc	2155
5.381.5.25good	2156
5.381.5.26ignore	2156
5.381.5.27ignore	2156
5.381.5.28ignore	2157
5.381.5.29imbue	2157
5.381.5.30init	2158
5.381.5.31iword	2158
5.381.5.32narrow	2159
5.381.5.33operator void *	2159
5.381.5.34operator!	2159
5.381.5.35operator<<	2160

5.381.5.36operator<<	2160
5.381.5.37operator<<	2160
5.381.5.38operator<<	2161
5.381.5.39operator<<	2161
5.381.5.40operator<<	2161
5.381.5.41operator<<	2162
5.381.5.42operator<<	2162
5.381.5.43operator<<	2162
5.381.5.44operator<<	2163
5.381.5.45operator<<	2163
5.381.5.46operator<<	2164
5.381.5.47operator<<	2164
5.381.5.48operator<<	2164
5.381.5.49operator<<	2165
5.381.5.50operator<<	2166
5.381.5.51operator<<	2166
5.381.5.52operator>>	2166
5.381.5.53operator>>	2167
5.381.5.54operator>>	2167
5.381.5.55operator>>	2168
5.381.5.56operator>>	2168
5.381.5.57operator>>	2168
5.381.5.58operator>>	2169
5.381.5.59operator>>	2169
5.381.5.60operator>>	2169
5.381.5.61operator>>	2170
5.381.5.62operator>>	2170
5.381.5.63operator>>	2171
5.381.5.64operator>>	2171
5.381.5.65operator>>	2172
5.381.5.66operator>>	2172

5.381.5.67operator>>	2172
5.381.5.68operator>>	2172
5.381.5.69peek	2173
5.381.5.70precision	2173
5.381.5.71precision	2174
5.381.5.72put	2174
5.381.5.73putback	2174
5.381.5.74pword	2175
5.381.5.75rdbuf	2176
5.381.5.76rdbuf	2176
5.381.5.77rdstate	2177
5.381.5.78read	2177
5.381.5.79readsome	2178
5.381.5.80register_callback	2178
5.381.5.81seekg	2179
5.381.5.82seekg	2179
5.381.5.83seekp	2180
5.381.5.84seekp	2180
5.381.5.85setf	2181
5.381.5.86setf	2181
5.381.5.87setstate	2182
5.381.5.88str	2182
5.381.5.89str	2182
5.381.5.90sync	2183
5.381.5.91sync_with_stdio	2183
5.381.5.92tellg	2184
5.381.5.93tellp	2184
5.381.5.94tie	2184
5.381.5.95tie	2185
5.381.5.96unget	2185
5.381.5.97unsetf	2186

5.381.5.98widen	2186
5.381.5.99width	2187
5.381.5.100width	2187
5.381.5.101write	2187
5.381.5.102alloc	2188
5.381.6 Member Data Documentation	2188
5.381.6.1 _M_gcount	2188
5.381.6.2 adjustfield	2188
5.381.6.3 app	2189
5.381.6.4 ate	2189
5.381.6.5 badbit	2189
5.381.6.6 basefield	2189
5.381.6.7 beg	2190
5.381.6.8 binary	2190
5.381.6.9 boolalpha	2190
5.381.6.10cur	2190
5.381.6.11dec	2190
5.381.6.12end	2190
5.381.6.13eofbit	2191
5.381.6.14failbit	2191
5.381.6.15fixed	2191
5.381.6.16floatfield	2191
5.381.6.17goodbit	2192
5.381.6.18hex	2192
5.381.6.19in	2192
5.381.6.20internal	2192
5.381.6.21left	2193
5.381.6.22oct	2193
5.381.6.23out	2193
5.381.6.24right	2193
5.381.6.25scientific	2193

5.381.6.26	showbase	2193
5.381.6.27	showpoint	2194
5.381.6.28	showpos	2194
5.381.6.29	skipws	2194
5.381.6.30	trunc	2194
5.381.6.31	unitbuf	2194
5.381.6.32	uppercase	2194
5.382	std::bernoulli_distribution Class Reference	2195
5.382.1	Detailed Description	2195
5.382.2	Member Typedef Documentation	2195
5.382.2.1	result_type	2195
5.382.3	Constructor & Destructor Documentation	2196
5.382.3.1	bernoulli_distribution	2196
5.382.4	Member Function Documentation	2196
5.382.4.1	max	2196
5.382.4.2	min	2196
5.382.4.3	operator()	2196
5.382.4.4	p	2196
5.382.4.5	param	2197
5.382.4.6	param	2197
5.382.4.7	reset	2197
5.383	std::bernoulli_distribution::param_type Struct Reference	2197
5.383.1	Detailed Description	2198
5.384	std::bidirectional_iterator_tag Struct Reference	2198
5.384.1	Detailed Description	2199
5.385	std::binary_function<_Arg1, _Arg2, _Result> Struct Template Reference	2199
5.385.1	Detailed Description	2200
5.385.2	Member Typedef Documentation	2200
5.385.2.1	first_argument_type	2200
5.385.2.2	result_type	2200

5.385.2.3 second_argument_type	2200
5.386std::binary_negate< _Predicate > Class Template Reference	2201
5.386.1 Detailed Description	2201
5.386.2 Member Typedef Documentation	2201
5.386.2.1 first_argument_type	2201
5.386.2.2 result_type	2202
5.386.2.3 second_argument_type	2202
5.387std::binder1st< _Operation > Class Template Reference	2202
5.387.1 Detailed Description	2203
5.387.2 Member Typedef Documentation	2203
5.387.2.1 argument_type	2203
5.387.2.2 result_type	2203
5.388std::binder2nd< _Operation > Class Template Reference	2204
5.388.1 Detailed Description	2205
5.388.2 Member Typedef Documentation	2205
5.388.2.1 argument_type	2205
5.388.2.2 result_type	2205
5.389std::binomial_distribution< _IntType > Class Template Reference	2205
5.389.1 Detailed Description	2206
5.389.2 Member Typedef Documentation	2207
5.389.2.1 result_type	2207
5.389.3 Member Function Documentation	2207
5.389.3.1 max	2207
5.389.3.2 min	2207
5.389.3.3 operator()	2207
5.389.3.4 operator()	2208
5.389.3.5 p	2208
5.389.3.6 param	2208
5.389.3.7 param	2208
5.389.3.8 reset	2209
5.389.3.9 t	2209

5.389.4 Friends And Related Function Documentation	2209
5.389.4.1 operator<<	2209
5.389.4.2 operator==	2209
5.389.4.3 operator>>	2210
5.390std::binomial_distribution< _IntType >::param_type Struct Reference	2210
5.390.1 Detailed Description	2211
5.391std::bitset< _Nb > Class Template Reference	2211
5.391.1 Detailed Description	2214
5.391.2 Constructor & Destructor Documentation	2215
5.391.2.1 bitset	2215
5.391.2.2 bitset	2216
5.391.2.3 bitset	2216
5.391.2.4 bitset	2216
5.391.2.5 bitset	2217
5.391.3 Member Function Documentation	2217
5.391.3.1 all	2217
5.391.3.2 any	2217
5.391.3.3 count	2217
5.391.3.4 flip	2218
5.391.3.5 flip	2218
5.391.3.6 none	2218
5.391.3.7 operator!=	2218
5.391.3.8 operator&=	2218
5.391.3.9 operator<<	2219
5.391.3.10operator<<=	2219
5.391.3.11operator==	2219
5.391.3.12operator>>	2219
5.391.3.13operator>>=	2219
5.391.3.14operator[]	2220
5.391.3.15operator[]	2220
5.391.3.16operator^=	2221

5.391.3.17operator =	2221
5.391.3.18operator~	2221
5.391.3.19reset	2222
5.391.3.20reset	2222
5.391.3.21set	2222
5.391.3.22set	2222
5.391.3.23size	2223
5.391.3.24test	2223
5.391.3.25to_string	2223
5.391.3.26to_ulong	2223
5.392std::bitset<_Nb>::reference Class Reference	2224
5.392.1 Detailed Description	2224
5.393std::cauchy_distribution<_RealType> Class Template Reference	2225
5.393.1 Detailed Description	2226
5.393.2 Member Typedef Documentation	2226
5.393.2.1 result_type	2226
5.393.3 Member Function Documentation	2226
5.393.3.1 max	2226
5.393.3.2 min	2226
5.393.3.3 operator()	2226
5.393.3.4 param	2227
5.393.3.5 param	2227
5.393.3.6 reset	2227
5.394std::cauchy_distribution<_RealType>::param_type Struct Reference	2227
5.394.1 Detailed Description	2228
5.395std::char_traits<_CharT> Struct Template Reference	2228
5.395.1 Detailed Description	2230
5.396std::char_traits<__gnu_cxx::character<V, I, S>> Struct Template Reference	2230
5.396.1 Detailed Description	2231
5.397std::char_traits<char> Struct Template Reference	2231

5.397.1 Detailed Description	2232
5.398std::char_traits< wchar_t > Struct Template Reference	2232
5.398.1 Detailed Description	2233
5.399std::chi_squared_distribution< _RealType > Class Template Reference	2233
5.399.1 Detailed Description	2234
5.399.2 Member Typedef Documentation	2234
5.399.2.1 result_type	2234
5.399.3 Member Function Documentation	2235
5.399.3.1 max	2235
5.399.3.2 min	2235
5.399.3.3 operator()	2235
5.399.3.4 param	2235
5.399.3.5 param	2235
5.399.3.6 reset	2236
5.399.4 Friends And Related Function Documentation	2236
5.399.4.1 operator<<	2236
5.399.4.2 operator==	2236
5.399.4.3 operator>>	2237
5.400std::chi_squared_distribution< _RealType >::param_type Struct Reference	2237
5.400.1 Detailed Description	2238
5.401std::chrono::duration< _Rep, _Period > Struct Template Reference	2238
5.401.1 Detailed Description	2239
5.402std::chrono::duration_values< _Rep > Struct Template Reference	2239
5.402.1 Detailed Description	2240
5.403std::chrono::system_clock Struct Reference	2240
5.403.1 Detailed Description	2241
5.404std::chrono::time_point< _Clock, _Duration > Struct Template Reference	2241
5.404.1 Detailed Description	2241
5.405std::chrono::treat_as_floating_point< _Rep > Struct Template Reference	2242
5.405.1 Detailed Description	2242

5.406std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference	2242
5.406.1 Detailed Description	2245
5.406.2 Member Function Documentation	2245
5.406.2.1 do_out	2245
5.406.2.2 in	2245
5.406.2.3 out	2246
5.406.2.4 unshift	2247
5.407std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference	2248
5.407.1 Detailed Description	2250
5.407.2 Member Function Documentation	2251
5.407.2.1 do_out	2251
5.407.2.2 in	2251
5.407.2.3 out	2252
5.407.2.4 unshift	2253
5.408std::codecvt< char, char, mbstate_t > Class Template Reference	2254
5.408.1 Detailed Description	2256
5.408.2 Member Function Documentation	2256
5.408.2.1 do_out	2256
5.408.2.2 in	2256
5.408.2.3 out	2257
5.408.2.4 unshift	2258
5.409std::codecvt< wchar_t, char, mbstate_t > Class Template Reference	2259
5.409.1 Detailed Description	2261
5.409.2 Member Function Documentation	2261
5.409.2.1 do_out	2261
5.409.2.2 in	2262
5.409.2.3 out	2263
5.409.2.4 unshift	2264
5.410std::codecvt_base Class Reference	2264
5.410.1 Detailed Description	2265

5.411	std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference	2265
5.411.1	Detailed Description	2268
5.411.2	Member Function Documentation	2268
5.411.2.1	do_out	2268
5.411.2.2	in	2268
5.411.2.3	out	2269
5.411.2.4	unshift	2270
5.412	std::collate< _CharT > Class Template Reference	2271
5.412.1	Detailed Description	2273
5.412.2	Member Typedef Documentation	2274
5.412.2.1	char_type	2274
5.412.2.2	string_type	2274
5.412.3	Constructor & Destructor Documentation	2274
5.412.3.1	collate	2274
5.412.3.2	collate	2274
5.412.3.3	~collate	2275
5.412.4	Member Function Documentation	2275
5.412.4.1	compare	2275
5.412.4.2	do_compare	2275
5.412.4.3	do_hash	2276
5.412.4.4	do_transform	2276
5.412.4.5	hash	2277
5.412.4.6	transform	2277
5.412.5	Member Data Documentation	2278
5.412.5.1	id	2278
5.413	std::collate_byname< _CharT > Class Template Reference	2278
5.413.1	Detailed Description	2280
5.413.2	Member Typedef Documentation	2281
5.413.2.1	char_type	2281
5.413.2.2	string_type	2281

5.413.3 Member Function Documentation	2281
5.413.3.1 compare	2281
5.413.3.2 do_compare	2282
5.413.3.3 do_hash	2282
5.413.3.4 do_transform	2283
5.413.3.5 hash	2283
5.413.3.6 transform	2284
5.413.4 Member Data Documentation	2284
5.413.4.1 id	2284
5.414std::complex<_Tp> Struct Template Reference	2284
5.414.1 Detailed Description	2285
5.414.2 Member Typedef Documentation	2286
5.414.2.1 value_type	2286
5.414.3 Constructor & Destructor Documentation	2286
5.414.3.1 complex	2286
5.414.3.2 complex	2286
5.414.4 Member Function Documentation	2286
5.414.4.1 operator+=	2286
5.414.4.2 operator-=	2286
5.415std::condition_variable Class Reference	2287
5.415.1 Detailed Description	2287
5.416std::condition_variable_any Class Reference	2288
5.416.1 Detailed Description	2288
5.417std::conditional<_Cond, _Iftrue, _Iffalse> Struct Template Reference	2289
5.417.1 Detailed Description	2289
5.418std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg> Class Template Reference	2289
5.418.1 Detailed Description	2290
5.418.2 Member Typedef Documentation	2291
5.418.2.1 first_argument_type	2291
5.418.2.2 result_type	2291

5.418.2.3 second_argument_type	2291
5.419std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2291
5.419.1 Detailed Description	2292
5.419.2 Member Typedef Documentation	2293
5.419.2.1 first_argument_type	2293
5.419.2.2 result_type	2293
5.419.2.3 second_argument_type	2293
5.420std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference .	2293
5.420.1 Detailed Description	2294
5.420.2 Member Typedef Documentation	2295
5.420.2.1 argument_type	2295
5.420.2.2 result_type	2295
5.421std::const_mem_fun_t< _Ret, _Tp > Class Template Reference . . .	2295
5.421.1 Detailed Description	2296
5.421.2 Member Typedef Documentation	2297
5.421.2.1 argument_type	2297
5.421.2.2 result_type	2297
5.422std::ctype< _CharT > Class Template Reference	2297
5.422.1 Detailed Description	2300
5.422.2 Member Typedef Documentation	2300
5.422.2.1 char_type	2300
5.422.3 Member Function Documentation	2301
5.422.3.1 do_is	2301
5.422.3.2 do_is	2301
5.422.3.3 do_narrow	2302
5.422.3.4 do_narrow	2302
5.422.3.5 do_scan_is	2303
5.422.3.6 do_scan_not	2303
5.422.3.7 do_tolower	2304
5.422.3.8 do_tolower	2304
5.422.3.9 do_toupper	2305

5.422.3.10do_toupper	2305
5.422.3.11do_widen	2306
5.422.3.12do_widen	2306
5.422.3.13is	2307
5.422.3.14is	2307
5.422.3.15narrow	2308
5.422.3.16narrow	2308
5.422.3.17scan_is	2309
5.422.3.18scan_not	2309
5.422.3.19tolower	2310
5.422.3.20tolower	2310
5.422.3.21toupper	2310
5.422.3.22toupper	2311
5.422.3.23widen	2311
5.422.3.24widen	2312
5.422.4 Member Data Documentation	2312
5.422.4.1 id	2312
5.423std::ctype< char > Class Template Reference	2313
5.423.1 Detailed Description	2315
5.423.2 Member Typedef Documentation	2316
5.423.2.1 char_type	2316
5.423.3 Constructor & Destructor Documentation	2316
5.423.3.1 ctype	2316
5.423.3.2 ctype	2316
5.423.3.3 ~ctype	2316
5.423.4 Member Function Documentation	2317
5.423.4.1 classic_table	2317
5.423.4.2 do_narrow	2317
5.423.4.3 do_narrow	2317
5.423.4.4 do_tolower	2318
5.423.4.5 do_tolower	2318

5.423.4.6 do_toupper	2319
5.423.4.7 do_toupper	2319
5.423.4.8 do_widen	2320
5.423.4.9 do_widen	2320
5.423.4.10 is	2321
5.423.4.11 is	2321
5.423.4.12 narrow	2321
5.423.4.13 narrow	2322
5.423.4.14 scan_is	2323
5.423.4.15 scan_not	2323
5.423.4.16 table	2323
5.423.4.17 tolower	2324
5.423.4.18 tolower	2324
5.423.4.19 toupper	2324
5.423.4.20 toupper	2325
5.423.4.21 widen	2325
5.423.4.22 widen	2326
5.423.5 Member Data Documentation	2326
5.423.5.1 id	2326
5.423.5.2 table_size	2326
5.424 std::ctype< wchar_t > Class Template Reference	2327
5.424.1 Detailed Description	2330
5.424.2 Member Typedef Documentation	2330
5.424.2.1 char_type	2330
5.424.3 Constructor & Destructor Documentation	2331
5.424.3.1 ctype	2331
5.424.3.2 ctype	2331
5.424.3.3 ~ctype	2331
5.424.4 Member Function Documentation	2331
5.424.4.1 do_is	2331
5.424.4.2 do_is	2332

5.424.4.3 do_is	2332
5.424.4.4 do_is	2333
5.424.4.5 do_narrow	2333
5.424.4.6 do_narrow	2334
5.424.4.7 do_narrow	2334
5.424.4.8 do_narrow	2335
5.424.4.9 do_scan_is	2335
5.424.4.10do_scan_is	2336
5.424.4.11do_scan_not	2336
5.424.4.12do_scan_not	2337
5.424.4.13do_tolower	2337
5.424.4.14do_tolower	2337
5.424.4.15do_tolower	2338
5.424.4.16do_tolower	2338
5.424.4.17do_toupper	2339
5.424.4.18do_toupper	2339
5.424.4.19do_toupper	2340
5.424.4.20do_toupper	2340
5.424.4.21do_widen	2340
5.424.4.22do_widen	2341
5.424.4.23do_widen	2341
5.424.4.24is	2342
5.424.4.25is	2342
5.424.4.26narrow	2343
5.424.4.27narrow	2343
5.424.4.28scan_is	2344
5.424.4.29scan_not	2344
5.424.4.30tolower	2345
5.424.4.31tolower	2345
5.424.4.32toupper	2346
5.424.4.33toupper	2346

5.424.4.34	widen	2346
5.424.4.35	widen	2347
5.424.5	Member Data Documentation	2347
5.424.5.1	id	2347
5.425	std::ctype_base Struct Reference	2348
5.425.1	Detailed Description	2348
5.426	std::ctype_byname< _CharT > Class Template Reference	2349
5.426.1	Detailed Description	2351
5.426.2	Member Typedef Documentation	2351
5.426.2.1	char_type	2351
5.426.3	Member Function Documentation	2352
5.426.3.1	do_is	2352
5.426.3.2	do_is	2352
5.426.3.3	do_narrow	2353
5.426.3.4	do_narrow	2353
5.426.3.5	do_scan_is	2354
5.426.3.6	do_scan_not	2354
5.426.3.7	do_tolower	2355
5.426.3.8	do_tolower	2355
5.426.3.9	do_toupper	2356
5.426.3.10	do_toupper	2356
5.426.3.11	do_widen	2357
5.426.3.12	do_widen	2357
5.426.3.13	is	2358
5.426.3.14	is	2358
5.426.3.15	narrow	2359
5.426.3.16	narrow	2359
5.426.3.17	scan_is	2360
5.426.3.18	scan_not	2360
5.426.3.19	tolower	2361
5.426.3.20	tolower	2361

5.426.3.2ltoupper	2361
5.426.3.2ltoupper	2362
5.426.3.23widen	2362
5.426.3.24widen	2363
5.426.4 Member Data Documentation	2363
5.426.4.1 id	2363
5.427std::ctype_byname< char > Class Template Reference	2364
5.427.1 Detailed Description	2366
5.427.2 Member Typedef Documentation	2366
5.427.2.1 char_type	2366
5.427.3 Member Function Documentation	2367
5.427.3.1 classic_table	2367
5.427.3.2 do_narrow	2367
5.427.3.3 do_narrow	2367
5.427.3.4 do_tolower	2368
5.427.3.5 do_tolower	2368
5.427.3.6 do_toupper	2369
5.427.3.7 do_toupper	2369
5.427.3.8 do_widen	2370
5.427.3.9 do_widen	2370
5.427.3.10is	2371
5.427.3.11is	2371
5.427.3.12narrow	2371
5.427.3.13narrow	2372
5.427.3.14scan_is	2373
5.427.3.15scan_not	2373
5.427.3.16table	2373
5.427.3.17tolower	2374
5.427.3.18tolower	2374
5.427.3.19toupper	2374
5.427.3.20toupper	2375

5.427.3.2l	widen	2375
5.427.3.2w	widen	2376
5.427.4	Member Data Documentation	2376
5.427.4.1	id	2376
5.427.4.2	table_size	2376
5.428	std::decay< _Tp > Class Template Reference	2377
5.428.1	Detailed Description	2377
5.429	std::decimal::decimal128 Class Reference	2377
5.429.1	Detailed Description	2379
5.429.2	Constructor & Destructor Documentation	2379
5.429.2.1	decimal128	2379
5.430	std::decimal::decimal32 Class Reference	2379
5.430.1	Detailed Description	2381
5.430.2	Constructor & Destructor Documentation	2381
5.430.2.1	decimal32	2381
5.431	std::decimal::decimal64 Class Reference	2381
5.431.1	Detailed Description	2383
5.431.2	Constructor & Destructor Documentation	2383
5.431.2.1	decimal64	2383
5.432	std::default_delete< _Tp > Struct Template Reference	2383
5.432.1	Detailed Description	2384
5.433	std::default_delete< _Tp[]> Struct Template Reference	2384
5.433.1	Detailed Description	2384
5.434	std::defer_lock_t Struct Reference	2384
5.434.1	Detailed Description	2384
5.435	std::deque< _Tp, _Alloc > Class Template Reference	2385
5.435.1	Detailed Description	2390
5.435.2	Constructor & Destructor Documentation	2391
5.435.2.1	deque	2391
5.435.2.2	deque	2391
5.435.2.3	deque	2392

5.435.2.4 deque	2392
5.435.2.5 deque	2392
5.435.2.6 deque	2393
5.435.2.7 deque	2393
5.435.2.8 deque	2394
5.435.2.9 ~deque	2394
5.435.3 Member Function Documentation	2394
5.435.3.1 _M_fill_initialize	2394
5.435.3.2 _M_initialize_map	2395
5.435.3.3 _M_new_elements_at_back	2395
5.435.3.4 _M_new_elements_at_front	2396
5.435.3.5 _M_pop_back_aux	2396
5.435.3.6 _M_pop_front_aux	2396
5.435.3.7 _M_push_back_aux	2396
5.435.3.8 _M_push_front_aux	2396
5.435.3.9 _M_range_check	2397
5.435.3.10 _M_range_initialize	2397
5.435.3.11 _M_range_initialize	2397
5.435.3.12 _M_reallocate_map	2398
5.435.3.13 _M_reserve_elements_at_back	2398
5.435.3.14 _M_reserve_elements_at_front	2398
5.435.3.15 _M_reserve_map_at_back	2399
5.435.3.16 _M_reserve_map_at_front	2399
5.435.3.17 assign	2399
5.435.3.18 assign	2400
5.435.3.19 assign	2400
5.435.3.20 at	2400
5.435.3.21 at	2401
5.435.3.22 back	2401
5.435.3.23 back	2402
5.435.3.24 begin	2402

5.435.3.25	begin	2402
5.435.3.26	begin	2402
5.435.3.27	end	2403
5.435.3.28	clear	2403
5.435.3.29	begin	2403
5.435.3.30	end	2403
5.435.3.31	emplace	2403
5.435.3.32	empty	2404
5.435.3.33	end	2404
5.435.3.34	end	2404
5.435.3.35	erase	2405
5.435.3.36	erase	2405
5.435.3.37	front	2406
5.435.3.38	front	2406
5.435.3.39	get_allocator	2406
5.435.3.40	insert	2406
5.435.3.41	insert	2407
5.435.3.42	insert	2407
5.435.3.43	insert	2407
5.435.3.44	insert	2408
5.435.3.45	max_size	2408
5.435.3.46	operator=	2408
5.435.3.47	operator=	2409
5.435.3.48	operator=	2409
5.435.3.49	operator[]	2410
5.435.3.50	operator[]	2410
5.435.3.51	pop_back	2410
5.435.3.52	pop_front	2411
5.435.3.53	push_back	2411
5.435.3.54	push_front	2411
5.435.3.55	begin	2412

5.435.3.56	<code>begin</code>	2412
5.435.3.57	<code>rend</code>	2412
5.435.3.58	<code>rend</code>	2412
5.435.3.59	<code>resize</code>	2413
5.435.3.60	<code>resize</code>	2413
5.435.3.61	<code>shrink_to_fit</code>	2413
5.435.3.62	<code>size</code>	2414
5.435.3.63	<code>swap</code>	2414
5.436	<code>std::discard_block_engine< _RandomNumberEngine, __p, __r ></code>	
	Class Template Reference	2414
5.436.1	Detailed Description	2416
5.436.2	Member Typedef Documentation	2416
5.436.2.1	<code>result_type</code>	2416
5.436.3	Constructor & Destructor Documentation	2416
5.436.3.1	<code>discard_block_engine</code>	2416
5.436.3.2	<code>discard_block_engine</code>	2416
5.436.3.3	<code>discard_block_engine</code>	2417
5.436.3.4	<code>discard_block_engine</code>	2417
5.436.3.5	<code>discard_block_engine</code>	2417
5.436.4	Member Function Documentation	2418
5.436.4.1	<code>base</code>	2418
5.436.4.2	<code>discard</code>	2418
5.436.4.3	<code>max</code>	2418
5.436.4.4	<code>min</code>	2418
5.436.4.5	<code>operator()</code>	2419
5.436.4.6	<code>seed</code>	2419
5.436.4.7	<code>seed</code>	2419
5.436.4.8	<code>seed</code>	2419
5.436.5	Friends And Related Function Documentation	2420
5.436.5.1	<code>operator<<</code>	2420
5.436.5.2	<code>operator==</code>	2420

5.436.5.3 operator>>	2421
5.437std::discrete_distribution< _IntType > Class Template Reference	2421
5.437.1 Detailed Description	2422
5.437.2 Member Typedef Documentation	2422
5.437.2.1 result_type	2422
5.437.3 Member Function Documentation	2423
5.437.3.1 max	2423
5.437.3.2 min	2423
5.437.3.3 operator()	2423
5.437.3.4 param	2423
5.437.3.5 param	2424
5.437.3.6 probabilities	2424
5.437.3.7 reset	2424
5.437.4 Friends And Related Function Documentation	2424
5.437.4.1 operator<<	2424
5.437.4.2 operator>>	2425
5.438std::discrete_distribution< _IntType >::param_type Struct Reference	2425
5.438.1 Detailed Description	2426
5.439std::divides< _Tp > Struct Template Reference	2426
5.439.1 Detailed Description	2427
5.439.2 Member Typedef Documentation	2427
5.439.2.1 first_argument_type	2427
5.439.2.2 result_type	2427
5.439.2.3 second_argument_type	2427
5.440std::domain_error Class Reference	2428
5.440.1 Detailed Description	2428
5.440.2 Member Function Documentation	2428
5.440.2.1 what	2428
5.441std::enable_if< bool, _Tp > Struct Template Reference	2429
5.441.1 Detailed Description	2429
5.442std::enable_shared_from_this< _Tp > Class Template Reference	2429

5.442.1 Detailed Description	2430
5.443std::equal_to< _Tp > Struct Template Reference	2430
5.443.1 Detailed Description	2431
5.443.2 Member Typedef Documentation	2432
5.443.2.1 first_argument_type	2432
5.443.2.2 result_type	2432
5.443.2.3 second_argument_type	2432
5.444std::error_category Class Reference	2432
5.444.1 Detailed Description	2433
5.445std::error_code Struct Reference	2433
5.445.1 Detailed Description	2433
5.446std::error_condition Struct Reference	2434
5.446.1 Detailed Description	2434
5.447std::exception Class Reference	2434
5.447.1 Detailed Description	2436
5.447.2 Member Function Documentation	2436
5.447.2.1 what	2436
5.448std::exponential_distribution< _RealType > Class Template Reference	2436
5.448.1 Detailed Description	2437
5.448.2 Member Typedef Documentation	2437
5.448.2.1 result_type	2437
5.448.3 Constructor & Destructor Documentation	2438
5.448.3.1 exponential_distribution	2438
5.448.4 Member Function Documentation	2438
5.448.4.1 lambda	2438
5.448.4.2 max	2438
5.448.4.3 min	2438
5.448.4.4 operator()	2438
5.448.4.5 param	2439
5.448.4.6 param	2439
5.448.4.7 reset	2439

5.449	std::exponential_distribution< _RealType >::param_type Struct Reference	2440
5.449.1	Detailed Description	2440
5.450	std::extreme_value_distribution< _RealType > Class Template Reference	2440
5.450.1	Detailed Description	2441
5.450.2	Member Typedef Documentation	2441
5.450.2.1	result_type	2441
5.450.3	Member Function Documentation	2442
5.450.3.1	a	2442
5.450.3.2	b	2442
5.450.3.3	max	2442
5.450.3.4	min	2442
5.450.3.5	operator()	2442
5.450.3.6	param	2443
5.450.3.7	param	2443
5.450.3.8	reset	2443
5.451	std::extreme_value_distribution< _RealType >::param_type Struct Reference	2443
5.451.1	Detailed Description	2444
5.452	std::fisher_f_distribution< _RealType > Class Template Reference	2444
5.452.1	Detailed Description	2445
5.452.2	Member Typedef Documentation	2445
5.452.2.1	result_type	2445
5.452.3	Member Function Documentation	2446
5.452.3.1	max	2446
5.452.3.2	min	2446
5.452.3.3	operator()	2446
5.452.3.4	param	2446
5.452.3.5	param	2446
5.452.3.6	reset	2447
5.452.4	Friends And Related Function Documentation	2447

5.452.4.1 operator<<	2447
5.452.4.2 operator==	2447
5.452.4.3 operator>>	2448
5.453std::fisher_f_distribution<_RealType >::param_type Struct Reference	2448
5.453.1 Detailed Description	2449
5.454std::forward_iterator_tag Struct Reference	2449
5.454.1 Detailed Description	2450
5.455std::forward_list<_Tp, _Alloc > Class Template Reference	2450
5.455.1 Detailed Description	2453
5.455.2 Constructor & Destructor Documentation	2454
5.455.2.1 forward_list	2454
5.455.2.2 forward_list	2454
5.455.2.3 forward_list	2454
5.455.2.4 forward_list	2454
5.455.2.5 forward_list	2455
5.455.2.6 forward_list	2455
5.455.2.7 forward_list	2456
5.455.2.8 forward_list	2456
5.455.2.9 forward_list	2456
5.455.2.10~forward_list	2457
5.455.3 Member Function Documentation	2457
5.455.3.1 assign	2457
5.455.3.2 assign	2457
5.455.3.3 assign	2458
5.455.3.4 before_begin	2458
5.455.3.5 before_begin	2458
5.455.3.6 begin	2458
5.455.3.7 begin	2459
5.455.3.8 cbefore_begin	2459
5.455.3.9 cbegin	2459
5.455.3.10end	2459

5.455.3.1	clear	2459
5.455.3.12	emplace_after	2460
5.455.3.13	emplace_front	2460
5.455.3.14	empty	2461
5.455.3.15	end	2461
5.455.3.16	end	2461
5.455.3.17	erase_after	2461
5.455.3.18	erase_after	2462
5.455.3.19	front	2462
5.455.3.20	front	2462
5.455.3.21	get_allocator	2463
5.455.3.22	insert_after	2463
5.455.3.23	insert_after	2463
5.455.3.24	insert_after	2464
5.455.3.25	insert_after	2464
5.455.3.26	max_size	2465
5.455.3.27	merge	2465
5.455.3.28	merge	2465
5.455.3.29	operator=	2466
5.455.3.30	operator=	2466
5.455.3.31	operator=	2466
5.455.3.32	pop_front	2467
5.455.3.33	push_front	2467
5.455.3.34	remove	2467
5.455.3.35	remove_if	2468
5.455.3.36	resize	2468
5.455.3.37	resize	2468
5.455.3.38	reverse	2469
5.455.3.39	sort	2469
5.455.3.40	sort	2469
5.455.3.41	splice_after	2470

5.455.3.42splice_after	2470
5.455.3.43splice_after	2470
5.455.3.44swap	2471
5.455.3.45unique	2471
5.455.3.46unique	2471
5.456std::fpos< _StateT > Class Template Reference	2472
5.456.1 Detailed Description	2472
5.456.2 Constructor & Destructor Documentation	2473
5.456.2.1 fpos	2473
5.456.3 Member Function Documentation	2473
5.456.3.1 operator streamoff	2473
5.456.3.2 operator+	2473
5.456.3.3 operator+=	2473
5.456.3.4 operator-	2473
5.456.3.5 operator-	2474
5.456.3.6 operator-=	2474
5.456.3.7 state	2474
5.456.3.8 state	2474
5.457std::front_insert_iterator< _Container > Class Template Reference	2474
5.457.1 Detailed Description	2476
5.457.2 Member Typedef Documentation	2476
5.457.2.1 container_type	2476
5.457.2.2 difference_type	2476
5.457.2.3 iterator_category	2476
5.457.2.4 pointer	2476
5.457.2.5 reference	2477
5.457.2.6 value_type	2477
5.457.3 Constructor & Destructor Documentation	2477
5.457.3.1 front_insert_iterator	2477
5.457.4 Member Function Documentation	2477
5.457.4.1 operator*	2477

5.457.4.2 operator++	2477
5.457.4.3 operator++	2478
5.457.4.4 operator=	2478
5.458std::function< _Res(_ArgTypes...)> Class Template Reference	2478
5.458.1 Detailed Description	2480
5.458.2 Constructor & Destructor Documentation	2481
5.458.2.1 function	2481
5.458.2.2 function	2481
5.458.2.3 function	2481
5.458.2.4 function	2481
5.458.2.5 function	2482
5.458.3 Member Function Documentation	2482
5.458.3.1 operator bool	2482
5.458.3.2 operator()	2483
5.458.3.3 operator=	2483
5.458.3.4 operator=	2483
5.458.3.5 operator=	2484
5.458.3.6 operator=	2484
5.458.3.7 operator=	2485
5.458.3.8 swap	2485
5.458.3.9 target	2485
5.458.3.10target	2486
5.458.3.11target_type	2486
5.459std::future< _Res > Class Template Reference	2486
5.459.1 Detailed Description	2488
5.459.2 Constructor & Destructor Documentation	2488
5.459.2.1 future	2488
5.459.3 Member Function Documentation	2488
5.459.3.1 _M_get_result	2488
5.459.3.2 get	2489
5.460std::future< _Res & > Class Template Reference	2489

5.460.1 Detailed Description	2491
5.460.2 Constructor & Destructor Documentation	2491
5.460.2.1 future	2491
5.460.3 Member Function Documentation	2492
5.460.3.1 _M_get_result	2492
5.460.3.2 get	2492
5.461 std::future< void > Class Template Reference	2492
5.461.1 Detailed Description	2494
5.461.2 Constructor & Destructor Documentation	2494
5.461.2.1 future	2494
5.461.3 Member Function Documentation	2495
5.461.3.1 _M_get_result	2495
5.461.3.2 get	2495
5.462 std::future_error Class Reference	2495
5.462.1 Detailed Description	2496
5.462.2 Member Function Documentation	2496
5.462.2.1 what	2496
5.463 std::gamma_distribution< _RealType > Class Template Reference	2496
5.463.1 Detailed Description	2498
5.463.2 Member Typedef Documentation	2498
5.463.2.1 result_type	2498
5.463.3 Constructor & Destructor Documentation	2498
5.463.3.1 gamma_distribution	2498
5.463.4 Member Function Documentation	2499
5.463.4.1 alpha	2499
5.463.4.2 beta	2499
5.463.4.3 max	2499
5.463.4.4 min	2499
5.463.4.5 operator()	2499
5.463.4.6 operator()	2500
5.463.4.7 param	2500

5.463.4.8 param	2500
5.463.4.9 reset	2500
5.463.5 Friends And Related Function Documentation	2501
5.463.5.1 operator<<	2501
5.463.5.2 operator==	2501
5.463.5.3 operator>>	2501
5.464std::gamma_distribution< _RealType >::param_type Struct Reference	2502
5.464.1 Detailed Description	2502
5.465std::geometric_distribution< _IntType > Class Template Reference .	2503
5.465.1 Detailed Description	2503
5.465.2 Member Typedef Documentation	2504
5.465.2.1 result_type	2504
5.465.3 Member Function Documentation	2504
5.465.3.1 max	2504
5.465.3.2 min	2504
5.465.3.3 operator()	2504
5.465.3.4 p	2504
5.465.3.5 param	2505
5.465.3.6 param	2505
5.465.3.7 reset	2505
5.466std::geometric_distribution< _IntType >::param_type Struct Reference	2505
5.466.1 Detailed Description	2506
5.467std::greater< _Tp > Struct Template Reference	2506
5.467.1 Detailed Description	2507
5.467.2 Member Typedef Documentation	2508
5.467.2.1 first_argument_type	2508
5.467.2.2 result_type	2508
5.467.2.3 second_argument_type	2508
5.468std::greater_equal< _Tp > Struct Template Reference	2508
5.468.1 Detailed Description	2509
5.468.2 Member Typedef Documentation	2510

5.468.2.1 first_argument_type	2510
5.468.2.2 result_type	2510
5.468.2.3 second_argument_type	2510
5.469std::gslice Class Reference	2510
5.469.1 Detailed Description	2511
5.470std::gslice_array< _Tp > Class Template Reference	2511
5.470.1 Detailed Description	2512
5.471std::has_nothrow_copy_assign< _Tp > Struct Template Reference	2513
5.471.1 Detailed Description	2513
5.472std::has_nothrow_copy_constructor< _Tp > Struct Template Reference	2513
5.472.1 Detailed Description	2513
5.473std::has_nothrow_default_constructor< _Tp > Struct Template Reference	2514
5.473.1 Detailed Description	2514
5.474std::has_trivial_copy_assign< _Tp > Struct Template Reference	2514
5.474.1 Detailed Description	2514
5.475std::has_trivial_copy_constructor< _Tp > Struct Template Reference	2515
5.475.1 Detailed Description	2515
5.476std::has_trivial_default_constructor< _Tp > Struct Template Reference	2515
5.476.1 Detailed Description	2515
5.477std::has_trivial_destructor< _Tp > Struct Template Reference	2515
5.477.1 Detailed Description	2516
5.478std::hash< _Tp > Struct Template Reference	2516
5.478.1 Detailed Description	2517
5.479std::hash< __debug::bitset< _Nb > > Struct Template Reference	2517
5.479.1 Detailed Description	2518
5.480std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference	2518
5.480.1 Detailed Description	2518
5.481std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference	2519
5.481.1 Detailed Description	2519
5.481.2 Member Typedef Documentation	2520

5.481.2.1 argument_type	2520
5.481.2.2 result_type	2520
5.482std::hash< __gnu_cxx::throw_value_random > Struct Template Reference	2520
5.482.1 Detailed Description	2521
5.482.2 Member Typedef Documentation	2522
5.482.2.1 argument_type	2522
5.482.2.2 result_type	2522
5.483std::hash< __profile::bitset< _Nb > > Struct Template Reference	2522
5.483.1 Detailed Description	2522
5.484std::hash< __profile::vector< bool, _Alloc > > Struct Template Reference	2523
5.484.1 Detailed Description	2523
5.485std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference	2523
5.485.1 Detailed Description	2524
5.485.2 Member Typedef Documentation	2525
5.485.2.1 argument_type	2525
5.485.2.2 result_type	2525
5.486std::hash< _Tp * > Struct Template Reference	2525
5.486.1 Detailed Description	2525
5.487std::hash< error_code > Struct Template Reference	2526
5.487.1 Detailed Description	2526
5.488std::hash< shared_ptr< _Tp > > Struct Template Reference	2526
5.488.1 Detailed Description	2527
5.488.2 Member Typedef Documentation	2528
5.488.2.1 argument_type	2528
5.488.2.2 result_type	2528
5.489std::hash< string > Struct Template Reference	2528
5.489.1 Detailed Description	2528
5.490std::hash< thread::id > Struct Template Reference	2529
5.490.1 Detailed Description	2529
5.491std::hash< u16string > Struct Template Reference	2529

5.491.1 Detailed Description	2530
5.492std::hash< u32string > Struct Template Reference	2530
5.492.1 Detailed Description	2530
5.493std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference	2531
5.493.1 Detailed Description	2531
5.493.2 Member Typedef Documentation	2532
5.493.2.1 argument_type	2532
5.493.2.2 result_type	2532
5.494std::hash< wstring > Struct Template Reference	2532
5.494.1 Detailed Description	2532
5.495std::hash<::bitset< _Nb > > Struct Template Reference	2533
5.495.1 Detailed Description	2533
5.496std::hash<::vector< bool, _Alloc > > Struct Template Reference	2533
5.496.1 Detailed Description	2534
5.497std::independent_bits_engine< _RandomNumberEngine, __w, _- UIntType > Class Template Reference	2534
5.497.1 Detailed Description	2535
5.497.2 Member Typedef Documentation	2535
5.497.2.1 result_type	2535
5.497.3 Constructor & Destructor Documentation	2536
5.497.3.1 independent_bits_engine	2536
5.497.3.2 independent_bits_engine	2536
5.497.3.3 independent_bits_engine	2536
5.497.3.4 independent_bits_engine	2537
5.497.3.5 independent_bits_engine	2537
5.497.4 Member Function Documentation	2537
5.497.4.1 base	2537
5.497.4.2 discard	2538
5.497.4.3 max	2538
5.497.4.4 min	2538
5.497.4.5 operator()	2538

5.497.4.6 seed	2539
5.497.4.7 seed	2539
5.497.4.8 seed	2539
5.497.5 Friends And Related Function Documentation	2540
5.497.5.1 operator==	2540
5.497.5.2 operator>>	2540
5.498std::indirect_array< _Tp > Class Template Reference	2541
5.498.1 Detailed Description	2542
5.498.2 Member Function Documentation	2542
5.498.2.1 operator%=	2542
5.498.2.2 operator&=	2542
5.498.2.3 operator*=	2543
5.498.2.4 operator+=	2543
5.498.2.5 operator-=	2543
5.498.2.6 operator/=	2543
5.498.2.7 operator<<=	2543
5.498.2.8 operator>>=	2543
5.498.2.9 operator^=	2543
5.498.2.10operator =	2543
5.499std::initializer_list< _E > Class Template Reference	2544
5.499.1 Detailed Description	2544
5.500std::input_iterator_tag Struct Reference	2544
5.500.1 Detailed Description	2545
5.501std::insert_iterator< _Container > Class Template Reference	2545
5.501.1 Detailed Description	2547
5.501.2 Member Typedef Documentation	2547
5.501.2.1 container_type	2547
5.501.2.2 difference_type	2547
5.501.2.3 iterator_category	2547
5.501.2.4 pointer	2547
5.501.2.5 reference	2548

5.501.2.6 value_type	2548
5.501.3 Constructor & Destructor Documentation	2548
5.501.3.1 insert_iterator	2548
5.501.4 Member Function Documentation	2548
5.501.4.1 operator*	2548
5.501.4.2 operator++	2548
5.501.4.3 operator++	2549
5.501.4.4 operator=	2549
5.502std::invalid_argument Class Reference	2550
5.502.1 Detailed Description	2550
5.502.2 Member Function Documentation	2550
5.502.2.1 what	2550
5.503std::ios_base Class Reference	2551
5.503.1 Detailed Description	2554
5.503.2 Member Typedef Documentation	2554
5.503.2.1 event_callback	2554
5.503.2.2 fmtflags	2555
5.503.2.3 iostate	2556
5.503.2.4 openmode	2556
5.503.2.5 seekdir	2556
5.503.3 Member Enumeration Documentation	2557
5.503.3.1 event	2557
5.503.4 Constructor & Destructor Documentation	2557
5.503.4.1 ~ios_base	2557
5.503.5 Member Function Documentation	2557
5.503.5.1 _M_getloc	2557
5.503.5.2 flags	2558
5.503.5.3 flags	2558
5.503.5.4 getloc	2558
5.503.5.5 imbue	2559
5.503.5.6 iword	2559

5.503.5.7 precision	2559
5.503.5.8 precision	2560
5.503.5.9 pword	2560
5.503.5.10 register_callback	2560
5.503.5.11 setf	2561
5.503.5.12 setf	2561
5.503.5.13 sync_with_stdio	2562
5.503.5.14 unsetf	2562
5.503.5.15 width	2562
5.503.5.16 width	2563
5.503.5.17 xalloc	2563
5.503.6 Member Data Documentation	2563
5.503.6.1 adjustfield	2563
5.503.6.2 app	2563
5.503.6.3 ate	2564
5.503.6.4 badbit	2564
5.503.6.5 basefield	2564
5.503.6.6 beg	2564
5.503.6.7 binary	2564
5.503.6.8 boolalpha	2565
5.503.6.9 cur	2565
5.503.6.10 dec	2565
5.503.6.11 lend	2565
5.503.6.12 eofbit	2565
5.503.6.13 failbit	2566
5.503.6.14 fixed	2566
5.503.6.15 floatfield	2566
5.503.6.16 goodbit	2566
5.503.6.17 hex	2567
5.503.6.18 n	2567
5.503.6.19 internal	2567

5.503.6.20left	2567
5.503.6.21loct	2568
5.503.6.22out	2568
5.503.6.23right	2568
5.503.6.24scientific	2568
5.503.6.25showbase	2568
5.503.6.26showpoint	2568
5.503.6.27showpos	2568
5.503.6.28skipws	2569
5.503.6.29trunc	2569
5.503.6.30unitbuf	2569
5.503.6.31uppercase	2569
5.504std::ios_base::failure Class Reference	2569
5.504.1 Detailed Description	2570
5.504.2 Member Function Documentation	2570
5.504.2.1 what	2570
5.505std::is_base_of< _Base, _Derived > Struct Template Reference	2571
5.505.1 Detailed Description	2571
5.506std::is_bind_expression< _Tp > Struct Template Reference	2571
5.506.1 Detailed Description	2571
5.507std::is_bind_expression< _Bind< _Signature > > Struct Template Reference	2572
5.507.1 Detailed Description	2572
5.508std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference	2572
5.508.1 Detailed Description	2572
5.509std::is_constructible< _Tp, _Args > Struct Template Reference	2573
5.509.1 Detailed Description	2573
5.510std::is_convertible< _From, _To > Struct Template Reference	2574
5.510.1 Detailed Description	2574
5.511std::is_error_code_enum< _Tp > Struct Template Reference	2574
5.511.1 Detailed Description	2574

5.512	<code>std::is_error_condition_enum< _Tp ></code>	Struct Template Reference	2575
5.512.1	Detailed Description		2575
5.513	<code>std::is_explicitly_convertible< _From, _To ></code>	Struct Template Reference	2575
5.513.1	Detailed Description		2576
5.514	<code>std::is_lvalue_reference< typename ></code>	Struct Template Reference	2576
5.514.1	Detailed Description		2577
5.515	<code>std::is_nothrow_constructible< _Tp, _Args ></code>	Struct Template Reference	2577
5.515.1	Detailed Description		2577
5.516	<code>std::is_placeholder< _Tp ></code>	Struct Template Reference	2577
5.516.1	Detailed Description		2578
5.517	<code>std::is_placeholder< _Placeholder< _Num > ></code>	Struct Template Reference	2578
5.517.1	Detailed Description		2578
5.518	<code>std::is_pod< _Tp ></code>	Struct Template Reference	2578
5.518.1	Detailed Description		2578
5.519	<code>std::is_reference< _Tp ></code>	Struct Template Reference	2579
5.519.1	Detailed Description		2579
5.520	<code>std::is_rvalue_reference< typename ></code>	Struct Template Reference	2579
5.520.1	Detailed Description		2579
5.521	<code>std::is_signed< _Tp ></code>	Struct Template Reference	2580
5.521.1	Detailed Description		2580
5.522	<code>std::is_standard_layout< _Tp ></code>	Struct Template Reference	2580
5.522.1	Detailed Description		2580
5.523	<code>std::is_trivial< _Tp ></code>	Struct Template Reference	2580
5.523.1	Detailed Description		2581
5.524	<code>std::is_unsigned< _Tp ></code>	Struct Template Reference	2581
5.524.1	Detailed Description		2581
5.525	<code>std::istream_iterator< _Tp, _CharT, _Traits, _Dist ></code>	Class Template Reference	2581
5.525.1	Detailed Description		2583
5.525.2	Member Typedef Documentation		2583
5.525.2.1	difference_type		2583

5.525.2.2 iterator_category	2583
5.525.2.3 pointer	2583
5.525.2.4 reference	2583
5.525.2.5 value_type	2583
5.525.3 Constructor & Destructor Documentation	2584
5.525.3.1 istream_iterator	2584
5.525.3.2 istream_iterator	2584
5.526 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference	2584
5.526.1 Detailed Description	2585
5.526.2 Member Typedef Documentation	2586
5.526.2.1 char_type	2586
5.526.2.2 difference_type	2586
5.526.2.3 int_type	2586
5.526.2.4 istream_type	2586
5.526.2.5 iterator_category	2586
5.526.2.6 pointer	2586
5.526.2.7 reference	2587
5.526.2.8 streambuf_type	2587
5.526.2.9 traits_type	2587
5.526.2.10 value_type	2587
5.526.3 Constructor & Destructor Documentation	2587
5.526.3.1 istreambuf_iterator	2587
5.526.3.2 istreambuf_iterator	2588
5.526.3.3 istreambuf_iterator	2588
5.526.4 Member Function Documentation	2588
5.526.4.1 equal	2588
5.526.4.2 operator*	2588
5.526.4.3 operator++	2588
5.526.4.4 operator++	2589
5.527 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference	2589

5.527.1 Detailed Description	2590
5.527.2 Member Typedef Documentation	2590
5.527.2.1 difference_type	2590
5.527.2.2 iterator_category	2590
5.527.2.3 pointer	2590
5.527.2.4 reference	2591
5.527.2.5 value_type	2591
5.528std::iterator_traits< _Tp * > Struct Template Reference	2591
5.528.1 Detailed Description	2591
5.529std::iterator_traits< const _Tp * > Struct Template Reference	2592
5.529.1 Detailed Description	2592
5.530std::length_error Class Reference	2593
5.530.1 Detailed Description	2593
5.530.2 Member Function Documentation	2593
5.530.2.1 what	2593
5.531std::less< _Tp > Struct Template Reference	2594
5.531.1 Detailed Description	2595
5.531.2 Member Typedef Documentation	2595
5.531.2.1 first_argument_type	2595
5.531.2.2 result_type	2595
5.531.2.3 second_argument_type	2595
5.532std::less_equal< _Tp > Struct Template Reference	2595
5.532.1 Detailed Description	2596
5.532.2 Member Typedef Documentation	2597
5.532.2.1 first_argument_type	2597
5.532.2.2 result_type	2597
5.532.2.3 second_argument_type	2597
5.533std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference	2597
5.533.1 Detailed Description	2598
5.533.2 Member Typedef Documentation	2599

5.533.2.1 result_type	2599
5.533.3 Constructor & Destructor Documentation	2599
5.533.3.1 linear_congruential_engine	2599
5.533.3.2 linear_congruential_engine	2599
5.533.4 Member Function Documentation	2600
5.533.4.1 discard	2600
5.533.4.2 max	2600
5.533.4.3 min	2600
5.533.4.4 operator()	2601
5.533.4.5 seed	2601
5.533.4.6 seed	2601
5.533.5 Friends And Related Function Documentation	2602
5.533.5.1 operator<<	2602
5.533.5.2 operator==	2602
5.533.5.3 operator>>	2603
5.533.6 Member Data Documentation	2603
5.533.6.1 increment	2603
5.533.6.2 modulus	2603
5.533.6.3 multiplier	2603
5.534std::list< _Tp, _Alloc > Class Template Reference	2604
5.534.1 Detailed Description	2608
5.534.2 Constructor & Destructor Documentation	2608
5.534.2.1 list	2608
5.534.2.2 list	2609
5.534.2.3 list	2609
5.534.2.4 list	2609
5.534.2.5 list	2610
5.534.2.6 list	2610
5.534.2.7 list	2610
5.534.2.8 list	2611
5.534.3 Member Function Documentation	2611

5.534.3.1	_M_create_node	2611
5.534.3.2	assign	2611
5.534.3.3	assign	2612
5.534.3.4	assign	2612
5.534.3.5	back	2612
5.534.3.6	back	2613
5.534.3.7	begin	2613
5.534.3.8	begin	2613
5.534.3.9	cbegin	2613
5.534.3.10	end	2614
5.534.3.11	clear	2614
5.534.3.12	rbegin	2614
5.534.3.13	rend	2614
5.534.3.14	emplace	2614
5.534.3.15	empty	2615
5.534.3.16	end	2615
5.534.3.17	end	2615
5.534.3.18	erase	2616
5.534.3.19	erase	2616
5.534.3.20	front	2617
5.534.3.21	lfront	2617
5.534.3.22	get_allocator	2617
5.534.3.23	insert	2617
5.534.3.24	insert	2618
5.534.3.25	insert	2618
5.534.3.26	insert	2619
5.534.3.27	insert	2619
5.534.3.28	max_size	2619
5.534.3.29	merge	2620
5.534.3.30	merge	2620
5.534.3.31	operator=	2620

5.534.3.32operator=	2621
5.534.3.33operator=	2621
5.534.3.34pop_back	2621
5.534.3.35pop_front	2622
5.534.3.36push_back	2622
5.534.3.37push_front	2622
5.534.3.38rbegin	2623
5.534.3.39rbegin	2623
5.534.3.40remove	2623
5.534.3.41remove_if	2624
5.534.3.42rend	2624
5.534.3.43rend	2624
5.534.3.44resize	2624
5.534.3.45resize	2625
5.534.3.46reverse	2625
5.534.3.47size	2625
5.534.3.48sort	2626
5.534.3.49sort	2626
5.534.3.50splice	2626
5.534.3.51splice	2627
5.534.3.52splice	2627
5.534.3.53swap	2627
5.534.3.54unique	2628
5.534.3.55unique	2628
5.535std::locale Class Reference	2629
5.535.1 Detailed Description	2630
5.535.2 Member Typedef Documentation	2631
5.535.2.1 category	2631
5.535.3 Constructor & Destructor Documentation	2631
5.535.3.1 locale	2631
5.535.3.2 locale	2631

5.535.3.3 locale	2631
5.535.3.4 locale	2632
5.535.3.5 locale	2632
5.535.3.6 locale	2632
5.535.3.7 ~locale	2633
5.535.4 Member Function Documentation	2633
5.535.4.1 classic	2633
5.535.4.2 combine	2633
5.535.4.3 global	2633
5.535.4.4 name	2634
5.535.4.5 operator!=	2634
5.535.4.6 operator()	2634
5.535.4.7 operator=	2635
5.535.4.8 operator==	2635
5.535.5 Friends And Related Function Documentation	2635
5.535.5.1 has_facet	2635
5.535.5.2 use_facet	2636
5.535.6 Member Data Documentation	2636
5.535.6.1 all	2636
5.535.6.2 collate	2636
5.535.6.3 ctype	2637
5.535.6.4 messages	2637
5.535.6.5 monetary	2637
5.535.6.6 none	2637
5.535.6.7 numeric	2638
5.535.6.8 time	2638
5.536std::locale::facet Class Reference	2638
5.536.1 Detailed Description	2640
5.536.2 Constructor & Destructor Documentation	2640
5.536.2.1 facet	2640
5.536.2.2 ~facet	2640

5.537std::locale::id Class Reference	2640
5.537.1 Detailed Description	2641
5.537.2 Constructor & Destructor Documentation	2641
5.537.2.1 id	2641
5.537.3 Friends And Related Function Documentation	2641
5.537.3.1 has_facet	2641
5.537.3.2 use_facet	2642
5.538std::lock_guard< _Mutex > Class Template Reference	2642
5.538.1 Detailed Description	2643
5.539std::logic_error Class Reference	2643
5.539.1 Detailed Description	2644
5.539.2 Constructor & Destructor Documentation	2644
5.539.2.1 logic_error	2644
5.539.3 Member Function Documentation	2644
5.539.3.1 what	2644
5.540std::logical_and< _Tp > Struct Template Reference	2644
5.540.1 Detailed Description	2645
5.540.2 Member Typedef Documentation	2646
5.540.2.1 first_argument_type	2646
5.540.2.2 result_type	2646
5.540.2.3 second_argument_type	2646
5.541std::logical_not< _Tp > Struct Template Reference	2646
5.541.1 Detailed Description	2647
5.541.2 Member Typedef Documentation	2648
5.541.2.1 argument_type	2648
5.541.2.2 result_type	2648
5.542std::logical_or< _Tp > Struct Template Reference	2648
5.542.1 Detailed Description	2649
5.542.2 Member Typedef Documentation	2650
5.542.2.1 first_argument_type	2650
5.542.2.2 result_type	2650

5.542.2.3 second_argument_type	2650
5.543std::lognormal_distribution< _RealType > Class Template Reference	2650
5.543.1 Detailed Description	2651
5.543.2 Member Typedef Documentation	2652
5.543.2.1 result_type	2652
5.543.3 Member Function Documentation	2652
5.543.3.1 max	2652
5.543.3.2 min	2652
5.543.3.3 operator()	2652
5.543.3.4 param	2652
5.543.3.5 param	2653
5.543.3.6 reset	2653
5.543.4 Friends And Related Function Documentation	2653
5.543.4.1 operator<<	2653
5.543.4.2 operator==	2654
5.543.4.3 operator>>	2654
5.544std::lognormal_distribution< _RealType >::param_type Struct Reference	2654
5.544.1 Detailed Description	2655
5.545std::make_signed< _Tp > Struct Template Reference	2655
5.545.1 Detailed Description	2655
5.546std::make_unsigned< _Tp > Struct Template Reference	2655
5.546.1 Detailed Description	2656
5.547std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2656
5.547.1 Detailed Description	2658
5.547.2 Constructor & Destructor Documentation	2659
5.547.2.1 map	2659
5.547.2.2 map	2659
5.547.2.3 map	2659
5.547.2.4 map	2660
5.547.2.5 map	2660

5.547.2.6 map	2660
5.547.2.7 map	2661
5.547.3 Member Function Documentation	2661
5.547.3.1 at	2661
5.547.3.2 begin	2662
5.547.3.3 begin	2662
5.547.3.4 cbegin	2662
5.547.3.5 cend	2662
5.547.3.6 clear	2663
5.547.3.7 count	2663
5.547.3.8 crbegin	2663
5.547.3.9 crend	2664
5.547.3.10empty	2664
5.547.3.11lend	2664
5.547.3.12end	2664
5.547.3.13equal_range	2665
5.547.3.14equal_range	2665
5.547.3.15erase	2666
5.547.3.16erase	2666
5.547.3.17erase	2667
5.547.3.18find	2667
5.547.3.19find	2668
5.547.3.20get_allocator	2668
5.547.3.21insert	2668
5.547.3.22insert	2669
5.547.3.23insert	2669
5.547.3.24insert	2670
5.547.3.25key_comp	2670
5.547.3.26lower_bound	2670
5.547.3.27lower_bound	2671
5.547.3.28max_size	2671

5.547.3.29	operator=	2671
5.547.3.30	operator=	2672
5.547.3.31	operator=	2672
5.547.3.32	operator[]	2673
5.547.3.33	rbegin	2673
5.547.3.34	rbegin	2673
5.547.3.35	rend	2674
5.547.3.36	rend	2674
5.547.3.37	size	2674
5.547.3.38	swap	2674
5.547.3.39	upper_bound	2675
5.547.3.40	upper_bound	2675
5.547.3.41	lvalue_comp	2675
5.548	std::mask_array< _Tp > Class Template Reference	2676
5.548.1	Detailed Description	2677
5.548.2	Member Function Documentation	2677
5.548.2.1	operator%=	2677
5.548.2.2	operator&=	2678
5.548.2.3	operator*=	2678
5.548.2.4	operator+=	2678
5.548.2.5	operator-=	2678
5.548.2.6	operator/=	2678
5.548.2.7	operator<<=	2678
5.548.2.8	operator>>=	2678
5.548.2.9	operator^=	2678
5.548.2.10	operator =	2679
5.549	std::match_results< _Bi_iter, _Allocator > Class Template Reference	2679
5.549.1	Detailed Description	2683
5.549.2	Constructor & Destructor Documentation	2683
5.549.2.1	match_results	2683
5.549.2.2	match_results	2684

5.549.2.3 ~match_results	2684
5.549.3 Member Function Documentation	2684
5.549.3.1 begin	2684
5.549.3.2 cbegin	2684
5.549.3.3 cend	2685
5.549.3.4 empty	2685
5.549.3.5 end	2685
5.549.3.6 format	2686
5.549.3.7 format	2686
5.549.3.8 get_allocator	2686
5.549.3.9 length	2686
5.549.3.10 max_size	2687
5.549.3.11 operator=	2687
5.549.3.12 operator[]	2687
5.549.3.13 position	2688
5.549.3.14 prefix	2688
5.549.3.15 size	2688
5.549.3.16 str	2689
5.549.3.17 suffix	2689
5.549.3.18 swap	2689
5.550 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2690
5.550.1 Detailed Description	2691
5.550.2 Member Typedef Documentation	2691
5.550.2.1 first_argument_type	2691
5.550.2.2 result_type	2691
5.550.2.3 second_argument_type	2691
5.551 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2691
5.551.1 Detailed Description	2692
5.551.2 Member Typedef Documentation	2693
5.551.2.1 first_argument_type	2693
5.551.2.2 result_type	2693

5.551.2.3 second_argument_type	2693
5.552std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2693
5.552.1 Detailed Description	2694
5.552.2 Member Typedef Documentation	2695
5.552.2.1 argument_type	2695
5.552.2.2 result_type	2695
5.553std::mem_fun_t< _Ret, _Tp > Class Template Reference	2695
5.553.1 Detailed Description	2696
5.553.2 Member Typedef Documentation	2697
5.553.2.1 argument_type	2697
5.553.2.2 result_type	2697
5.554std::messages< _CharT > Class Template Reference	2697
5.554.1 Detailed Description	2699
5.554.2 Member Typedef Documentation	2700
5.554.2.1 char_type	2700
5.554.2.2 string_type	2700
5.554.3 Constructor & Destructor Documentation	2700
5.554.3.1 messages	2700
5.554.3.2 messages	2701
5.554.3.3 ~messages	2701
5.554.4 Member Data Documentation	2701
5.554.4.1 id	2701
5.555std::messages_base Struct Reference	2701
5.555.1 Detailed Description	2702
5.556std::messages_byname< _CharT > Class Template Reference	2702
5.556.1 Detailed Description	2704
5.556.2 Member Typedef Documentation	2705
5.556.2.1 char_type	2705
5.556.2.2 string_type	2705
5.556.3 Member Data Documentation	2705
5.556.3.1 id	2705

5.557std::minus< _Tp > Struct Template Reference	2705
5.557.1 Detailed Description	2706
5.557.2 Member Typedef Documentation	2707
5.557.2.1 first_argument_type	2707
5.557.2.2 result_type	2707
5.557.2.3 second_argument_type	2707
5.558std::modulus< _Tp > Struct Template Reference	2707
5.558.1 Detailed Description	2708
5.558.2 Member Typedef Documentation	2709
5.558.2.1 first_argument_type	2709
5.558.2.2 result_type	2709
5.558.2.3 second_argument_type	2709
5.559std::money_base Class Reference	2709
5.559.1 Detailed Description	2710
5.560std::money_get< _CharT, _InIter > Class Template Reference	2711
5.560.1 Detailed Description	2713
5.560.2 Member Typedef Documentation	2713
5.560.2.1 char_type	2713
5.560.2.2 iter_type	2713
5.560.2.3 string_type	2713
5.560.3 Constructor & Destructor Documentation	2714
5.560.3.1 money_get	2714
5.560.3.2 ~money_get	2714
5.560.4 Member Function Documentation	2714
5.560.4.1 do_get	2714
5.560.4.2 do_get	2715
5.560.4.3 get	2715
5.560.4.4 get	2716
5.560.5 Member Data Documentation	2716
5.560.5.1 id	2716
5.561std::money_put< _CharT, _OutIter > Class Template Reference . . .	2717

5.561.1 Detailed Description	2718
5.561.2 Member Typedef Documentation	2719
5.561.2.1 char_type	2719
5.561.2.2 iter_type	2719
5.561.2.3 string_type	2719
5.561.3 Constructor & Destructor Documentation	2719
5.561.3.1 money_put	2719
5.561.3.2 ~money_put	2719
5.561.4 Member Function Documentation	2720
5.561.4.1 do_put	2720
5.561.4.2 do_put	2720
5.561.4.3 put	2721
5.561.4.4 put	2722
5.561.5 Member Data Documentation	2722
5.561.5.1 id	2722
5.562std::moneypunct< _CharT, _Intl > Class Template Reference	2723
5.562.1 Detailed Description	2725
5.562.2 Member Typedef Documentation	2725
5.562.2.1 char_type	2725
5.562.2.2 string_type	2726
5.562.3 Constructor & Destructor Documentation	2726
5.562.3.1 moneypunct	2726
5.562.3.2 moneypunct	2726
5.562.3.3 moneypunct	2726
5.562.3.4 ~moneypunct	2727
5.562.4 Member Function Documentation	2727
5.562.4.1 curr_symbol	2727
5.562.4.2 decimal_point	2727
5.562.4.3 do_curr_symbol	2728
5.562.4.4 do_decimal_point	2728
5.562.4.5 do_frac_digits	2728

5.562.4.6 do_grouping	2729
5.562.4.7 do_neg_format	2729
5.562.4.8 do_negative_sign	2730
5.562.4.9 do_pos_format	2730
5.562.4.10do_positive_sign	2730
5.562.4.11do_thousands_sep	2731
5.562.4.12frac_digits	2731
5.562.4.13grouping	2732
5.562.4.14neg_format	2732
5.562.4.15negative_sign	2733
5.562.4.16pos_format	2733
5.562.4.17positive_sign	2734
5.562.4.18thousands_sep	2734
5.562.5 Member Data Documentation	2735
5.562.5.1 id	2735
5.562.5.2 intl	2735
5.563std::moneypunct_byname< _CharT, _Intl > Class Template Reference	2735
5.563.1 Detailed Description	2738
5.563.2 Member Typedef Documentation	2738
5.563.2.1 char_type	2738
5.563.2.2 string_type	2738
5.563.3 Member Function Documentation	2739
5.563.3.1 curr_symbol	2739
5.563.3.2 decimal_point	2739
5.563.3.3 do_curr_symbol	2739
5.563.3.4 do_decimal_point	2740
5.563.3.5 do_frac_digits	2740
5.563.3.6 do_grouping	2741
5.563.3.7 do_neg_format	2741
5.563.3.8 do_negative_sign	2741
5.563.3.9 do_pos_format	2742

5.563.3.10do_positive_sign	2742
5.563.3.11do_thousands_sep	2743
5.563.3.12frac_digits	2743
5.563.3.13grouping	2743
5.563.3.14neg_format	2744
5.563.3.15negative_sign	2745
5.563.3.16pos_format	2745
5.563.3.17positive_sign	2746
5.563.3.18thousands_sep	2746
5.563.4 Member Data Documentation	2747
5.563.4.1 id	2747
5.563.4.2 intl	2747
5.564std::move_iterator< _Iterator > Class Template Reference	2747
5.564.1 Detailed Description	2748
5.565std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2748
5.565.1 Detailed Description	2751
5.565.2 Constructor & Destructor Documentation	2751
5.565.2.1 multimap	2751
5.565.2.2 multimap	2751
5.565.2.3 multimap	2752
5.565.2.4 multimap	2752
5.565.2.5 multimap	2752
5.565.2.6 multimap	2753
5.565.2.7 multimap	2753
5.565.3 Member Function Documentation	2754
5.565.3.1 begin	2754
5.565.3.2 begin	2754
5.565.3.3 cbegin	2754
5.565.3.4 cend	2754
5.565.3.5 clear	2755

5.565.3.6	count	2755
5.565.3.7	crbegin	2755
5.565.3.8	crend	2755
5.565.3.9	empty	2756
5.565.3.10	end	2756
5.565.3.1	lend	2756
5.565.3.12	equal_range	2756
5.565.3.13	equal_range	2757
5.565.3.14	erase	2757
5.565.3.15	erase	2758
5.565.3.16	erase	2758
5.565.3.17	find	2759
5.565.3.18	find	2759
5.565.3.19	get_allocator	2760
5.565.3.20	insert	2760
5.565.3.2	insert	2760
5.565.3.22	insert	2761
5.565.3.23	insert	2761
5.565.3.24	key_comp	2762
5.565.3.25	lower_bound	2762
5.565.3.26	lower_bound	2762
5.565.3.27	max_size	2763
5.565.3.28	operator=	2763
5.565.3.29	operator=	2763
5.565.3.30	operator=	2764
5.565.3.3	lrbegin	2764
5.565.3.32	rbegin	2764
5.565.3.33	rend	2765
5.565.3.34	rend	2765
5.565.3.35	size	2765
5.565.3.36	swap	2765

5.565.3.37upper_bound	2766
5.565.3.38upper_bound	2766
5.565.3.39value_comp	2766
5.566std::multiplies< _Tp > Struct Template Reference	2767
5.566.1 Detailed Description	2768
5.566.2 Member Typedef Documentation	2768
5.566.2.1 first_argument_type	2768
5.566.2.2 result_type	2768
5.566.2.3 second_argument_type	2768
5.567std::multiset< _Key, _Compare, _Alloc > Class Template Reference	2768
5.567.1 Detailed Description	2770
5.567.2 Constructor & Destructor Documentation	2771
5.567.2.1 multiset	2771
5.567.2.2 multiset	2771
5.567.2.3 multiset	2771
5.567.2.4 multiset	2772
5.567.2.5 multiset	2772
5.567.2.6 multiset	2773
5.567.2.7 multiset	2773
5.567.3 Member Function Documentation	2773
5.567.3.1 begin	2773
5.567.3.2 cbegin	2774
5.567.3.3 cend	2774
5.567.3.4 clear	2774
5.567.3.5 count	2774
5.567.3.6 crbegin	2775
5.567.3.7 crend	2775
5.567.3.8 empty	2775
5.567.3.9 end	2775
5.567.3.10equal_range	2775
5.567.3.11lequal_range	2776

5.567.3.12	erase	2776
5.567.3.13	erase	2777
5.567.3.14	erase	2777
5.567.3.15	find	2778
5.567.3.16	find	2778
5.567.3.17	get_allocator	2779
5.567.3.18	insert	2779
5.567.3.19	insert	2779
5.567.3.20	insert	2780
5.567.3.21	insert	2780
5.567.3.22	key_comp	2781
5.567.3.23	lower_bound	2781
5.567.3.24	lower_bound	2781
5.567.3.25	max_size	2782
5.567.3.26	operator=	2782
5.567.3.27	operator=	2782
5.567.3.28	operator=	2783
5.567.3.29	begin	2783
5.567.3.30	end	2783
5.567.3.31	size	2783
5.567.3.32	swap	2784
5.567.3.33	upper_bound	2784
5.567.3.34	upper_bound	2784
5.567.3.35	value_comp	2785
5.567.4	Friends And Related Function Documentation	2785
5.567.4.1	operator<	2785
5.567.4.2	operator==	2786
5.568	std::mutex Class Reference	2786
5.568.1	Detailed Description	2787
5.569	std::negate< _Tp > Struct Template Reference	2787
5.569.1	Detailed Description	2788

5.569.2 Member Typedef Documentation	2788
5.569.2.1 argument_type	2788
5.569.2.2 result_type	2788
5.570std::negative_binomial_distribution< _IntType > Class Template Reference	2788
5.570.1 Detailed Description	2789
5.570.2 Member Typedef Documentation	2790
5.570.2.1 result_type	2790
5.570.3 Member Function Documentation	2790
5.570.3.1 k	2790
5.570.3.2 max	2790
5.570.3.3 min	2790
5.570.3.4 operator()	2790
5.570.3.5 p	2791
5.570.3.6 param	2791
5.570.3.7 param	2791
5.570.3.8 reset	2791
5.570.4 Friends And Related Function Documentation	2792
5.570.4.1 operator<<	2792
5.570.4.2 operator==	2792
5.570.4.3 operator>>	2792
5.571std::negative_binomial_distribution< _IntType >::param_type Struct Reference	2793
5.571.1 Detailed Description	2793
5.572std::nested_exception Class Reference	2793
5.572.1 Detailed Description	2794
5.573std::normal_distribution< _RealType > Class Template Reference	2794
5.573.1 Detailed Description	2795
5.573.2 Member Typedef Documentation	2796
5.573.2.1 result_type	2796
5.573.3 Constructor & Destructor Documentation	2796
5.573.3.1 normal_distribution	2796

5.573.4 Member Function Documentation	2796
5.573.4.1 max	2796
5.573.4.2 mean	2796
5.573.4.3 min	2797
5.573.4.4 operator()	2797
5.573.4.5 operator()	2797
5.573.4.6 param	2797
5.573.4.7 param	2798
5.573.4.8 reset	2798
5.573.4.9 stddev	2798
5.573.5 Friends And Related Function Documentation	2798
5.573.5.1 operator<<	2798
5.573.5.2 operator==	2799
5.573.5.3 operator>>	2799
5.574std::normal_distribution< _RealType >::param_type Struct Reference	2799
5.574.1 Detailed Description	2800
5.575std::not_equal_to< _Tp > Struct Template Reference	2800
5.575.1 Detailed Description	2801
5.575.2 Member Typedef Documentation	2802
5.575.2.1 first_argument_type	2802
5.575.2.2 result_type	2802
5.575.2.3 second_argument_type	2802
5.576std::num_get< _CharT, _InIter > Class Template Reference	2802
5.576.1 Detailed Description	2805
5.576.2 Member Typedef Documentation	2805
5.576.2.1 char_type	2805
5.576.2.2 iter_type	2806
5.576.3 Constructor & Destructor Documentation	2806
5.576.3.1 num_get	2806
5.576.3.2 ~num_get	2806
5.576.4 Member Function Documentation	2806

5.576.4.1 do_get	2806
5.576.4.2 do_get	2807
5.576.4.3 do_get	2808
5.576.4.4 do_get	2808
5.576.4.5 do_get	2809
5.576.4.6 do_get	2809
5.576.4.7 do_get	2810
5.576.4.8 do_get	2811
5.576.4.9 do_get	2811
5.576.4.10do_get	2812
5.576.4.1ldo_get	2813
5.576.4.12get	2813
5.576.4.13get	2814
5.576.4.14get	2815
5.576.4.15get	2816
5.576.4.16get	2816
5.576.4.17get	2817
5.576.4.18get	2818
5.576.4.19get	2819
5.576.4.20get	2820
5.576.4.2lget	2820
5.576.4.22get	2821
5.576.5 Member Data Documentation	2822
5.576.5.1 id	2822
5.577std::num_put< _CharT, _OutIter > Class Template Reference	2822
5.577.1 Detailed Description	2825
5.577.2 Member Typedef Documentation	2825
5.577.2.1 char_type	2825
5.577.2.2 iter_type	2825
5.577.3 Constructor & Destructor Documentation	2826
5.577.3.1 num_put	2826

5.577.3.2 ~num_put	2826
5.577.4 Member Function Documentation	2826
5.577.4.1 do_put	2826
5.577.4.2 do_put	2827
5.577.4.3 do_put	2827
5.577.4.4 do_put	2828
5.577.4.5 do_put	2828
5.577.4.6 do_put	2829
5.577.4.7 do_put	2829
5.577.4.8 do_put	2830
5.577.4.9 put	2830
5.577.4.10put	2831
5.577.4.11put	2832
5.577.4.12put	2833
5.577.4.13put	2833
5.577.4.14put	2834
5.577.4.15put	2835
5.577.4.16put	2836
5.577.5 Member Data Documentation	2837
5.577.5.1 id	2837
5.578std::numeric_limits< _Tp > Struct Template Reference	2837
5.578.1 Detailed Description	2839
5.578.2 Member Function Documentation	2839
5.578.2.1 denorm_min	2839
5.578.2.2 epsilon	2839
5.578.2.3 infinity	2839
5.578.2.4 lowest	2839
5.578.2.5 max	2840
5.578.2.6 min	2840
5.578.2.7 quiet_NaN	2840
5.578.2.8 round_error	2840

5.578.2.9 signaling_NaN	2840
5.578.3 Member Data Documentation	2840
5.578.3.1 digits	2840
5.578.3.2 digits10	2841
5.578.3.3 has_denorm	2841
5.578.3.4 has_denorm_loss	2841
5.578.3.5 has_infinity	2841
5.578.3.6 has_quiet_NaN	2841
5.578.3.7 has_signaling_NaN	2841
5.578.3.8 is_bounded	2842
5.578.3.9 is_exact	2842
5.578.3.10 is_iec559	2842
5.578.3.11 is_integer	2842
5.578.3.12 is_modulo	2842
5.578.3.13 is_signed	2842
5.578.3.14 is_specialized	2843
5.578.3.15 max_digits10	2843
5.578.3.16 max_exponent	2843
5.578.3.17 max_exponent10	2843
5.578.3.18 min_exponent	2843
5.578.3.19 min_exponent10	2844
5.578.3.20 radix	2844
5.578.3.21 round_style	2844
5.578.3.22 tinyness_before	2844
5.578.3.23 traps	2844
5.579 std::numeric_limits< bool > Struct Template Reference	2845
5.579.1 Detailed Description	2846
5.580 std::numeric_limits< char > Struct Template Reference	2846
5.580.1 Detailed Description	2847
5.581 std::numeric_limits< char16_t > Struct Template Reference	2847
5.581.1 Detailed Description	2848

5.582std::numeric_limits< char32_t > Struct Template Reference	2848
5.582.1 Detailed Description	2850
5.583std::numeric_limits< double > Struct Template Reference	2850
5.583.1 Detailed Description	2851
5.584std::numeric_limits< float > Struct Template Reference	2851
5.584.1 Detailed Description	2852
5.585std::numeric_limits< int > Struct Template Reference	2852
5.585.1 Detailed Description	2854
5.586std::numeric_limits< long > Struct Template Reference	2854
5.586.1 Detailed Description	2855
5.587std::numeric_limits< long double > Struct Template Reference	2855
5.587.1 Detailed Description	2856
5.588std::numeric_limits< long long > Struct Template Reference	2856
5.588.1 Detailed Description	2858
5.589std::numeric_limits< short > Struct Template Reference	2858
5.589.1 Detailed Description	2859
5.590std::numeric_limits< signed char > Struct Template Reference	2859
5.590.1 Detailed Description	2860
5.591std::numeric_limits< unsigned char > Struct Template Reference	2860
5.591.1 Detailed Description	2862
5.592std::numeric_limits< unsigned int > Struct Template Reference	2862
5.592.1 Detailed Description	2863
5.593std::numeric_limits< unsigned long > Struct Template Reference	2863
5.593.1 Detailed Description	2864
5.594std::numeric_limits< unsigned long long > Struct Template Reference	2864
5.594.1 Detailed Description	2866
5.595std::numeric_limits< unsigned short > Struct Template Reference	2866
5.595.1 Detailed Description	2867
5.596std::numeric_limits< wchar_t > Struct Template Reference	2867
5.596.1 Detailed Description	2868
5.597std::num_punct< _CharT > Class Template Reference	2868

5.597.1 Detailed Description	2871
5.597.2 Member Typedef Documentation	2871
5.597.2.1 char_type	2871
5.597.2.2 string_type	2871
5.597.3 Constructor & Destructor Documentation	2871
5.597.3.1 numpunct	2871
5.597.3.2 numpunct	2872
5.597.3.3 numpunct	2872
5.597.3.4 ~numpunct	2872
5.597.4 Member Function Documentation	2872
5.597.4.1 decimal_point	2872
5.597.4.2 do_decimal_point	2873
5.597.4.3 do_falsename	2873
5.597.4.4 do_grouping	2874
5.597.4.5 do_thousands_sep	2874
5.597.4.6 do_truename	2874
5.597.4.7 falsename	2875
5.597.4.8 grouping	2875
5.597.4.9 thousands_sep	2876
5.597.4.10truename	2876
5.597.5 Member Data Documentation	2876
5.597.5.1 id	2876
5.598std::numpunct_byname<_CharT> Class Template Reference	2877
5.598.1 Detailed Description	2878
5.598.2 Member Typedef Documentation	2879
5.598.2.1 char_type	2879
5.598.2.2 string_type	2879
5.598.3 Member Function Documentation	2879
5.598.3.1 decimal_point	2879
5.598.3.2 do_decimal_point	2879
5.598.3.3 do_falsename	2880

5.598.3.4 do_grouping	2880
5.598.3.5 do_thousands_sep	2881
5.598.3.6 do_truename	2881
5.598.3.7 falsename	2881
5.598.3.8 grouping	2882
5.598.3.9 thousands_sep	2882
5.598.3.10truename	2882
5.598.4 Member Data Documentation	2883
5.598.4.1 id	2883
5.599std::once_flag Struct Reference	2883
5.599.1 Detailed Description	2883
5.599.2 Friends And Related Function Documentation	2884
5.599.2.1 call_once	2884
5.600std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference	2884
5.600.1 Detailed Description	2885
5.600.2 Member Typedef Documentation	2885
5.600.2.1 char_type	2885
5.600.2.2 difference_type	2886
5.600.2.3 iterator_category	2886
5.600.2.4 ostream_type	2886
5.600.2.5 pointer	2886
5.600.2.6 reference	2886
5.600.2.7 traits_type	2886
5.600.2.8 value_type	2887
5.600.3 Constructor & Destructor Documentation	2887
5.600.3.1 ostream_iterator	2887
5.600.3.2 ostream_iterator	2887
5.600.3.3 ostream_iterator	2887
5.600.4 Member Function Documentation	2888
5.600.4.1 operator=	2888
5.601std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference	2888

5.601.1 Detailed Description	2889
5.601.2 Member Typedef Documentation	2890
5.601.2.1 char_type	2890
5.601.2.2 difference_type	2890
5.601.2.3 iterator_category	2890
5.601.2.4 ostream_type	2890
5.601.2.5 pointer	2890
5.601.2.6 reference	2890
5.601.2.7 streambuf_type	2891
5.601.2.8 traits_type	2891
5.601.2.9 value_type	2891
5.601.3 Constructor & Destructor Documentation	2891
5.601.3.1 ostreambuf_iterator	2891
5.601.3.2 ostreambuf_iterator	2891
5.601.4 Member Function Documentation	2892
5.601.4.1 failed	2892
5.601.4.2 operator*	2892
5.601.4.3 operator++	2892
5.601.4.4 operator++	2892
5.601.4.5 operator=	2892
5.602std::out_of_range Class Reference	2893
5.602.1 Detailed Description	2893
5.602.2 Member Function Documentation	2893
5.602.2.1 what	2893
5.603std::output_iterator_tag Struct Reference	2894
5.603.1 Detailed Description	2894
5.604std::overflow_error Class Reference	2895
5.604.1 Detailed Description	2895
5.604.2 Member Function Documentation	2895
5.604.2.1 what	2895
5.605std::owner_less< shared_ptr< _Tp > > Struct Template Reference	2896

5.605.1 Detailed Description	2896
5.605.2 Member Typedef Documentation	2896
5.605.2.1 first_argument_type	2896
5.605.2.2 result_type	2897
5.605.2.3 second_argument_type	2897
5.606std::owner_less< weak_ptr< _Tp > > Struct Template Reference	2897
5.606.1 Detailed Description	2897
5.606.2 Member Typedef Documentation	2898
5.606.2.1 first_argument_type	2898
5.606.2.2 result_type	2898
5.606.2.3 second_argument_type	2898
5.607std::packaged_task< _Res(_ArgTypes...) > Class Template Reference	2898
5.607.1 Detailed Description	2899
5.608std::pair< _T1, _T2 > Struct Template Reference	2899
5.608.1 Detailed Description	2900
5.608.2 Member Typedef Documentation	2901
5.608.2.1 first_type	2901
5.608.2.2 second_type	2901
5.608.3 Constructor & Destructor Documentation	2901
5.608.3.1 pair	2901
5.608.3.2 pair	2901
5.608.3.3 pair	2901
5.608.4 Member Data Documentation	2902
5.608.4.1 first	2902
5.608.4.2 second	2902
5.609std::piecewise_constant_distribution< _RealType > Class Template Reference	2902
5.609.1 Detailed Description	2903
5.609.2 Member Typedef Documentation	2904
5.609.2.1 result_type	2904
5.609.3 Member Function Documentation	2904

5.609.3.1 densities	2904
5.609.3.2 intervals	2904
5.609.3.3 max	2904
5.609.3.4 min	2904
5.609.3.5 operator()	2905
5.609.3.6 param	2905
5.609.3.7 param	2905
5.609.3.8 reset	2905
5.609.4 Friends And Related Function Documentation	2906
5.609.4.1 operator<<	2906
5.609.4.2 operator>>	2906
5.610std::piecewise_constant_distribution< _RealType >::param_type Struct Reference	2907
5.610.1 Detailed Description	2907
5.611std::piecewise_linear_distribution< _RealType > Class Template Ref- erence	2908
5.611.1 Detailed Description	2909
5.611.2 Member Typedef Documentation	2909
5.611.2.1 result_type	2909
5.611.3 Member Function Documentation	2909
5.611.3.1 densities	2909
5.611.3.2 intervals	2909
5.611.3.3 max	2910
5.611.3.4 min	2910
5.611.3.5 operator()	2910
5.611.3.6 param	2910
5.611.3.7 param	2911
5.611.3.8 reset	2911
5.611.4 Friends And Related Function Documentation	2911
5.611.4.1 operator<<	2911
5.611.4.2 operator>>	2912

5.612std::piecewise_linear_distribution< _RealType >::param_type Struct Reference	2912
5.612.1 Detailed Description	2913
5.613std::plus< _Tp > Struct Template Reference	2913
5.613.1 Detailed Description	2914
5.613.2 Member Typedef Documentation	2914
5.613.2.1 first_argument_type	2914
5.613.2.2 result_type	2914
5.613.2.3 second_argument_type	2914
5.614std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference	2915
5.614.1 Detailed Description	2916
5.614.2 Member Typedef Documentation	2916
5.614.2.1 first_argument_type	2916
5.614.2.2 result_type	2916
5.614.2.3 second_argument_type	2916
5.615std::pointer_to_unary_function< _Arg, _Result > Class Template Reference	2916
5.615.1 Detailed Description	2917
5.615.2 Member Typedef Documentation	2918
5.615.2.1 argument_type	2918
5.615.2.2 result_type	2918
5.616std::poisson_distribution< _IntType > Class Template Reference	2918
5.616.1 Detailed Description	2919
5.616.2 Member Typedef Documentation	2919
5.616.2.1 result_type	2919
5.616.3 Member Function Documentation	2920
5.616.3.1 max	2920
5.616.3.2 mean	2920
5.616.3.3 min	2920
5.616.3.4 operator()	2920
5.616.3.5 operator()	2921

5.616.3.6 param	2921
5.616.3.7 param	2921
5.616.3.8 reset	2921
5.616.4 Friends And Related Function Documentation	2922
5.616.4.1 operator<<	2922
5.616.4.2 operator==	2922
5.616.4.3 operator>>	2922
5.617std::poisson_distribution< _IntType >::param_type Struct Reference	2923
5.617.1 Detailed Description	2923
5.618std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference	2923
5.618.1 Detailed Description	2924
5.618.2 Constructor & Destructor Documentation	2925
5.618.2.1 priority_queue	2925
5.618.2.2 priority_queue	2925
5.618.3 Member Function Documentation	2926
5.618.3.1 empty	2926
5.618.3.2 pop	2926
5.618.3.3 push	2927
5.618.3.4 size	2927
5.618.3.5 top	2927
5.619std::promise< _Res > Class Template Reference	2928
5.619.1 Detailed Description	2928
5.620std::promise< _Res & > Class Template Reference	2928
5.620.1 Detailed Description	2929
5.621std::promise< void > Class Template Reference	2929
5.621.1 Detailed Description	2930
5.622std::queue< _Tp, _Sequence > Class Template Reference	2930
5.622.1 Detailed Description	2931
5.622.2 Constructor & Destructor Documentation	2932
5.622.2.1 queue	2932

5.622.3 Member Function Documentation	2932
5.622.3.1 back	2932
5.622.3.2 back	2932
5.622.3.3 empty	2932
5.622.3.4 front	2932
5.622.3.5 front	2933
5.622.3.6 pop	2933
5.622.3.7 push	2933
5.622.3.8 size	2933
5.622.4 Member Data Documentation	2934
5.622.4.1 c	2934
5.623std::random_access_iterator_tag Struct Reference	2934
5.623.1 Detailed Description	2935
5.624std::random_device Class Reference	2935
5.624.1 Detailed Description	2936
5.624.2 Member Typedef Documentation	2936
5.624.2.1 result_type	2936
5.625std::range_error Class Reference	2937
5.625.1 Detailed Description	2937
5.625.2 Member Function Documentation	2937
5.625.2.1 what	2937
5.626std::ratio< _Num, _Den > Struct Template Reference	2938
5.626.1 Detailed Description	2938
5.627std::ratio_add< _R1, _R2 > Struct Template Reference	2939
5.627.1 Detailed Description	2939
5.628std::ratio_divide< _R1, _R2 > Struct Template Reference	2939
5.628.1 Detailed Description	2940
5.629std::ratio_equal< _R1, _R2 > Struct Template Reference	2940
5.629.1 Detailed Description	2940
5.630std::ratio_multiply< _R1, _R2 > Struct Template Reference	2940
5.630.1 Detailed Description	2941

5.631std::ratio_not_equal< _R1, _R2 > Struct Template Reference	2941
5.631.1 Detailed Description	2941
5.632std::ratio_subtract< _R1, _R2 > Struct Template Reference	2941
5.632.1 Detailed Description	2942
5.633std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference	2942
5.633.1 Detailed Description	2943
5.633.2 Member Typedef Documentation	2943
5.633.2.1 difference_type	2943
5.633.2.2 iterator_category	2943
5.633.2.3 pointer	2943
5.633.2.4 reference	2944
5.633.2.5 value_type	2944
5.634std::recursive_mutex Class Reference	2944
5.634.1 Detailed Description	2944
5.635std::recursive_timed_mutex Class Reference	2945
5.635.1 Detailed Description	2945
5.636std::reference_wrapper< _Tp > Class Template Reference	2945
5.636.1 Detailed Description	2947
5.637std::regex_error Class Reference	2947
5.637.1 Detailed Description	2948
5.637.2 Constructor & Destructor Documentation	2948
5.637.2.1 regex_error	2948
5.637.3 Member Function Documentation	2948
5.637.3.1 code	2948
5.637.3.2 what	2948
5.638std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	2949
5.638.1 Detailed Description	2949
5.638.2 Constructor & Destructor Documentation	2950
5.638.2.1 regex_iterator	2950
5.638.2.2 regex_iterator	2950

5.638.2.3 <code>regex_iterator</code>	2950
5.638.3 Member Function Documentation	2951
5.638.3.1 <code>operator!=</code>	2951
5.638.3.2 <code>operator*</code>	2951
5.638.3.3 <code>operator++</code>	2951
5.638.3.4 <code>operator++</code>	2952
5.638.3.5 <code>operator-></code>	2952
5.638.3.6 <code>operator=</code>	2952
5.638.3.7 <code>operator==</code>	2952
5.639 <code>std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class</code> Template Reference	2953
5.639.1 Detailed Description	2954
5.639.2 Constructor & Destructor Documentation	2954
5.639.2.1 <code>regex_token_iterator</code>	2954
5.639.2.2 <code>regex_token_iterator</code>	2954
5.639.2.3 <code>regex_token_iterator</code>	2955
5.639.2.4 <code>regex_token_iterator</code>	2956
5.639.2.5 <code>regex_token_iterator</code>	2956
5.639.3 Member Function Documentation	2957
5.639.3.1 <code>operator!=</code>	2957
5.639.3.2 <code>operator*</code>	2957
5.639.3.3 <code>operator++</code>	2957
5.639.3.4 <code>operator++</code>	2958
5.639.3.5 <code>operator-></code>	2958
5.639.3.6 <code>operator=</code>	2958
5.639.3.7 <code>operator==</code>	2959
5.640 <code>std::regex_traits< _Ch_type > Struct Template Reference</code>	2959
5.640.1 Detailed Description	2960
5.640.2 Constructor & Destructor Documentation	2960
5.640.2.1 <code>regex_traits</code>	2960
5.640.3 Member Function Documentation	2960

5.640.3.1	getloc	2960
5.640.3.2	imbue	2961
5.640.3.3	length	2961
5.640.3.4	lookup_classname	2961
5.640.3.5	lookup_collatename	2963
5.640.3.6	transform	2963
5.640.3.7	transform_primary	2964
5.640.3.8	translate	2964
5.640.3.9	translate_nocase	2965
5.641	std::remove_reference<_Tp> Struct Template Reference	2965
5.641.1	Detailed Description	2965
5.642	std::reverse_iterator<_Iterator> Class Template Reference	2966
5.642.1	Detailed Description	2967
5.642.2	Member Typedef Documentation	2967
5.642.2.1	difference_type	2967
5.642.2.2	iterator_category	2967
5.642.2.3	pointer	2968
5.642.2.4	reference	2968
5.642.2.5	value_type	2968
5.642.3	Constructor & Destructor Documentation	2968
5.642.3.1	reverse_iterator	2968
5.642.3.2	reverse_iterator	2969
5.642.3.3	reverse_iterator	2969
5.642.3.4	reverse_iterator	2969
5.642.4	Member Function Documentation	2969
5.642.4.1	base	2969
5.642.4.2	operator*	2969
5.642.4.3	operator+	2970
5.642.4.4	operator++	2970
5.642.4.5	operator++	2970
5.642.4.6	operator+=	2971

5.642.4.7 operator-	2971
5.642.4.8 operator--	2971
5.642.4.9 operator--	2972
5.642.4.10operator-=	2972
5.642.4.11operator->	2972
5.642.4.12operator[]	2972
5.643std::runtime_error Class Reference	2973
5.643.1 Detailed Description	2974
5.643.2 Constructor & Destructor Documentation	2974
5.643.2.1 runtime_error	2974
5.643.3 Member Function Documentation	2974
5.643.3.1 what	2974
5.644std::seed_seq Class Reference	2974
5.644.1 Detailed Description	2975
5.644.2 Member Typedef Documentation	2975
5.644.2.1 result_type	2975
5.644.3 Constructor & Destructor Documentation	2975
5.644.3.1 seed_seq	2975
5.645std::set< _Key, _Compare, _Alloc > Class Template Reference	2975
5.645.1 Detailed Description	2978
5.645.2 Member Typedef Documentation	2978
5.645.2.1 allocator_type	2978
5.645.2.2 const_iterator	2979
5.645.2.3 const_pointer	2979
5.645.2.4 const_reference	2979
5.645.2.5 const_reverse_iterator	2979
5.645.2.6 difference_type	2979
5.645.2.7 iterator	2980
5.645.2.8 key_compare	2980
5.645.2.9 key_type	2980
5.645.2.10pointer	2980

5.645.2.1	reference	2980
5.645.2.12	reverse_iterator	2981
5.645.2.13	size_type	2981
5.645.2.14	value_compare	2981
5.645.2.15	value_type	2981
5.645.3	Constructor & Destructor Documentation	2981
5.645.3.1	set	2981
5.645.3.2	set	2982
5.645.3.3	set	2982
5.645.3.4	set	2982
5.645.3.5	set	2983
5.645.3.6	set	2983
5.645.3.7	set	2983
5.645.4	Member Function Documentation	2984
5.645.4.1	begin	2984
5.645.4.2	cbegin	2984
5.645.4.3	cend	2984
5.645.4.4	clear	2985
5.645.4.5	count	2985
5.645.4.6	crbegin	2985
5.645.4.7	crend	2985
5.645.4.8	empty	2986
5.645.4.9	end	2986
5.645.4.10	equal_range	2986
5.645.4.11	lequal_range	2987
5.645.4.12	erase	2987
5.645.4.13	erase	2988
5.645.4.14	erase	2988
5.645.4.15	find	2988
5.645.4.16	find	2989
5.645.4.17	get_allocator	2989

5.645.4.18	insert	2989
5.645.4.19	insert	2990
5.645.4.20	insert	2991
5.645.4.21	insert	2991
5.645.4.22	key_comp	2991
5.645.4.23	lower_bound	2991
5.645.4.24	lower_bound	2992
5.645.4.25	max_size	2992
5.645.4.26	operator=	2993
5.645.4.27	operator=	2993
5.645.4.28	operator=	2993
5.645.4.29	begin	2994
5.645.4.30	end	2994
5.645.4.31	size	2994
5.645.4.32	swap	2994
5.645.4.33	upper_bound	2995
5.645.4.34	upper_bound	2995
5.645.4.35	value_comp	2995
5.646	std::shared_future< _Res > Class Template Reference	2996
5.646.1	Detailed Description	2997
5.646.2	Constructor & Destructor Documentation	2997
5.646.2.1	shared_future	2997
5.646.2.2	shared_future	2997
5.646.2.3	shared_future	2998
5.646.3	Member Function Documentation	2998
5.646.3.1	_M_get_result	2998
5.646.3.2	get	2998
5.647	std::shared_future< _Res & > Class Template Reference	2998
5.647.1	Detailed Description	3000
5.647.2	Constructor & Destructor Documentation	3000
5.647.2.1	shared_future	3000

5.647.2.2 shared_future	3000
5.647.2.3 shared_future	3001
5.647.3 Member Function Documentation	3001
5.647.3.1 _M_get_result	3001
5.647.3.2 get	3001
5.648std::shared_future< void > Class Template Reference	3001
5.648.1 Detailed Description	3003
5.648.2 Constructor & Destructor Documentation	3003
5.648.2.1 shared_future	3003
5.648.2.2 shared_future	3003
5.648.2.3 shared_future	3004
5.648.3 Member Function Documentation	3004
5.648.3.1 _M_get_result	3004
5.649std::shared_ptr< _Tp > Class Template Reference	3004
5.649.1 Detailed Description	3006
5.649.2 Constructor & Destructor Documentation	3006
5.649.2.1 shared_ptr	3006
5.649.2.2 shared_ptr	3007
5.649.2.3 shared_ptr	3007
5.649.2.4 shared_ptr	3008
5.649.2.5 shared_ptr	3008
5.649.2.6 shared_ptr	3009
5.649.2.7 shared_ptr	3009
5.649.2.8 shared_ptr	3010
5.649.2.9 shared_ptr	3010
5.649.2.10shared_ptr	3010
5.649.2.11shared_ptr	3011
5.649.2.12shared_ptr	3011
5.649.3 Friends And Related Function Documentation	3011
5.649.3.1 allocate_shared	3011

5.650std::shuffle_order_engine< _RandomNumberEngine, __k > Class	
Template Reference	3012
5.650.1 Detailed Description	3013
5.650.2 Member Typedef Documentation	3013
5.650.2.1 result_type	3013
5.650.3 Constructor & Destructor Documentation	3014
5.650.3.1 shuffle_order_engine	3014
5.650.3.2 shuffle_order_engine	3014
5.650.3.3 shuffle_order_engine	3014
5.650.3.4 shuffle_order_engine	3014
5.650.3.5 shuffle_order_engine	3015
5.650.4 Member Function Documentation	3015
5.650.4.1 base	3015
5.650.4.2 discard	3015
5.650.4.3 max	3016
5.650.4.4 min	3016
5.650.4.5 operator()	3016
5.650.4.6 seed	3016
5.650.4.7 seed	3017
5.650.4.8 seed	3017
5.650.5 Friends And Related Function Documentation	3017
5.650.5.1 operator<<	3017
5.650.5.2 operator==	3018
5.650.5.3 operator>>	3018
5.651std::slice Class Reference	3018
5.651.1 Detailed Description	3019
5.652std::slice_array< _Tp > Class Template Reference	3019
5.652.1 Detailed Description	3020
5.652.2 Member Function Documentation	3021
5.652.2.1 operator%=	3021
5.652.2.2 operator&=	3021

5.652.2.3 operator*= 5.652.2.4 operator+= 5.652.2.5 operator-= 5.652.2.6 operator/=	3021 3021 3021 3021
5.652.2.7 operator<<= 5.652.2.8 operator>>= 5.652.2.9 operator^= 5.652.2.10operator = 5.653std::stack< _Tp, _Sequence > Class Template Reference	3022 3022 3022 3022 3022
5.653.1 Detailed Description 5.653.2 Constructor & Destructor Documentation	3024 3024
5.653.2.1 stack 5.653.3 Member Function Documentation	3024 3025
5.653.3.1 empty 5.653.3.2 pop 5.653.3.3 push 5.653.3.4 size 5.653.3.5 top 5.653.3.6 top	3025 3025 3025 3025 3026 3026
5.654std::student_t_distribution< _RealType > Class Template Reference 5.654.1 Detailed Description 5.654.2 Member Typedef Documentation	3026 3027 3027
5.654.2.1 result_type 5.654.3 Member Function Documentation	3027 3028
5.654.3.1 max 5.654.3.2 min 5.654.3.3 operator() 5.654.3.4 param 5.654.3.5 param 5.654.3.6 reset 5.654.4 Friends And Related Function Documentation	3028 3028 3028 3028 3028 3029 3029

5.654.4.1 operator<<	3029
5.654.4.2 operator==	3029
5.654.4.3 operator>>	3030
5.655std::student_t_distribution<_RealType>::param_type Struct Reference	3030
5.655.1 Detailed Description	3031
5.656std::sub_match<_BiIter> Class Template Reference	3031
5.656.1 Detailed Description	3032
5.656.2 Member Typedef Documentation	3033
5.656.2.1 first_type	3033
5.656.2.2 second_type	3033
5.656.3 Member Function Documentation	3033
5.656.3.1 compare	3033
5.656.3.2 compare	3033
5.656.3.3 compare	3034
5.656.3.4 length	3034
5.656.3.5 operator string_type	3034
5.656.3.6 str	3035
5.656.4 Member Data Documentation	3035
5.656.4.1 first	3035
5.656.4.2 second	3035
5.657std::system_error Class Reference	3036
5.657.1 Detailed Description	3036
5.657.2 Member Function Documentation	3037
5.657.2.1 what	3037
5.658std::thread Class Reference	3037
5.658.1 Detailed Description	3038
5.658.2 Member Function Documentation	3038
5.658.2.1 native_handle	3038
5.659std::thread::id Class Reference	3038
5.659.1 Detailed Description	3039
5.660std::time_base Class Reference	3039

5.660.1 Detailed Description	3040
5.661std::time_get< _CharT, _InIter > Class Template Reference	3040
5.661.1 Detailed Description	3042
5.661.2 Member Typedef Documentation	3043
5.661.2.1 char_type	3043
5.661.2.2 iter_type	3043
5.661.3 Constructor & Destructor Documentation	3043
5.661.3.1 time_get	3043
5.661.3.2 ~time_get	3043
5.661.4 Member Function Documentation	3044
5.661.4.1 date_order	3044
5.661.4.2 do_date_order	3044
5.661.4.3 do_get_date	3044
5.661.4.4 do_get_monthname	3045
5.661.4.5 do_get_time	3046
5.661.4.6 do_get_weekday	3046
5.661.4.7 do_get_year	3047
5.661.4.8 get_date	3048
5.661.4.9 get_monthname	3048
5.661.4.10get_time	3049
5.661.4.11get_weekday	3050
5.661.4.12get_year	3050
5.661.5 Member Data Documentation	3051
5.661.5.1 id	3051
5.662std::time_get_byname< _CharT, _InIter > Class Template Reference	3051
5.662.1 Detailed Description	3054
5.662.2 Member Typedef Documentation	3054
5.662.2.1 char_type	3054
5.662.2.2 iter_type	3054
5.662.3 Member Function Documentation	3054
5.662.3.1 date_order	3054

5.662.3.2 do_date_order	3055
5.662.3.3 do_get_date	3055
5.662.3.4 do_get_monthname	3056
5.662.3.5 do_get_time	3056
5.662.3.6 do_get_weekday	3057
5.662.3.7 do_get_year	3058
5.662.3.8 get_date	3058
5.662.3.9 get_monthname	3059
5.662.3.10get_time	3060
5.662.3.11get_weekday	3060
5.662.3.12get_year	3061
5.662.4 Member Data Documentation	3062
5.662.4.1 id	3062
5.663std::time_put< _CharT, _OutIter > Class Template Reference	3062
5.663.1 Detailed Description	3064
5.663.2 Member Typedef Documentation	3064
5.663.2.1 char_type	3064
5.663.2.2 iter_type	3065
5.663.3 Constructor & Destructor Documentation	3065
5.663.3.1 time_put	3065
5.663.3.2 ~time_put	3065
5.663.4 Member Function Documentation	3065
5.663.4.1 do_put	3065
5.663.4.2 put	3066
5.663.4.3 put	3067
5.663.5 Member Data Documentation	3067
5.663.5.1 id	3067
5.664std::time_put_byname< _CharT, _OutIter > Class Template Reference	3068
5.664.1 Detailed Description	3069
5.664.2 Member Typedef Documentation	3069
5.664.2.1 char_type	3069

5.664.2.2 iter_type	3069
5.664.3 Member Function Documentation	3070
5.664.3.1 do_put	3070
5.664.3.2 put	3070
5.664.3.3 put	3071
5.664.4 Member Data Documentation	3072
5.664.4.1 id	3072
5.665std::timed_mutex Class Reference	3072
5.665.1 Detailed Description	3072
5.666std::tr1::__is_member_pointer_helper< _Tp > Struct Template Reference	3073
5.666.1 Detailed Description	3073
5.667std::tr1::add_const< _Tp > Struct Template Reference	3074
5.667.1 Detailed Description	3074
5.668std::tr1::add_cv< _Tp > Struct Template Reference	3074
5.668.1 Detailed Description	3074
5.669std::tr1::add_pointer< _Tp > Struct Template Reference	3075
5.669.1 Detailed Description	3075
5.670std::tr1::add_volatile< _Tp > Struct Template Reference	3075
5.670.1 Detailed Description	3075
5.671std::tr1::alignment_of< _Tp > Struct Template Reference	3076
5.671.1 Detailed Description	3076
5.672std::tr1::array< _Tp, _Nm > Struct Template Reference	3077
5.672.1 Detailed Description	3078
5.673std::tr1::bad_weak_ptr Class Reference	3078
5.673.1 Detailed Description	3079
5.673.2 Member Function Documentation	3079
5.673.2.1 what	3079
5.674std::tr1::extent< typename, _UInt > Struct Template Reference	3080
5.674.1 Detailed Description	3080
5.675std::tr1::has_virtual_destructor< _Tp > Struct Template Reference	3081

5.675.1 Detailed Description	3082
5.676std::tr1::integral_constant< _Tp, __v > Struct Template Reference	3082
5.676.1 Detailed Description	3084
5.677std::tr1::is_abstract< _Tp > Struct Template Reference	3084
5.677.1 Detailed Description	3085
5.678std::tr1::is_arithmetic< _Tp > Struct Template Reference	3085
5.678.1 Detailed Description	3086
5.679std::tr1::is_array< typename > Struct Template Reference	3086
5.679.1 Detailed Description	3087
5.680std::tr1::is_class< _Tp > Struct Template Reference	3087
5.680.1 Detailed Description	3088
5.681std::tr1::is_compound< _Tp > Struct Template Reference	3088
5.681.1 Detailed Description	3089
5.682std::tr1::is_const< typename > Struct Template Reference	3089
5.682.1 Detailed Description	3090
5.683std::tr1::is_empty< _Tp > Struct Template Reference	3090
5.683.1 Detailed Description	3091
5.684std::tr1::is_enum< _Tp > Struct Template Reference	3092
5.684.1 Detailed Description	3092
5.685std::tr1::is_floating_point< _Tp > Struct Template Reference	3093
5.685.1 Detailed Description	3093
5.686std::tr1::is_function< typename > Struct Template Reference	3094
5.686.1 Detailed Description	3094
5.687std::tr1::is_fundamental< _Tp > Struct Template Reference	3095
5.687.1 Detailed Description	3095
5.688std::tr1::is_integral< _Tp > Struct Template Reference	3095
5.688.1 Detailed Description	3096
5.689std::tr1::is_member_function_pointer< _Tp > Struct Template Reference	3096
5.689.1 Detailed Description	3097
5.690std::tr1::is_member_object_pointer< _Tp > Struct Template Reference	3097

5.690.1 Detailed Description	3098
5.691std::tr1::is_object< _Tp > Struct Template Reference	3098
5.691.1 Detailed Description	3098
5.692std::tr1::is_pointer< _Tp > Struct Template Reference	3099
5.692.1 Detailed Description	3099
5.693std::tr1::is_polymorphic< _Tp > Struct Template Reference	3099
5.693.1 Detailed Description	3100
5.694std::tr1::is_same< typename, typename > Struct Template Reference	3101
5.694.1 Detailed Description	3101
5.695std::tr1::is_scalar< _Tp > Struct Template Reference	3102
5.695.1 Detailed Description	3102
5.696std::tr1::is_union< _Tp > Struct Template Reference	3102
5.696.1 Detailed Description	3103
5.697std::tr1::is_void< _Tp > Struct Template Reference	3104
5.697.1 Detailed Description	3104
5.698std::tr1::is_volatile< typename > Struct Template Reference	3104
5.698.1 Detailed Description	3105
5.699std::tr1::rank< typename > Struct Template Reference	3105
5.699.1 Detailed Description	3106
5.700std::tr1::remove_all_extents< _Tp > Struct Template Reference	3107
5.700.1 Detailed Description	3107
5.701std::tr1::remove_const< _Tp > Struct Template Reference	3107
5.701.1 Detailed Description	3107
5.702std::tr1::remove_cv< _Tp > Struct Template Reference	3108
5.702.1 Detailed Description	3108
5.703std::tr1::remove_extent< _Tp > Struct Template Reference	3108
5.703.1 Detailed Description	3108
5.704std::tr1::remove_pointer< _Tp > Struct Template Reference	3109
5.704.1 Detailed Description	3109
5.705std::tr1::remove_volatile< _Tp > Struct Template Reference	3109
5.705.1 Detailed Description	3109

5.706std::try_to_lock_t Struct Reference	3110
5.706.1 Detailed Description	3110
5.707std::tuple< _Elements > Class Template Reference	3110
5.707.1 Detailed Description	3111
5.708std::tuple< _T1 > Class Template Reference	3111
5.708.1 Detailed Description	3112
5.709std::tuple< _T1, _T2 > Class Template Reference	3112
5.709.1 Detailed Description	3113
5.710std::tuple_element< 0, tuple< _Head, _Tail...> > Struct Template Reference	3113
5.710.1 Detailed Description	3113
5.711std::tuple_element< __i, tuple< _Head, _Tail...> > Struct Template Reference	3114
5.711.1 Detailed Description	3114
5.712std::tuple_size< tuple< _Elements...> > Struct Template Reference .	3114
5.712.1 Detailed Description	3114
5.713std::type_info Class Reference	3115
5.713.1 Detailed Description	3115
5.713.2 Constructor & Destructor Documentation	3116
5.713.2.1 ~type_info	3116
5.713.3 Member Function Documentation	3116
5.713.3.1 name	3116
5.714std::unary_function< _Arg, _Result > Struct Template Reference . .	3117
5.714.1 Detailed Description	3118
5.714.2 Member Typedef Documentation	3118
5.714.2.1 argument_type	3118
5.714.2.2 result_type	3118
5.715std::unary_negate< _Predicate > Class Template Reference	3118
5.715.1 Detailed Description	3119
5.715.2 Member Typedef Documentation	3120
5.715.2.1 argument_type	3120
5.715.2.2 result_type	3120

5.716std::underflow_error Class Reference	3121
5.716.1 Detailed Description	3121
5.716.2 Member Function Documentation	3121
5.716.2.1 what	3121
5.717std::uniform_int_distribution< _IntType > Class Template Reference	3122
5.717.1 Detailed Description	3123
5.717.2 Member Typedef Documentation	3123
5.717.2.1 result_type	3123
5.717.3 Constructor & Destructor Documentation	3123
5.717.3.1 uniform_int_distribution	3123
5.717.4 Member Function Documentation	3123
5.717.4.1 max	3123
5.717.4.2 min	3124
5.717.4.3 operator()	3124
5.717.4.4 param	3124
5.717.4.5 param	3124
5.717.4.6 reset	3125
5.718std::uniform_int_distribution< _IntType >::param_type Struct Refer-	
ence	3125
5.718.1 Detailed Description	3125
5.719std::uniform_real_distribution< _RealType > Class Template Reference	3126
5.719.1 Detailed Description	3126
5.719.2 Member Typedef Documentation	3127
5.719.2.1 result_type	3127
5.719.3 Constructor & Destructor Documentation	3127
5.719.3.1 uniform_real_distribution	3127
5.719.4 Member Function Documentation	3127
5.719.4.1 max	3127
5.719.4.2 min	3127
5.719.4.3 operator()	3128
5.719.4.4 param	3128

5.719.4.5 param	3128
5.719.4.6 reset	3128
5.720std::uniform_real_distribution<_RealType>::param_type Struct Reference	3129
5.720.1 Detailed Description	3129
5.721std::unique_lock<_Mutex> Class Template Reference	3129
5.721.1 Detailed Description	3130
5.722std::unique_ptr<_Tp, _Dp> Class Template Reference	3131
5.722.1 Detailed Description	3132
5.723std::unique_ptr<_Tp[], _Dp> Class Template Reference	3132
5.723.1 Detailed Description	3133
5.724std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> Class Template Reference	3134
5.724.1 Detailed Description	3136
5.725std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> Class Template Reference	3136
5.725.1 Detailed Description	3138
5.726std::unordered_multiset<_Value, _Hash, _Pred, _Alloc> Class Template Reference	3139
5.726.1 Detailed Description	3141
5.727std::unordered_set<_Value, _Hash, _Pred, _Alloc> Class Template Reference	3141
5.727.1 Detailed Description	3144
5.728std::uses_allocator<_Tp, _Alloc> Struct Template Reference	3144
5.728.1 Detailed Description	3144
5.729std::valarray<_Tp> Class Template Reference	3145
5.729.1 Detailed Description	3147
5.729.2 Constructor & Destructor Documentation	3148
5.729.2.1 valarray	3148
5.730std::vector<_Tp, _Alloc> Class Template Reference	3148
5.730.1 Detailed Description	3152
5.730.2 Constructor & Destructor Documentation	3152
5.730.2.1 vector	3152

5.730.2.2	vector	3153
5.730.2.3	vector	3153
5.730.2.4	vector	3153
5.730.2.5	vector	3154
5.730.2.6	vector	3154
5.730.2.7	vector	3154
5.730.2.8	vector	3155
5.730.2.9	~vector	3155
5.730.3	Member Function Documentation	3155
5.730.3.1	_M_allocate_and_copy	3155
5.730.3.2	_M_range_check	3156
5.730.3.3	assign	3156
5.730.3.4	assign	3156
5.730.3.5	assign	3157
5.730.3.6	at	3157
5.730.3.7	at	3157
5.730.3.8	back	3158
5.730.3.9	back	3158
5.730.3.10	begin	3158
5.730.3.11	begin	3159
5.730.3.12	capacity	3159
5.730.3.13	begin	3159
5.730.3.14	end	3159
5.730.3.15	clear	3160
5.730.3.16	rbegin	3160
5.730.3.17	rend	3160
5.730.3.18	data	3160
5.730.3.19	emplace	3160
5.730.3.20	empty	3161
5.730.3.21	lend	3161
5.730.3.22	end	3161

5.730.3.23	erase	3162
5.730.3.24	erase	3162
5.730.3.25	front	3163
5.730.3.26	front	3163
5.730.3.27	insert	3163
5.730.3.28	insert	3163
5.730.3.29	insert	3164
5.730.3.30	insert	3164
5.730.3.31	insert	3165
5.730.3.32	max_size	3165
5.730.3.33	operator=	3165
5.730.3.34	operator=	3166
5.730.3.35	operator=	3166
5.730.3.36	operator[]	3167
5.730.3.37	operator[]	3167
5.730.3.38	pop_back	3167
5.730.3.39	push_back	3168
5.730.3.40	begin	3168
5.730.3.41	rbegin	3168
5.730.3.42	rend	3168
5.730.3.43	rend	3169
5.730.3.44	reserve	3169
5.730.3.45	resize	3169
5.730.3.46	resize	3170
5.730.3.47	shrink_to_fit	3170
5.730.3.48	size	3170
5.730.3.49	swap	3171
5.731	std::vector< bool, _Alloc > Class Template Reference	3171
5.731.1	Detailed Description	3175
5.732	std::weak_ptr< _Tp > Class Template Reference	3175
5.732.1	Detailed Description	3176

5.733	std::weibull_distribution< _RealType > Class Template Reference	3176
5.733.1	Detailed Description	3177
5.733.2	Member Typedef Documentation	3177
5.733.2.1	result_type	3177
5.733.3	Member Function Documentation	3178
5.733.3.1	a	3178
5.733.3.2	b	3178
5.733.3.3	max	3178
5.733.3.4	min	3178
5.733.3.5	operator()	3178
5.733.3.6	param	3179
5.733.3.7	param	3179
5.733.3.8	reset	3179
5.734	std::weibull_distribution< _RealType >::param_type Struct Reference	3179
5.734.1	Detailed Description	3180
6	File Documentation	3181
6.1	algo.h File Reference	3181
6.1.1	Detailed Description	3195
6.2	algbase.h File Reference	3195
6.2.1	Detailed Description	3197
6.3	algorithm File Reference	3197
6.3.1	Detailed Description	3197
6.4	algorithm File Reference	3197
6.4.1	Detailed Description	3199
6.5	algorithm File Reference	3199
6.5.1	Detailed Description	3199
6.6	algorithmfwd.h File Reference	3199
6.6.1	Detailed Description	3206
6.7	algorithmfwd.h File Reference	3206
6.7.1	Detailed Description	3218

6.8	allocator.h File Reference	3219
6.8.1	Detailed Description	3220
6.9	array File Reference	3220
6.9.1	Detailed Description	3220
6.10	array File Reference	3220
6.10.1	Detailed Description	3221
6.11	array_allocator.h File Reference	3221
6.11.1	Detailed Description	3222
6.12	assoc_container.hpp File Reference	3222
6.12.1	Detailed Description	3223
6.13	atomic File Reference	3223
6.13.1	Detailed Description	3227
6.14	atomic_0.h File Reference	3228
6.14.1	Detailed Description	3228
6.15	atomic_2.h File Reference	3228
6.15.1	Detailed Description	3229
6.16	atomic_base.h File Reference	3229
6.16.1	Detailed Description	3230
6.17	atomic_word.h File Reference	3231
6.17.1	Detailed Description	3231
6.18	atomicfwd_c.h File Reference	3231
6.18.1	Detailed Description	3232
6.19	atomicfwd_cxx.h File Reference	3232
6.19.1	Detailed Description	3233
6.20	atomicity.h File Reference	3233
6.20.1	Detailed Description	3234
6.21	auto_ptr.h File Reference	3234
6.21.1	Detailed Description	3234
6.22	balanced_quicksort.h File Reference	3234
6.22.1	Detailed Description	3235
6.23	base.h File Reference	3235

6.23.1 Detailed Description	3237
6.24 base.h File Reference	3237
6.24.1 Detailed Description	3237
6.25 basic_file.h File Reference	3238
6.25.1 Detailed Description	3238
6.26 basic_ios.h File Reference	3238
6.26.1 Detailed Description	3238
6.27 basic_ios.tcc File Reference	3239
6.27.1 Detailed Description	3239
6.28 basic_iterator.h File Reference	3239
6.28.1 Detailed Description	3239
6.29 basic_string.h File Reference	3239
6.29.1 Detailed Description	3243
6.30 basic_string.tcc File Reference	3243
6.30.1 Detailed Description	3243
6.31 basic_types.hpp File Reference	3244
6.31.1 Detailed Description	3244
6.32 binders.h File Reference	3244
6.32.1 Detailed Description	3245
6.33 bitmap_allocator.h File Reference	3245
6.33.1 Detailed Description	3247
6.33.2 Define Documentation	3247
6.33.2.1 _BALLOC_ALIGN_BYTES	3247
6.34 bitset File Reference	3247
6.34.1 Detailed Description	3248
6.35 bitset File Reference	3248
6.35.1 Detailed Description	3249
6.36 bitset File Reference	3249
6.36.1 Detailed Description	3250
6.37 boost_concept_check.h File Reference	3250
6.37.1 Detailed Description	3251

6.38	boost_sp_counted_base.h File Reference	3251
6.38.1	Detailed Description	3251
6.39	c++0x_warning.h File Reference	3252
6.39.1	Detailed Description	3252
6.40	c++allocator.h File Reference	3252
6.40.1	Detailed Description	3252
6.41	c++config.h File Reference	3252
6.41.1	Detailed Description	3257
6.42	c++io.h File Reference	3257
6.42.1	Detailed Description	3258
6.43	c++locale.h File Reference	3258
6.43.1	Detailed Description	3258
6.44	c++locale_internal.h File Reference	3259
6.44.1	Detailed Description	3259
6.45	cassert File Reference	3259
6.45.1	Detailed Description	3259
6.46	ccomplex File Reference	3259
6.46.1	Detailed Description	3259
6.47	ccomplex File Reference	3259
6.47.1	Detailed Description	3259
6.48	cctype File Reference	3260
6.48.1	Detailed Description	3260
6.49	cctype File Reference	3260
6.49.1	Detailed Description	3260
6.50	cctype File Reference	3260
6.50.1	Detailed Description	3260
6.51	cerrno File Reference	3261
6.51.1	Detailed Description	3261
6.52	cfenv File Reference	3261
6.52.1	Detailed Description	3261
6.53	cfenv File Reference	3261

6.53.1 Detailed Description	3261
6.54 cfenv File Reference	3262
6.54.1 Detailed Description	3262
6.55 cfloat File Reference	3262
6.55.1 Detailed Description	3262
6.56 cfloat File Reference	3262
6.56.1 Detailed Description	3262
6.57 char_traits.h File Reference	3263
6.57.1 Detailed Description	3263
6.58 checkers.h File Reference	3263
6.58.1 Detailed Description	3264
6.59 chrono File Reference	3264
6.59.1 Detailed Description	3267
6.60 cinttypes File Reference	3267
6.60.1 Detailed Description	3267
6.61 cinttypes File Reference	3267
6.61.1 Detailed Description	3268
6.62 cinttypes File Reference	3268
6.62.1 Detailed Description	3268
6.63 ciso646 File Reference	3268
6.63.1 Detailed Description	3268
6.64 climits File Reference	3268
6.64.1 Detailed Description	3268
6.65 climits File Reference	3269
6.65.1 Detailed Description	3269
6.66 clocale File Reference	3269
6.66.1 Detailed Description	3269
6.67 cmath File Reference	3270
6.67.1 Detailed Description	3272
6.68 cmath File Reference	3273
6.68.1 Detailed Description	3276

6.69	<code>cmath</code> File Reference	3276
6.69.1	Detailed Description	3276
6.70	<code>codecvt.h</code> File Reference	3276
6.70.1	Detailed Description	3277
6.71	<code>codecvt_specializations.h</code> File Reference	3277
6.71.1	Detailed Description	3278
6.72	<code>compatibility.h</code> File Reference	3278
6.72.1	Detailed Description	3278
6.73	<code>compatibility.h</code> File Reference	3278
6.73.1	Detailed Description	3279
6.74	<code>compiletime_settings.h</code> File Reference	3279
6.74.1	Detailed Description	3279
6.74.2	Define Documentation	3280
6.74.2.1	<code>_GLIBCXX_ASSERTIONS</code>	3280
6.74.2.2	<code>_GLIBCXX_CALL</code>	3280
6.74.2.3	<code>_GLIBCXX_RANDOM_SHUFFLE_-</code> <code>CONSIDER_L1</code>	3280
6.74.2.4	<code>_GLIBCXX_RANDOM_SHUFFLE_-</code> <code>CONSIDER_TLB</code>	3281
6.74.2.5	<code>_GLIBCXX_SCALE_DOWN_FPU</code>	3281
6.74.2.6	<code>_GLIBCXX_VERBOSE_LEVEL</code>	3281
6.75	<code>complex</code> File Reference	3281
6.75.1	Detailed Description	3285
6.76	<code>complex</code> File Reference	3285
6.76.1	Detailed Description	3286
6.77	<code>complex</code> File Reference	3286
6.77.1	Detailed Description	3287
6.78	<code>complex.h</code> File Reference	3287
6.78.1	Detailed Description	3287
6.79	<code>concept_check.h</code> File Reference	3288
6.79.1	Detailed Description	3288
6.80	<code>concurrency.h</code> File Reference	3288

6.80.1 Detailed Description	3289
6.81 cond_dealtor.hpp File Reference	3289
6.81.1 Detailed Description	3289
6.82 condition_variable File Reference	3289
6.82.1 Detailed Description	3290
6.83 constructors_destructor_fn_imps.hpp File Reference	3290
6.83.1 Detailed Description	3291
6.84 container_base_dispatch.hpp File Reference	3291
6.84.1 Detailed Description	3291
6.85 cpp_type_traits.h File Reference	3291
6.85.1 Detailed Description	3291
6.86 cpu_defines.h File Reference	3292
6.86.1 Detailed Description	3292
6.87 csetjmp File Reference	3292
6.87.1 Detailed Description	3292
6.88 csignal File Reference	3292
6.88.1 Detailed Description	3292
6.89 cstdarg File Reference	3293
6.89.1 Detailed Description	3293
6.90 cstdarg File Reference	3293
6.90.1 Detailed Description	3293
6.91 cstdbool File Reference	3293
6.91.1 Detailed Description	3293
6.92 cstdbool File Reference	3294
6.92.1 Detailed Description	3294
6.93 cstddef File Reference	3294
6.93.1 Detailed Description	3294
6.94 cstdint File Reference	3294
6.94.1 Detailed Description	3294
6.95 cstdint File Reference	3294
6.95.1 Detailed Description	3294

6.96	cstdint File Reference	3295
6.96.1	Detailed Description	3295
6.97	cstdio File Reference	3295
6.97.1	Detailed Description	3295
6.98	cstdio File Reference	3295
6.98.1	Detailed Description	3296
6.99	cstdio File Reference	3296
6.99.1	Detailed Description	3296
6.100	cstdlib File Reference	3296
6.100.1	Detailed Description	3297
6.101	cstdlib File Reference	3297
6.101.1	Detailed Description	3297
6.102	cstdlib File Reference	3297
6.102.1	Detailed Description	3297
6.103	cstring File Reference	3297
6.103.1	Detailed Description	3298
6.104	ctgmath File Reference	3298
6.104.1	Detailed Description	3298
6.105	ctgmath File Reference	3298
6.105.1	Detailed Description	3298
6.106	ctime File Reference	3298
6.106.1	Detailed Description	3299
6.107	ctime File Reference	3299
6.107.1	Detailed Description	3299
6.108	ctype_base.h File Reference	3299
6.108.1	Detailed Description	3299
6.109	ctype_inline.h File Reference	3300
6.109.1	Detailed Description	3300
6.110	ctype_noninline.h File Reference	3300
6.110.1	Detailed Description	3300
6.111	cwchar File Reference	3300

6.111.1 Detailed Description	3300
6.112cwchar File Reference	3301
6.112.1 Detailed Description	3301
6.113cwchar File Reference	3301
6.113.1 Detailed Description	3301
6.114cwctype File Reference	3301
6.114.1 Detailed Description	3302
6.115cwctype File Reference	3302
6.115.1 Detailed Description	3302
6.116cwctype File Reference	3302
6.116.1 Detailed Description	3302
6.117cxxabi-forced.h File Reference	3303
6.117.1 Detailed Description	3303
6.118cxxabi.h File Reference	3303
6.118.1 Detailed Description	3305
6.119cxxabi_tweaks.h File Reference	3305
6.119.1 Detailed Description	3305
6.120debug.h File Reference	3305
6.120.1 Detailed Description	3306
6.121debug_allocator.h File Reference	3306
6.121.1 Detailed Description	3307
6.122debug_map_base.hpp File Reference	3307
6.122.1 Detailed Description	3307
6.123decimal File Reference	3307
6.123.1 Detailed Description	3319
6.124deque File Reference	3319
6.124.1 Detailed Description	3319
6.125deque File Reference	3319
6.125.1 Detailed Description	3320
6.126deque File Reference	3320
6.126.1 Detailed Description	3321

6.127deque.tcc File Reference	3321
6.127.1 Detailed Description	3322
6.128enc_filebuf.h File Reference	3322
6.128.1 Detailed Description	3322
6.129equally_split.h File Reference	3323
6.129.1 Detailed Description	3323
6.130error_constants.h File Reference	3323
6.130.1 Detailed Description	3324
6.131exception File Reference	3324
6.131.1 Detailed Description	3325
6.132exception.hpp File Reference	3325
6.132.1 Detailed Description	3326
6.133exception_ptr.h File Reference	3326
6.133.1 Detailed Description	3326
6.134extptr_allocator.h File Reference	3327
6.134.1 Detailed Description	3327
6.135features.h File Reference	3327
6.135.1 Detailed Description	3328
6.135.2 Define Documentation	3328
6.135.2.1 _GLIBCXX_BAL_QUICKSORT	3328
6.135.2.2 _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS	3328
6.135.2.3 _GLIBCXX_FIND_EQUAL_SPLIT	3328
6.135.2.4 _GLIBCXX_FIND_GROWING_BLOCKS	3329
6.135.2.5 _GLIBCXX_MERGESORT	3329
6.135.2.6 _GLIBCXX_QUICKSORT	3329
6.135.2.7 _GLIBCXX_TREE_DYNAMIC_BALANCING	3329
6.135.2.8 _GLIBCXX_TREE_FULL_COPY	3330
6.135.2.9 _GLIBCXX_TREE_INITIAL_SPLITTING	3330
6.136fenv.h File Reference	3330
6.136.1 Detailed Description	3330
6.137find.h File Reference	3330

6.137.1 Detailed Description	3331
6.138find_selectors.h File Reference	3331
6.138.1 Detailed Description	3332
6.139for_each.h File Reference	3332
6.139.1 Detailed Description	3332
6.140for_each_selectors.h File Reference	3333
6.140.1 Detailed Description	3334
6.141formatter.h File Reference	3335
6.141.1 Detailed Description	3336
6.142forward_list.h File Reference	3336
6.142.1 Detailed Description	3337
6.143forward_list.tcc File Reference	3337
6.143.1 Detailed Description	3338
6.144fstream File Reference	3338
6.144.1 Detailed Description	3338
6.145fstream.tcc File Reference	3339
6.145.1 Detailed Description	3339
6.146functexcept.h File Reference	3339
6.146.1 Detailed Description	3340
6.147functional File Reference	3340
6.147.1 Detailed Description	3344
6.148functional File Reference	3344
6.148.1 Detailed Description	3345
6.149functional_hash.h File Reference	3345
6.149.1 Detailed Description	3346
6.150functions.h File Reference	3346
6.150.1 Detailed Description	3348
6.151future File Reference	3349
6.151.1 Detailed Description	3351
6.152gslice.h File Reference	3351
6.152.1 Detailed Description	3351

6.153gslice_array.h File Reference	3352
6.153.1 Detailed Description	3352
6.154hash_fun.h File Reference	3352
6.154.1 Detailed Description	3352
6.155hash_map File Reference	3353
6.155.1 Detailed Description	3353
6.156hash_policy.hpp File Reference	3354
6.156.1 Detailed Description	3354
6.157hash_set File Reference	3354
6.157.1 Detailed Description	3355
6.158hashtable.h File Reference	3355
6.158.1 Detailed Description	3356
6.159hashtable.h File Reference	3356
6.159.1 Detailed Description	3356
6.160hashtable_policy.h File Reference	3357
6.160.1 Detailed Description	3357
6.161indirect_array.h File Reference	3358
6.161.1 Detailed Description	3358
6.162initializer_list File Reference	3358
6.162.1 Detailed Description	3359
6.163iomanip File Reference	3359
6.163.1 Detailed Description	3360
6.164ios File Reference	3360
6.164.1 Detailed Description	3360
6.165ios_base.h File Reference	3361
6.165.1 Detailed Description	3363
6.166iosfwd File Reference	3363
6.166.1 Detailed Description	3364
6.167iostream File Reference	3364
6.167.1 Detailed Description	3364
6.168istream File Reference	3365

6.168.1 Detailed Description	3366
6.169istream.tcc File Reference	3366
6.169.1 Detailed Description	3366
6.170iterator File Reference	3367
6.170.1 Detailed Description	3367
6.171iterator File Reference	3367
6.171.1 Detailed Description	3367
6.172iterator.h File Reference	3367
6.172.1 Detailed Description	3368
6.173iterator_tracker.h File Reference	3368
6.173.1 Detailed Description	3370
6.174limits File Reference	3370
6.174.1 Detailed Description	3372
6.175list File Reference	3372
6.175.1 Detailed Description	3372
6.176list File Reference	3373
6.176.1 Detailed Description	3373
6.177list File Reference	3374
6.177.1 Detailed Description	3374
6.178list.tcc File Reference	3375
6.178.1 Detailed Description	3375
6.179list_partition.h File Reference	3375
6.179.1 Detailed Description	3375
6.180list_update_policy.hpp File Reference	3376
6.180.1 Detailed Description	3376
6.181locale File Reference	3376
6.181.1 Detailed Description	3376
6.182locale_classes.h File Reference	3376
6.182.1 Detailed Description	3377
6.183locale_classes.tcc File Reference	3377
6.183.1 Detailed Description	3378

6.184	locale_facets.h File Reference	3378
6.184.1	Detailed Description	3380
6.185	locale_facets.tcc File Reference	3381
6.185.1	Detailed Description	3381
6.186	locale_facets_nonio.h File Reference	3381
6.186.1	Detailed Description	3383
6.187	locale_facets_nonio.tcc File Reference	3383
6.187.1	Detailed Description	3383
6.188	localefwd.h File Reference	3383
6.188.1	Detailed Description	3384
6.189	losertree.h File Reference	3384
6.189.1	Detailed Description	3386
6.190	macros.h File Reference	3386
6.190.1	Detailed Description	3386
6.190.2	Define Documentation	3387
6.190.2.1	__glibcxx_check_erase	3387
6.190.2.2	__glibcxx_check_erase_after	3387
6.190.2.3	__glibcxx_check_erase_range	3387
6.190.2.4	__glibcxx_check_erase_range_after	3387
6.190.2.5	__glibcxx_check_heap_pred	3387
6.190.2.6	__glibcxx_check_insert	3387
6.190.2.7	__glibcxx_check_insert_after	3388
6.190.2.8	__glibcxx_check_insert_range	3388
6.190.2.9	__glibcxx_check_insert_range_after	3388
6.190.2.10	__glibcxx_check_partitioned_lower	3389
6.190.2.11	__glibcxx_check_partitioned_lower_pred	3389
6.190.2.12	__glibcxx_check_partitioned_upper_pred	3389
6.190.2.13	__glibcxx_check_sorted_pred	3389
6.190.2.14	GLIBCXX_DEBUG_VERIFY	3389
6.191	malloc_allocator.h File Reference	3390
6.191.1	Detailed Description	3390

6.192map File Reference	3390
6.192.1 Detailed Description	3390
6.193map File Reference	3391
6.193.1 Detailed Description	3391
6.194map File Reference	3391
6.194.1 Detailed Description	3391
6.195map.h File Reference	3391
6.195.1 Detailed Description	3392
6.196map.h File Reference	3392
6.196.1 Detailed Description	3393
6.197mask_array.h File Reference	3393
6.197.1 Detailed Description	3394
6.198memory File Reference	3394
6.198.1 Detailed Description	3394
6.199memory File Reference	3394
6.199.1 Detailed Description	3395
6.200merge.h File Reference	3395
6.200.1 Detailed Description	3396
6.201messages_members.h File Reference	3396
6.201.1 Detailed Description	3396
6.202move.h File Reference	3396
6.202.1 Detailed Description	3397
6.203mt_allocator.h File Reference	3397
6.203.1 Detailed Description	3399
6.204multimap.h File Reference	3399
6.204.1 Detailed Description	3400
6.205multimap.h File Reference	3400
6.205.1 Detailed Description	3401
6.206multiseq_selection.h File Reference	3401
6.206.1 Detailed Description	3402
6.207multiset.h File Reference	3402

6.207.1 Detailed Description	3403
6.208multiset.h File Reference	3403
6.208.1 Detailed Description	3404
6.209multiway_merge.h File Reference	3404
6.209.1 Detailed Description	3409
6.209.2 Define Documentation	3410
6.209.2.1 _GLIBCXX_PARALLEL_LENGTH	3410
6.210multiway_mergesort.h File Reference	3410
6.210.1 Detailed Description	3411
6.211mutex File Reference	3411
6.211.1 Detailed Description	3412
6.212nested_exception.h File Reference	3413
6.212.1 Detailed Description	3413
6.213new File Reference	3413
6.213.1 Detailed Description	3414
6.213.2 Function Documentation	3415
6.213.2.1 operator delete	3415
6.213.2.2 operator delete	3415
6.213.2.3 operator delete	3415
6.213.2.4 operator delete[]	3416
6.213.2.5 operator delete[]	3416
6.213.2.6 operator delete[]	3416
6.213.2.7 operator new	3417
6.213.2.8 operator new	3417
6.213.2.9 operator new	3417
6.213.2.10operator new[]	3418
6.213.2.11operator new[]	3418
6.213.2.12operator new[]	3419
6.214new_allocator.h File Reference	3419
6.214.1 Detailed Description	3420
6.215numeric File Reference	3420

6.215.1 Detailed Description	3420
6.216numeric File Reference	3420
6.216.1 Detailed Description	3420
6.217numeric File Reference	3421
6.217.1 Detailed Description	3424
6.218numeric_traits.h File Reference	3424
6.218.1 Detailed Description	3424
6.219numericfwd.h File Reference	3424
6.219.1 Detailed Description	3427
6.220omp_loop.h File Reference	3427
6.220.1 Detailed Description	3427
6.221omp_loop_static.h File Reference	3428
6.221.1 Detailed Description	3428
6.222os_defines.h File Reference	3428
6.222.1 Detailed Description	3428
6.223ostream File Reference	3429
6.223.1 Detailed Description	3430
6.224ostream.tcc File Reference	3430
6.224.1 Detailed Description	3431
6.225ostream_insert.h File Reference	3431
6.225.1 Detailed Description	3431
6.226par_loop.h File Reference	3431
6.226.1 Detailed Description	3432
6.227parallel.h File Reference	3432
6.227.1 Detailed Description	3432
6.228partial_sum.h File Reference	3432
6.228.1 Detailed Description	3433
6.229partition.h File Reference	3433
6.229.1 Detailed Description	3434
6.229.2 Define Documentation	3434
6.229.2.1 _GLIBCXX_VOLATILE	3434

6.230pod_char_traits.h File Reference	3434
6.230.1 Detailed Description	3435
6.231pointer.h File Reference	3435
6.231.1 Detailed Description	3437
6.232pool_allocator.h File Reference	3438
6.232.1 Detailed Description	3438
6.233postypes.h File Reference	3438
6.233.1 Detailed Description	3439
6.234priority_queue.hpp File Reference	3439
6.234.1 Detailed Description	3440
6.235priority_queue_base_dispatch.hpp File Reference	3440
6.235.1 Detailed Description	3440
6.236profiler.h File Reference	3440
6.236.1 Detailed Description	3443
6.237profiler_algos.h File Reference	3443
6.237.1 Detailed Description	3444
6.238profiler_hashtable_size.h File Reference	3444
6.238.1 Detailed Description	3445
6.239profiler_list_to_slist.h File Reference	3445
6.239.1 Detailed Description	3445
6.240profiler_list_to_vector.h File Reference	3445
6.240.1 Detailed Description	3446
6.241profiler_map_to_unordered_map.h File Reference	3446
6.241.1 Detailed Description	3447
6.242profiler_node.h File Reference	3448
6.242.1 Detailed Description	3448
6.243profiler_state.h File Reference	3449
6.243.1 Detailed Description	3449
6.244profiler_trace.h File Reference	3449
6.244.1 Detailed Description	3452
6.245profiler_vector_size.h File Reference	3452

6.245.1 Detailed Description	3452
6.246profiler_vector_to_list.h File Reference	3452
6.246.1 Detailed Description	3453
6.247queue File Reference	3453
6.247.1 Detailed Description	3453
6.248queue.h File Reference	3454
6.248.1 Detailed Description	3454
6.248.2 Define Documentation	3454
6.248.2.1 _GLIBCXX_VOLATILE	3454
6.249quicksort.h File Reference	3455
6.249.1 Detailed Description	3455
6.250random File Reference	3455
6.250.1 Detailed Description	3455
6.251random.h File Reference	3456
6.251.1 Detailed Description	3463
6.252random.tcc File Reference	3463
6.252.1 Detailed Description	3468
6.253random_number.h File Reference	3468
6.253.1 Detailed Description	3468
6.254random_shuffle.h File Reference	3468
6.254.1 Detailed Description	3470
6.255range_access.h File Reference	3470
6.255.1 Detailed Description	3470
6.256ratio File Reference	3470
6.256.1 Detailed Description	3471
6.257rb_tree File Reference	3471
6.257.1 Detailed Description	3472
6.258rc_string_base.h File Reference	3472
6.258.1 Detailed Description	3472
6.259regex File Reference	3472
6.259.1 Detailed Description	3472

6.260	regex.h File Reference	3472
6.260.1	Detailed Description	3478
6.261	regex_compiler.h File Reference	3479
6.261.1	Detailed Description	3479
6.262	regex_constants.h File Reference	3479
6.262.1	Detailed Description	3480
6.263	regex_cursor.h File Reference	3481
6.263.1	Detailed Description	3481
6.264	regex_error.h File Reference	3481
6.264.1	Detailed Description	3482
6.265	regex_grep_matcher.h File Reference	3482
6.265.1	Detailed Description	3483
6.266	regex_grep_matcher.tcc File Reference	3483
6.266.1	Detailed Description	3483
6.267	regex_nfa.h File Reference	3483
6.267.1	Detailed Description	3484
6.268	regex_nfa.tcc File Reference	3484
6.268.1	Detailed Description	3484
6.269	rope File Reference	3484
6.269.1	Detailed Description	3488
6.270	ropeimpl.h File Reference	3489
6.270.1	Detailed Description	3489
6.271	safe_base.h File Reference	3489
6.271.1	Detailed Description	3490
6.272	safe_iterator.h File Reference	3490
6.272.1	Detailed Description	3491
6.273	safe_iterator.tcc File Reference	3492
6.273.1	Detailed Description	3492
6.274	safe_sequence.h File Reference	3492
6.274.1	Detailed Description	3492
6.275	search.h File Reference	3492

6.275.1 Detailed Description	3493
6.276set File Reference	3493
6.276.1 Detailed Description	3493
6.277set File Reference	3493
6.277.1 Detailed Description	3493
6.278set File Reference	3493
6.278.1 Detailed Description	3493
6.279set.h File Reference	3494
6.279.1 Detailed Description	3494
6.280set.h File Reference	3495
6.280.1 Detailed Description	3495
6.281set_operations.h File Reference	3496
6.281.1 Detailed Description	3496
6.282settings.h File Reference	3497
6.282.1 Detailed Description	3497
6.282.2 parallelization_decision	3497
6.282.3 Define Documentation	3498
6.282.3.1 _GLIBCXX_PARALLEL_CONDITION	3498
6.283shared_ptr.h File Reference	3498
6.283.1 Detailed Description	3500
6.284shared_ptr_base.h File Reference	3500
6.284.1 Detailed Description	3502
6.285slice_array.h File Reference	3502
6.285.1 Detailed Description	3502
6.286slist File Reference	3503
6.286.1 Detailed Description	3504
6.287sort.h File Reference	3504
6.287.1 Detailed Description	3505
6.288sso_string_base.h File Reference	3505
6.288.1 Detailed Description	3505
6.289sstream File Reference	3505

6.289.1 Detailed Description	3506
6.290sstream.tcc File Reference	3506
6.290.1 Detailed Description	3506
6.291stack File Reference	3507
6.291.1 Detailed Description	3507
6.292standard_policies.hpp File Reference	3507
6.292.1 Detailed Description	3507
6.293stdatomic.h File Reference	3507
6.293.1 Detailed Description	3507
6.294stdexcept File Reference	3507
6.294.1 Detailed Description	3508
6.295stdio_filebuf.h File Reference	3508
6.295.1 Detailed Description	3508
6.296stdio_sync_filebuf.h File Reference	3509
6.296.1 Detailed Description	3509
6.297stl_algo.h File Reference	3509
6.297.1 Detailed Description	3522
6.298stl_algobase.h File Reference	3522
6.298.1 Detailed Description	3525
6.299stl_bvector.h File Reference	3525
6.299.1 Detailed Description	3526
6.300stl_construct.h File Reference	3526
6.300.1 Detailed Description	3527
6.301stl_deque.h File Reference	3527
6.301.1 Detailed Description	3530
6.301.2 Define Documentation	3530
6.301.2.1 _GLIBCXX_DEQUE_BUF_SIZE	3530
6.302stl_function.h File Reference	3531
6.302.1 Detailed Description	3533
6.303stl_heap.h File Reference	3533
6.303.1 Detailed Description	3535

6.304stl_iterator.h File Reference	3535
6.304.1 Detailed Description	3539
6.305stl_iterator_base_funcs.h File Reference	3539
6.305.1 Detailed Description	3540
6.306stl_iterator_base_types.h File Reference	3540
6.306.1 Detailed Description	3541
6.307stl_list.h File Reference	3542
6.307.1 Detailed Description	3543
6.308stl_map.h File Reference	3543
6.308.1 Detailed Description	3544
6.309stl_multimap.h File Reference	3544
6.309.1 Detailed Description	3545
6.310stl_multiset.h File Reference	3545
6.310.1 Detailed Description	3546
6.311stl_numeric.h File Reference	3546
6.311.1 Detailed Description	3547
6.312stl_pair.h File Reference	3547
6.312.1 Detailed Description	3548
6.313stl_queue.h File Reference	3549
6.313.1 Detailed Description	3550
6.314stl_raw_storage_iter.h File Reference	3550
6.314.1 Detailed Description	3550
6.315stl_relops.h File Reference	3550
6.315.1 Detailed Description	3551
6.316stl_set.h File Reference	3551
6.316.1 Detailed Description	3552
6.317stl_stack.h File Reference	3552
6.317.1 Detailed Description	3553
6.318stl_tempbuf.h File Reference	3553
6.318.1 Detailed Description	3554
6.319stl_tree.h File Reference	3554

6.319.1 Detailed Description	3555
6.320stl_uninitialized.h File Reference	3555
6.320.1 Detailed Description	3557
6.321stl_vector.h File Reference	3557
6.321.1 Detailed Description	3558
6.322stream_iterator.h File Reference	3559
6.322.1 Detailed Description	3559
6.323streambuf File Reference	3559
6.323.1 Detailed Description	3560
6.324streambuf.tcc File Reference	3560
6.324.1 Detailed Description	3560
6.325streambuf_iterator.h File Reference	3561
6.325.1 Detailed Description	3562
6.326string File Reference	3562
6.326.1 Detailed Description	3562
6.327string File Reference	3562
6.327.1 Detailed Description	3565
6.328stringfwd.h File Reference	3565
6.328.1 Detailed Description	3565
6.329system_error File Reference	3565
6.329.1 Detailed Description	3567
6.330tag_and_trait.hpp File Reference	3567
6.330.1 Detailed Description	3569
6.331tags.h File Reference	3569
6.331.1 Detailed Description	3571
6.332tgmth.h File Reference	3571
6.332.1 Detailed Description	3571
6.333thread File Reference	3571
6.333.1 Detailed Description	3572
6.334throw_allocator.h File Reference	3572
6.334.1 Detailed Description	3575

6.335time_members.h File Reference	3575
6.335.1 Detailed Description	3575
6.336tree_policy.hpp File Reference	3575
6.336.1 Detailed Description	3576
6.337tree_trace_base.hpp File Reference	3576
6.337.1 Detailed Description	3576
6.338trie_policy.hpp File Reference	3576
6.338.1 Detailed Description	3576
6.339tuple File Reference	3576
6.339.1 Detailed Description	3579
6.340type_traits File Reference	3579
6.340.1 Detailed Description	3582
6.341type_traits File Reference	3582
6.341.1 Detailed Description	3585
6.342type_traits.h File Reference	3585
6.342.1 Detailed Description	3586
6.343type_utils.hpp File Reference	3586
6.343.1 Detailed Description	3586
6.344typeinfo File Reference	3586
6.344.1 Detailed Description	3587
6.345typelist.h File Reference	3587
6.345.1 Detailed Description	3588
6.346types.h File Reference	3588
6.346.1 Detailed Description	3589
6.347types_traits.hpp File Reference	3589
6.347.1 Detailed Description	3590
6.348unique_copy.h File Reference	3590
6.348.1 Detailed Description	3590
6.349unique_ptr.h File Reference	3590
6.349.1 Detailed Description	3592
6.350unordered_map File Reference	3592

6.350.1 Detailed Description	3592
6.351unordered_map File Reference	3592
6.351.1 Detailed Description	3593
6.352unordered_map File Reference	3593
6.352.1 Detailed Description	3595
6.353unordered_map.h File Reference	3595
6.353.1 Detailed Description	3596
6.354unordered_set File Reference	3597
6.354.1 Detailed Description	3597
6.355unordered_set File Reference	3597
6.355.1 Detailed Description	3598
6.356unordered_set File Reference	3598
6.356.1 Detailed Description	3599
6.357unordered_set.h File Reference	3599
6.357.1 Detailed Description	3601
6.358utility File Reference	3601
6.358.1 Detailed Description	3601
6.359utility File Reference	3601
6.359.1 Detailed Description	3602
6.360valarray File Reference	3602
6.360.1 Detailed Description	3607
6.361valarray_after.h File Reference	3608
6.361.1 Detailed Description	3622
6.362valarray_array.h File Reference	3622
6.362.1 Detailed Description	3632
6.363valarray_array.tcc File Reference	3633
6.363.1 Detailed Description	3634
6.364valarray_before.h File Reference	3634
6.364.1 Detailed Description	3634
6.365vector File Reference	3634
6.365.1 Detailed Description	3634

6.366vector File Reference	3634
6.366.1 Detailed Description	3635
6.367vector File Reference	3635
6.367.1 Detailed Description	3636
6.368vector.tcc File Reference	3637
6.368.1 Detailed Description	3637
6.369vstring.h File Reference	3637
6.369.1 Detailed Description	3641
6.370vstring.tcc File Reference	3641
6.370.1 Detailed Description	3642
6.371vstring_fwd.h File Reference	3642
6.371.1 Detailed Description	3643
6.372vstring_util.h File Reference	3643
6.372.1 Detailed Description	3643
6.373workstealing.h File Reference	3643
6.373.1 Detailed Description	3644

Chapter 1

Todo List

Member [__glibcxx_check_insert_range](#)([_Position](#), [_First](#), [_Last](#)) We would like to be able to check for noninterference of [_Position](#) and the range [[_First](#), [_Last](#)), but that can't (in general) be done.

Member [__glibcxx_check_insert_range_after](#)([_Position](#), [_First](#), [_Last](#)) We would like to be able to check for noninterference of [_Position](#) and the range [[_First](#), [_Last](#)), but that can't (in general) be done.

Member [__gnu_cxx::distance](#)([_InputIterator](#) [__first](#), [_InputIterator](#) [__last](#), [_Distance](#) & [__n](#))
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Class [__gnu_cxx::hash_map](#)< [_Key](#), [_Tp](#), [_HashFn](#), [_EqualKey](#), [_Alloc](#) >
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Class [__gnu_cxx::hash_multimap](#)< [_Key](#), [_Tp](#), [_HashFn](#), [_EqualKey](#), [_Alloc](#) >
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Class [__gnu_cxx::hash_multiset](#)< [_Value](#), [_HashFn](#), [_EqualKey](#), [_Alloc](#) >
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Class [__gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc >](#)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::iota](#)(_ForwardIter __first, _ForwardIter __last, _Tp __value)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::is_heap](#)(_RandomAccessIterator __first, _RandomAccessIterator __last)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::is_heap](#)(_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::is_sorted](#)(_ForwardIterator __first, _ForwardIterator __last)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::is_sorted](#)(_ForwardIterator __first, _ForwardIterator __last, _StrictWeakOrdering __comp)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::power](#)(_Tp __x, _Integer __n, _MonoidOperation __monoid_op)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::power](#)(_Tp __x, _Integer __n)

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample](#)([_InputIterator __first](#), [_InputIterator __last](#), [_RandomAccessIterator __out_first](#), [_RandomAccessIterator __out_last](#))
 Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample](#)([_InputIterator __first](#), [_InputIterator __last](#), [_RandomAccessIterator __out_first](#), [_RandomAccessIterator __out_last](#))
 Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample_n](#)([_ForwardIterator __first](#), [_ForwardIterator __last](#), [_OutputIterator __out](#), [_RandomAccessIterator __out_last](#), [_RandomAccessIterator __out_first](#))
 Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_cxx::random_sample_n](#)([_ForwardIterator __first](#), [_ForwardIterator __last](#), [_OutputIterator __out](#), [_RandomAccessIterator __out_last](#), [_RandomAccessIterator __out_first](#))
 Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::rb_tree](#)<[_Key](#), [_Value](#), [_KeyOfValue](#), [_Compare](#), [_Alloc](#)>
 Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::rope](#)<[_CharT](#), [_Alloc](#)> Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Class [__gnu_cxx::slist](#)<[_Tp](#), [_Alloc](#)> Doc me! See [doc/doxygen/TODO](#) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member [__gnu_debug::Safe_iterator](#)<[_Iterator](#), [_Sequence](#)>::[operator->\(\)](#) const
 Make this correct w.r.t. iterators that return proxies
 Use [addressof\(\)](#) instead of [&](#) operator

Group **Constants** These should be constexpr.

Class **std::basic_string<_CharT, _Traits, _Alloc>**

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard(unsigned long long __z)**

Look for a faster way to do discard.

Member **std::discard_block_engine<_RandomNumberEngine, __p, __r>::max() const**

This should be constexpr.

Member **std::discard_block_engine<_RandomNumberEngine, __p, __r>::min() const**

This should be constexpr.

Member **std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::discard(unsigned long long __z)**

Look for a faster way to do discard.

Member **std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::max() const**

This should be constexpr.

Member **std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::min() const**

This should be constexpr.

Member **std::linear_congruential_engine<_UIntType, __a, __c, __m>::discard(unsigned long long __z)**

Look for a faster way to do discard.

Member **std::linear_congruential_engine<_UIntType, __a, __c, __m>::max() const**

This should be constexpr.

Member **std::linear_congruential_engine<_UIntType, __a, __c, __m>::min() const**

This should be constexpr.

Member **std::operator==(const match_results<_Bi_iter, _Allocator> &__m1, const match_results<_Bi_iter, _Allocator> &__m2)**
Implement this function.

Doc me! See [doc/doxygen/TODO](http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html) and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator=(const regex_iterator &__rhs)`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator==(const regex_iterator &__rhs)`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator()`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator(_Bi_iter __a, _Bi_iter __b,`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator(const regex_iterator &__r,`**

Implement this function.

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member **`std::regex_match(_Bi_iter __s, _Bi_iter __e, match_results<_Bi_iter, _Allocator> &__m, const`**

Implement this function.

Member **`std::regex_replace(const basic_string<_Ch_type> &__s, const basic_regex<_Ch_type, _Rx_traits>`**

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_replace`(`_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Implement this function.

Implement this function.

Member `std::regex_search`(`_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_c`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_search`(`_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Allocator > &__m, const basic_r`

Implement this function.

Member `std::regex_search`(`const _Ch_type *__s, match_results< const _Ch_type *, _Allocator > &__m, const basic_re`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_search`(`const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _C`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_search`(`const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::m`

Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!=(const regex_token_iterator &__rhs)`

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator*()`

Implement this function.

Member `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++()`

Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator++(int)`
 Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator->()`
 Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator=(const regex_token_itera`
 Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator==(const regex_token_iter`
 Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator(_Bi_iter __a,`
 Implement this function.
 Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator(_Bi_iter __a,`
 Implement this function.
 Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator(_Bi_iter __a,`
 Implement this function.
 Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
 for more.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator()`
 Implement this function.

Member `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator(const regex_t`
 Implement this function.

Member **`std::regex_traits<_Ch_type>::lookup_classname`**(_Fwd_iter __first, _Fwd_iter __last, bool __icase=false) const
Implement this function.

Member **`std::regex_traits<_Ch_type>::lookup_collatename`**(_Fwd_iter __first, _Fwd_iter __last) const
Implement this function.

Member **`std::regex_traits<_Ch_type>::transform_primary`**(_Fwd_iter __first, _Fwd_iter __last) const
Implement this function.

Member **`std::reverse_iterator<_Iterator>::operator*() const`**
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::reverse_iterator<_Iterator>::operator+(difference_type __n) const`**
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::reverse_iterator<_Iterator>::operator++()`**
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::reverse_iterator<_Iterator>::operator++(int)`**
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::reverse_iterator<_Iterator>::operator+=(difference_type __n)`**
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member **`std::reverse_iterator<_Iterator>::operator-(difference_type __n) const`**
Doc me! See doc/doxygen/TODO and
<http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html>
for more.

Member `std::reverse_iterator<_Iterator>::operator--(int)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator--()`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator--(difference_type __n)`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator->() const`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::reverse_iterator<_Iterator>::operator[](difference_type __n) const`

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::discard(unsigned long long __z)`

Look for a faster way to do discard.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::max() const`

This should be constexpr.

Member `std::shuffle_order_engine<_RandomNumberEngine, __k>::min() const`

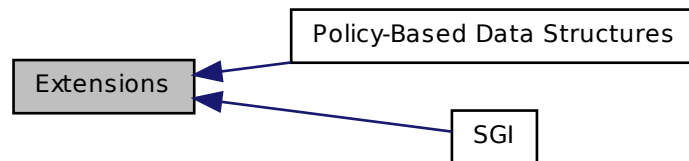
This should be constexpr.

Chapter 2

Module Documentation

2.1 Extensions

Collaboration diagram for Extensions:



Classes

- class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`
Template class `__versa_string`.
Data structure managing sequences of characters and character-like objects.

Modules

- [SGI](#)

- [Policy-Based Data Structures](#)

2.1.1 Detailed Description

Components generally useful that are not part of any standard.

2.2 SGI

Collaboration diagram for SGI:



Classes

- class `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`
An SGI extension .
- struct `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`
An SGI extension .
- struct `__gnu_cxx::constant_unary_fun< _Result, _Argument >`
An SGI extension .
- struct `__gnu_cxx::constant_void_fun< _Result >`
An SGI extension .
- class `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- struct `__gnu_cxx::project1st< _Arg1, _Arg2 >`
An SGI extension .

- struct `__gnu_cxx::project2nd< _Arg1, _Arg2 >`
An SGI extension .
- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope< _CharT, _Alloc >`
- struct `__gnu_cxx::select1st< _Pair >`
An SGI extension .
- struct `__gnu_cxx::select2nd< _Pair >`
An SGI extension .
- class `__gnu_cxx::slist< _Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`
An SGI extension .

Functions

- template<typename _Tp >
const _Tp & `__gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare >
const _Tp & `__gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- size_t `std::bitset::_Find_first` () const
- size_t `std::bitset::_Find_next` (size_t __prev) const
- template<class _Operation1, class _Operation2 >
unary_compose< _Operation1, _Operation2 > `__gnu_cxx::compose1` (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template<class _Operation1, class _Operation2, class _Operation3 >
binary_compose< _Operation1, _Operation2, _Operation3 > `__gnu_cxx::compose2` (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result >
constant_void_fun< _Result > `__gnu_cxx::constant0` (const _Result &__val)
- template<class _Result >
constant_unary_fun< _Result, _Result > `__gnu_cxx::constant1` (const _Result &__val)
- template<class _Result >
constant_binary_fun< _Result, _Result, _Result > `__gnu_cxx::constant2` (const _Result &__val)

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __`
`first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Distance >`
`void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _`
`Distance &__n)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
- `template<typename _ForwardIter, typename _Tp >`
`void __gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _`
`RandomAccessIterator __last, _StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`
`bool __gnu_cxx::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _`
`StrictWeakOrdering __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp, typename _Integer >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid-`
`op)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _`
`RandomNumberGenerator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __`
`first, _InputIterator __last, _RandomAccessIterator __out_first, _`
`RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __`
`first, _InputIterator __last, _RandomAccessIterator __out_first, _`
`RandomAccessIterator __out_last)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _`
`ForwardIterator __last, _OutputIterator __out, const _Distance __n)`

- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _-`
`ForwardIterator __last, _OutputIterator __out, const _Distance __n, _-`
`RandomNumberGenerator &__rand)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter`
`__first, _Size __count, _ForwardIter __result)`
- `bitset< _Nb > & std::bitset::_Unchecked_set (size_t __pos)`
- `bitset< _Nb > & std::bitset::_Unchecked_set (size_t __pos, int __val)`
- `bitset< _Nb > & std::bitset::_Unchecked_reset (size_t __pos)`
- `bitset< _Nb > & std::bitset::_Unchecked_flip (size_t __pos)`
- `bool std::bitset::_Unchecked_test (size_t __pos) const`

2.2.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functions and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int templ = g1(x);
int temp2 = g2(x);
int answer = f(templ,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

2.2.2 Function Documentation

2.2.2.1 `template<typename _Tp> const _Tp& __gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp & __c)`

Find the median of three values.

Parameters

- a* A value.
- b* A value.
- c* A value.

Returns

One of *a*, *b* or *c*.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that $1 \leq m \leq n$ then the value returned will be m . This is an SGI extension.

Definition at line 537 of file ext/algorithm.

2.2.2.2 `template<typename _Tp, typename _Compare> const _Tp&
__gnu_cxx::__median (const _Tp & __a, const _Tp & __b, const _Tp
& __c, _Compare __comp)`

Find the median of three values using a predicate for comparison.

Parameters

a A value.
b A value.
c A value.
comp A binary predicate.

Returns

One of *a*, *b* or *c*.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that `comp(1, m)` and `comp(m, n)` are both true then the value returned will be m . This is an SGI extension.

Definition at line 571 of file ext/algorithm.

2.2.2.3 `template<size_t _Nb> size_t std::bitset<_Nb>::__Find_first () const
[inline, inherited]`

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or `size()` if not found.

See also

`_Find_next`

Definition at line 1303 of file bitset.

2.2.2.4 `template<size_t _Nb> size_t std::bitset<_Nb>::__Find_next (size_t
__prev) const [inline, inherited]`

Finds the index of the next "on" bit after *prev*.

Returns

The index of the next bit set, or `size()` if not found.

Parameters

prev Where to start searching.

See also

`_Find_first`

Definition at line 1314 of file `bitset`.

2.2.2.5 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::_Unchecked_flip (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 986 of file `bitset`.

2.2.2.6 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::_Unchecked_reset (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 979 of file `bitset`.

2.2.2.7 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::_Unchecked_set (size_t __pos) [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 962 of file `bitset`.

2.2.2.8 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::_Unchecked_set (size_t __pos, int __val) [inline,
inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 969 of file `bitset`.

2.2.2.9 `template<size_t _Nb> bool std::bitset< _Nb >::_Unchecked_test (`
`size_t __pos) const [inline, inherited]`

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 993 of file `bitset`.

2.2.2.10 `template<class _Operation1 , class _Operation2 >`
`unary_compose<_Operation1, _Operation2> __gnu_cxx::compose1`
`(const _Operation1 & __fn1, const _Operation2 & __fn2)`
`[inline]`

An [SGI extension](#) .

Definition at line 144 of file `ext/functional`.

2.2.2.11 `template<class _Operation1 , class _Operation2 , class _Operation3`
`> binary_compose<_Operation1, _Operation2, _Operation3>`
`__gnu_cxx::compose2 (const _Operation1 & __fn1, const`
`_Operation2 & __fn2, const _Operation3 & __fn3) [inline]`

An [SGI extension](#) .

Definition at line 171 of file `ext/functional`.

2.2.2.12 `template<class _Result > constant_void_fun<_Result>`
`__gnu_cxx::constant0 (const _Result & __val) [inline]`

An [SGI extension](#) .

Definition at line 325 of file `ext/functional`.

2.2.2.13 `template<class _Result > constant_unary_fun<_Result, _Result>`
`__gnu_cxx::constant1 (const _Result & __val) [inline]`

An [SGI extension](#) .

Definition at line 331 of file `ext/functional`.

2.2.2.14 `template<class _Result > constant_binary_fun<_Result, _-`
`Result, _Result> __gnu_cxx::constant2 (const _Result & __val)`
`[inline]`

An [SGI extension](#) .

Definition at line 337 of file ext/functional.

```
2.2.2.15  template<typename _InputIterator , typename _Size , typename
            _OutputIterator > pair<_InputIterator, _OutputIterator>
            __gnu_cxx::copy_n ( _InputIterator __first, _Size __count,
            _OutputIterator __result ) [inline]
```

Copies the range [first,first+count) into [result,result+count).

Parameters

first An input iterator.

count The number of elements to copy.

result An output iterator.

Returns

A [std::pair](#) composed of first+count and result+count.

This is an SGI extension. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 119 of file ext/algorithm.

References `std::__iterator_category()`.

```
2.2.2.16  template<typename _InputIterator , typename _Distance > void
            __gnu_cxx::distance ( _InputIterator __first, _InputIterator __last,
            _Distance & __n ) [inline]
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 103 of file ext/iterator.

References `std::__iterator_category()`.

Referenced by `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, and `std::list<_Tp, _Alloc >::size()`.

2.2.2.17 `template<class _Tp > _Tp __gnu_cxx::identity_element (std::plus<_Tp >) [inline]`

An [SGI extension](#) .

Definition at line 86 of file ext/functional.

2.2.2.18 `template<class _Tp > _Tp __gnu_cxx::identity_element (std::multiplies<_Tp >) [inline]`

An [SGI extension](#) .

Definition at line 92 of file ext/functional.

2.2.2.19 `template<typename _ForwardIter , typename _Tp > void __gnu_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 132 of file ext/numeric.

2.2.2.20 `template<typename _RandomAccessIterator , typename _StrictWeakOrdering > bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp) [inline]`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 453 of file ext/algorithm.

2.2.2.21 `template<typename _RandomAccessIterator > bool __gnu_cxx::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 434 of file ext/algorithm.

2.2.2.22 `template<typename _ForwardIterator > bool __gnu_cxx::is_sorted (`
`_FForwardIterator __first, _ForwardIterator __last)`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 478 of file ext/algorithm.

2.2.2.23 `template<typename _ForwardIterator , typename`
`_StrictWeakOrdering > bool __gnu_cxx::is_sorted (`
`_FForwardIterator __first, _ForwardIterator __last,`
`_StrictWeakOrdering __comp)`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 503 of file ext/algorithm.

2.2.2.24 `template<typename _InputIterator1 , typename _InputIterator2 >`
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1`
`__first1, _InputIterator1 __last1, _InputIterator2 __first2,`
`_InputIterator2 __last2)`

memcmp on steroids.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

last2 An input iterator.

Returns

An int, as with `memcmp`.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

Definition at line 200 of file `ext/algorithm`.

2.2.2.25 `template<typename _Tp , typename _Integer > _Tp
__gnu_cxx::power (_Tp __x, _Integer __n) [inline]`

This is an SGI extension.

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 121 of file `ext/numeric`.

2.2.2.26 `template<typename _Tp , typename _Integer , typename
_MonoidOperation > _Tp __gnu_cxx::power (_Tp __x, _Integer
__n, _MonoidOperation __monoid_op) [inline]`

This is an SGI extension.

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 111 of file `ext/numeric`.

2.2.2.27 `template<typename _InputIterator , typename
_RandomAccessIterator , typename _RandomNumberGenerator >
_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator
__first, _InputIterator __last, _RandomAccessIterator __out_first,
_RandomAccessIterator __out_last, _RandomNumberGenerator &
__rand) [inline]`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 410 of file ext/algorithm.

```
2.2.2.28  template<typename _InputIterator , typename
           _RandomAccessIterator > _RandomAccessIterator
           __gnu_cxx::random_sample ( _InputIterator __first, _InputIterator
           __last, _RandomAccessIterator __out_first, _RandomAccessIterator
           __out_last ) [inline]
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 387 of file ext/algorithm.

```
2.2.2.29  template<typename _ForwardIterator , typename _OutputIterator ,
           typename _Distance , typename _RandomNumberGenerator >
           _OutputIterator __gnu_cxx::random_sample_n ( _ForwardIterator
           __first, _ForwardIterator __last, _OutputIterator __out, const
           _Distance __n, _RandomNumberGenerator & __rand )
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 300 of file ext/algorithm.

References `std::distance()`, and `std::min()`.

```
2.2.2.30  template<typename _ForwardIterator , typename _OutputIterator ,
           typename _Distance > _OutputIterator __gnu_cxx::random_sample_n
           ( _ForwardIterator __first, _ForwardIterator __last,
           _OutputIterator __out, const _Distance __n )
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 266 of file ext/algorithm.

References `std::distance()`, and `std::min()`.

```
2.2.2.31  template<typename _InputIter, typename _Size,
              typename _ForwardIter > pair<_InputIter, _ForwardIter>
              __gnu_cxx::uninitialized_copy_n( _InputIter __first, _Size __count,
              _ForwardIter __result ) [inline]
```

Copies the range `[first,last)` into `result`.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

`result + (first - last)`

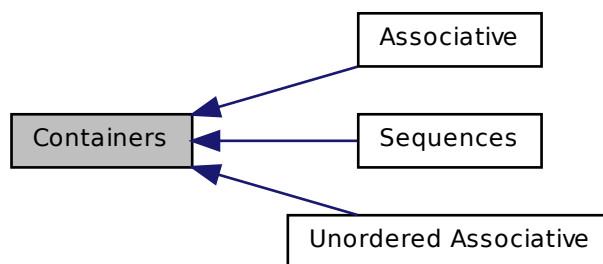
Like `copy()`, but does not require an initialized output range.

Definition at line 121 of file ext/memory.

References `std::__iterator_category()`.

2.3 Containers

Collaboration diagram for Containers:



Classes

- class `std::bitset<_Nb>`

The `bitset` class represents a fixed-size sequence of bits.

Modules

- [Sequences](#)
- [Associative](#)
- [Unordered Associative](#)

2.3.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in [tables](#).

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

2.4 Sequences

Collaboration diagram for Sequences:



Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
Managing sequences of characters and character-like objects.
- class `std::deque<_Tp, _Alloc>`
A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.
- class `std::forward_list<_Tp, _Alloc>`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class `std::list<_Tp, _Alloc>`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

- class `std::priority_queue< _Tp, _Sequence, _Compare >`

A standard container automatically sorting its contents.

- class `std::queue< _Tp, _Sequence >`

A standard container giving FIFO behavior.

- class `std::stack< _Tp, _Sequence >`

A standard container giving FILO behavior.

- struct `std::tr1::array< _Tp, _Nm >`

A standard container for storing a fixed size sequence of elements.

- class `std::vector< _Tp, _Alloc >`

A standard container which offers fixed time access to individual elements in any order.

- class `std::vector< bool, _Alloc >`

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

2.4.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

2.5 Associative

Collaboration diagram for Associative:



Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class `std::multiset< _Key, _Compare, _Alloc >`
A standard container made up of elements, which can be retrieved in logarithmic time.
- class `std::set< _Key, _Compare, _Alloc >`
A standard container made up of unique keys, which can be retrieved in logarithmic time.

2.5.1 Detailed Description

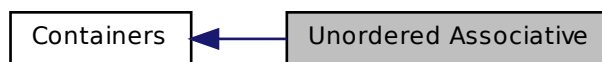
Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

2.6 Unordered Associative

Collaboration diagram for Unordered Associative:



Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`
A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

2.6.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

2.7 Diagnostics

Collaboration diagram for Diagnostics:



Modules

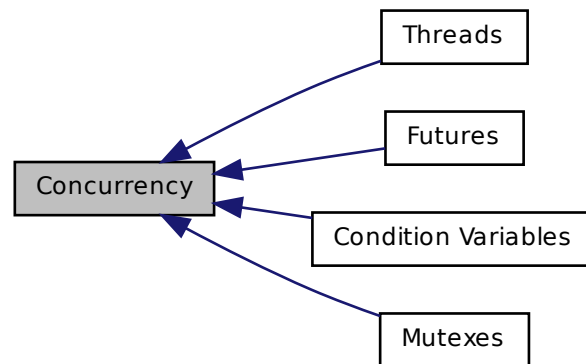
- [Exceptions](#)

2.7.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

2.8 Concurrency

Collaboration diagram for Concurrency:



Modules

- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

2.8.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

2.9 Exceptions

Collaboration diagram for Exceptions:



Classes

- class `__cxxabiv1::__forced_unwind`
*Thrown as part of forced unwinding.
 A magic placeholder class that can be caught by reference to recognize forced unwinding.*
- struct `__gnu_cxx::forced_error`
Thrown by exception safety machinery.
- class `__gnu_cxx::recursive_init_error`
*Exception thrown by `__cxa_guard_acquire`.
 6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*
- class `std::__exception_ptr::exception_ptr`
An opaque pointer to an arbitrary exception.
- class `std::bad_alloc`
*Exception possibly thrown by `new`.
`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*
- class `std::bad_cast`
*Thrown during incorrect typecasting.
 If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class `std::bad_exception`
- class `std::bad_function_call`

Exception class thrown when class template function's operator() is called with an empty target.

- class [std::bad_typeid](#)

Thrown when a NULL pointer in a typeid expression is used.

- class [std::domain_error](#)
- class [std::exception](#)

Base class for all library exceptions.

- class [std::future_error](#)

Exception type thrown by futures.

- class [std::invalid_argument](#)
- class [std::ios_base::failure](#)

*These are thrown to indicate problems with io.
27.4.2.1.1 Class [ios_base::failure](#).*

- class [std::length_error](#)
- class [std::logic_error](#)

One of two subclasses of exception.

- class [std::nested_exception](#)

Exception class with exception_ptr data member.

- class [std::out_of_range](#)
- class [std::overflow_error](#)
- class [std::range_error](#)
- class [std::regex_error](#)

*A regular expression exception class.
The regular expression library throws objects of this class on error.*

- class [std::runtime_error](#)

One of two subclasses of exception.

- class [std::system_error](#)

Thrown to indicate error code of underlying system.

- class [std::tr1::bad_weak_ptr](#)

Exception possibly thrown by [shared_ptr](#).

- class [std::underflow_error](#)

Typedefs

- typedef void(* [std::terminate_handler](#))()
- typedef void(* [std::unexpected_handler](#))()

Functions

- template<typename _Ex >
const nested_exception * [std::__get_nested_exception](#) (const _Ex &__ex)
- template<typename _Ex >
void [std::__throw_with_nested](#) (_Ex &&, const nested_exception * = 0) [__attribute__\(\(__noreturn__\)\)](#)
- template<typename _Ex >
void [std::__throw_with_nested](#) (_Ex &&, ...) [__attribute__\(\(__noreturn__\)\)](#)
- void [__gnu_cxx::__verbose_terminate_handler](#) ()
- template<typename _Ex >
exception_ptr [std::copy_exception](#) (_Ex __ex) throw ()
- exception_ptr [std::current_exception](#) () throw ()
- template<typename _Ex >
exception_ptr [std::make_exception_ptr](#) (_Ex __ex) throw ()
- void [std::rethrow_exception](#) (exception_ptr) [__attribute__\(\(__noreturn__\)\)](#)
- void [std::rethrow_if_nested](#) (const nested_exception &__ex)
- template<typename _Ex >
void [std::rethrow_if_nested](#) (const _Ex &__ex)
- terminate_handler [std::set_terminate](#) (terminate_handler) throw ()
- unexpected_handler [std::set_unexpected](#) (unexpected_handler) throw ()
- void [std::terminate](#) () [__attribute__\(\(__noreturn__\)\)](#) throw ()
- template<typename _Ex >
void [std::throw_with_nested](#) (_Ex __ex)
- bool [std::uncaught_exception](#) () [__attribute__\(\(__pure__\)\)](#) throw ()
- void [std::unexpected](#) () [__attribute__\(\(__noreturn__\)\)](#)

2.9.1 Detailed Description

Classes and functions for reporting errors via exception classes.

2.9.2 Typedef Documentation

2.9.2.1 typedef void(* [std::terminate_handler](#))()

If you write a replacement terminate handler, it must be of this type.

Definition at line 88 of file exception.

2.9.2.2 `typedef void(* std::unexpected_handler)()`

If you write a replacement unexpected handler, it must be of this type.

Definition at line 91 of file `exception`.

2.9.3 Function Documentation

2.9.3.1 `void __gnu_cxx::__verbose_terminate_handler ()`

A replacement for the standard `terminate_handler` which prints more information about the terminating exception (if any) on `stderr`.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06>

In 3.4 and later, this is on by default.

2.9.3.2 `template<typename _Ex > exception_ptr std::copy_exception (_Ex __ex) throw ()`

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 162 of file `exception_ptr.h`.

References `std::current_exception()`.

2.9.3.3 `exception_ptr std::current_exception () throw ()`

Obtain an `exception_ptr` to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

Referenced by `std::copy_exception()`.

2.9.3.4 `template<typename _Ex > exception_ptr std::make_exception_ptr (_Ex __ex) throw ()`

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 181 of file `exception_ptr.h`.

2.9.3.5 void std::rethrow_exception (exception_ptr)

Throw the object pointed to by the exception_ptr.

Referenced by std::__basic_future<_Res & >::_M_get_result().

**2.9.3.6 void std::rethrow_if_nested (const nested_exception & __ex)
[inline]**

Overload, See N2619.

Definition at line 156 of file nested_exception.h.

2.9.3.7 template<typename _Ex > void std::rethrow_if_nested (const _Ex & __ex) [inline]

If __ex is derived from [nested_exception](#), __ex.rethrow_nested().

Definition at line 148 of file nested_exception.h.

2.9.3.8 terminate_handler std::set_terminate (terminate_handler) throw ()

Takes a new handler function as an argument, returns the old function.

**2.9.3.9 unexpected_handler std::set_unexpected (unexpected_handler)
throw ()**

Takes a new handler function as an argument, returns the old function.

2.9.3.10 void std::terminate () throw ()

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

**2.9.3.11 template<typename _Ex > void std::throw_with_nested (_Ex __ex)
[inline]**

If __ex is derived from [nested_exception](#), __ex. // Else, an implementation-defined object derived from both.

Definition at line 138 of file nested_exception.h.

2.9.3.12 `bool std::uncaught_exception () throw ()`

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of `terminate()` (15.5.1).'

Referenced by `std::basic_ostream<_CharT, _Traits>::sentry::~~sentry()`.

2.9.3.13 `void std::unexpected ()`

The runtime will call this function if an exception is thrown which violates the function's exception specification.

2.10 Time

Collaboration diagram for Time:



Namespaces

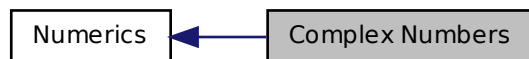
- namespace `std::chrono`

2.10.1 Detailed Description

Classes and functions for time.

2.11 Complex Numbers

Collaboration diagram for Complex Numbers:



Classes

- struct `std::complex< _Tp >`

Functions

- `std::complex< float >::complex` (const `complex< double >` &)
- `std::complex< float >::complex` (const `complex< long double >` &)
- `std::complex< double >::complex` (const `complex< long double >` &)
- `template<typename _Tp >`
`_Tp std::__complex_abs` (const `complex< _Tp >` &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_acos` (const `std::complex< _Tp >` &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_acosh` (const `std::complex< _Tp >` &__z)
- `template<typename _Tp >`
`_Tp std::__complex_arg` (const `complex< _Tp >` &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_asin` (const `std::complex< _Tp >` &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_asinh` (const `std::complex< _Tp >` &__z)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_atan` (const `std::complex< _Tp >` &__z)

- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::__complex_atanh (const std::complex< _Tp >`
`&__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const`
`complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::arg (_Tp __x)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::conj (const complex< _Tp > &)`

- `template<typename _Tp >`
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::imag (_Tp)`
- `template<typename _Tp >`
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::norm (_Tp __x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator*= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator*= (const _Tp &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator+= (const complex< _Up > &)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator-= (const complex< _Up > &)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator/= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator/= (const _Tp &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Up >`
`complex< _Tp > & std::complex::operator= (const complex< _Up > &)`
- `complex< _Tp > & std::complex::operator= (const _Tp &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`
`_Traits > &__is, complex< _Tp > &__x)`

- `template<typename _Tp >`
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`
`_Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::real (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::tanh (const complex< _Tp > &)`

- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

2.11.1 Detailed Description

Classes and functions for complex numbers.

2.11.2 Function Documentation

2.11.2.1 `template<typename _Tp> _Tp std::abs (const complex< _Tp> & __z) [inline]`

Return magnitude of z .

Definition at line 594 of file `complex`.

Referenced by `std::tr1::fabs()`, `std::binomial_distribution< _IntType>::operator()()`, and `std::poisson_distribution< _IntType>::operator()()`.

2.11.2.2 `template<typename _Tp> std::complex< _Tp> std::tr1::acos (const std::complex< _Tp> & __z) [inline]`

`acos(__z)` [8.1.2].

Definition at line 86 of file `tr1_impl/complex`.

2.11.2.3 `template<typename _Tp> std::complex< _Tp> std::tr1::acosh (const std::complex< _Tp> & __z) [inline]`

`acosh(__z)` [8.1.5].

Definition at line 205 of file `tr1_impl/complex`.

2.11.2.4 `template<typename _Tp> __gnu_cxx::__promote< _Tp>::__type std::tr1::arg (_Tp __x) [inline]`

Additional overloads [8.1.9].

Definition at line 311 of file `tr1_impl/complex`.

References `std::arg()`.

2.11.2.5 `template<typename _Tp> _Tp std::arg (const complex< _Tp> & __z) [inline]`

Return phase angle of z .

Definition at line 621 of file `complex`.

Referenced by `std::tr1::arg()`.

2.11.2.6 `template<typename _Tp > std::complex< _Tp > std::tr1::asin (`
`const std::complex< _Tp > & __z) [inline]`

`asin(__z)` [8.1.3].

Definition at line 122 of file `tr1_impl/complex`.

2.11.2.7 `template<typename _Tp > std::complex< _Tp > std::tr1::asinh (`
`const std::complex< _Tp > & __z) [inline]`

`asinh(__z)` [8.1.6].

Definition at line 244 of file `tr1_impl/complex`.

2.11.2.8 `template<typename _Tp > std::complex< _Tp > std::tr1::atan (`
`const std::complex< _Tp > & __z) [inline]`

`atan(__z)` [8.1.4].

Definition at line 166 of file `tr1_impl/complex`.

2.11.2.9 `template<typename _Tp > std::complex< _Tp > std::tr1::atanh (`
`const std::complex< _Tp > & __z) [inline]`

`atanh(__z)` [8.1.7].

Definition at line 288 of file `tr1_impl/complex`.

2.11.2.10 `template<typename _Tp > complex< _Tp > std::conj (const`
`complex< _Tp > & __z) [inline]`

Return complex conjugate of z .

Definition at line 667 of file `complex`.

2.11.2.11 `template<typename _Tp > complex< _Tp > std::cos (const`
`complex< _Tp > & __z) [inline]`

Return complex cosine of z .

Definition at line 699 of file `complex`.

Referenced by `std::polar()`.

2.11.2.12 `template<typename _Tp > complex< _Tp > std::cosh (const
complex< _Tp > & __z) [inline]`

Return complex hyperbolic cosine of z .

Definition at line 729 of file `complex`.

2.11.2.13 `template<typename _Tp > complex< _Tp > std::exp (const
complex< _Tp > & __z) [inline]`

Return complex base e exponential of z .

Definition at line 755 of file `complex`.

Referenced by `std::pow()`.

2.11.2.14 `template<typename _Tp > _Tp std::tr1::fabs (const std::complex<
_Tp > & __z) [inline]`

`fabs(__z)` [8.1.8].

Definition at line 301 of file `tr1_impl/complex`.

References `std::abs()`.

2.11.2.15 `template<typename _Tp > complex< _Tp > std::log (const
complex< _Tp > & __z) [inline]`

Return complex natural logarithm of z .

Definition at line 782 of file `complex`.

Referenced by `std::generate_canonical()`, `std::log10()`, `std::gamma_distribution< _RealType >::operator()`, `std::normal_distribution< _RealType >::operator()`, `std::binomial_distribution< _IntType >::operator()`, `std::poisson_distribution< _IntType >::operator()`, `std::independent_bits_engine< _RandomNumberEngine, _w, _UIntType >::operator()`, and `std::pow()`.

2.11.2.16 `template<typename _Tp > complex< _Tp > std::log10 (const
complex< _Tp > & __z) [inline]`

Return complex base 10 logarithm of z .

Definition at line 787 of file `complex`.

References `std::log()`.

2.11.2.17 `template<typename _Tp > _Tp std::norm (const complex< _Tp > & __z) [inline]`

Return z magnitude squared.

Definition at line 654 of file complex.

Referenced by `std::complex< _Tp >::operator/=()`.

2.11.2.18 `template<typename _Tp > bool std::operator!=(const complex< _Tp > & __x, const complex< _Tp > & __y) [inline]`

Return false if x is equal to y .

Definition at line 469 of file complex.

2.11.2.19 `template<typename _Tp > bool std::operator!=(const complex< _Tp > & __x, const _Tp & __y) [inline]`

Return false if x is equal to y .

Definition at line 474 of file complex.

2.11.2.20 `template<typename _Tp > bool std::operator!=(const _Tp & __x, const complex< _Tp > & __y) [inline]`

Return false if x is equal to y .

Definition at line 479 of file complex.

2.11.2.21 `template<typename _Tp > complex<_Tp> std::operator* (const complex< _Tp > & __x, const complex< _Tp > & __y) [inline]`

Return new complex value x times y .

Definition at line 379 of file complex.

2.11.2.22 `template<typename _Tp > complex<_Tp> std::operator* (const complex< _Tp > & __x, const _Tp & __y) [inline]`

Return new complex value x times y .

Definition at line 388 of file complex.

2.11.2.23 `template<typename _Tp > complex<_Tp> std::operator* (const
_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x times y .

Definition at line 397 of file complex.

2.11.2.24 `template<typename _Tp > complex<_Tp> & std::complex<_Tp
>::operator*=(const _Tp & __t) [inherited]`

Multiply this complex number by t .

Definition at line 238 of file complex.

2.11.2.25 `template<typename _Tp > template<typename _Up > complex<
_Tp> & std::complex<_Tp>::operator*=(const complex<_Up>
& __z) [inherited]`

Multiply this complex number by z .

Definition at line 292 of file complex.

2.11.2.26 `template<typename _Tp > complex<_Tp> std::operator+ (const
complex<_Tp> & __x) [inline]`

Return x .

Definition at line 438 of file complex.

2.11.2.27 `template<typename _Tp > complex<_Tp> std::operator+ (const
_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x plus y .

Definition at line 337 of file complex.

2.11.2.28 `template<typename _Tp > complex<_Tp> std::operator+ (const
complex<_Tp> & __x, const _Tp & __y) [inline]`

Return new complex value x plus y .

Definition at line 328 of file complex.

2.11.2.29 `template<typename _Tp> complex<_Tp> std::operator+ (`
 `const complex<_Tp> & __x, const complex<_Tp> & __y)`
 `[inline]`

Return new complex value x plus y .

Definition at line 319 of file complex.

2.11.2.30 `template<typename _Tp> template<typename _Up> complex<`
 `_Tp> & std::complex<_Tp>::operator+=(const complex<_Up>`
 `& __z) [inherited]`

Add z to this complex number.

Definition at line 269 of file complex.

2.11.2.31 `template<typename _Tp> complex<_Tp> std::operator- (const`
 `complex<_Tp> & __x) [inline]`

Return complex negation of x .

Definition at line 444 of file complex.

2.11.2.32 `template<typename _Tp> complex<_Tp> std::operator- (`
 `const complex<_Tp> & __x, const complex<_Tp> & __y)`
 `[inline]`

Return new complex value x minus y .

Definition at line 349 of file complex.

2.11.2.33 `template<typename _Tp> complex<_Tp> std::operator- (const`
 `complex<_Tp> & __x, const _Tp & __y) [inline]`

Return new complex value x minus y .

Definition at line 358 of file complex.

2.11.2.34 `template<typename _Tp> complex<_Tp> std::operator- (const`
 `_Tp & __x, const complex<_Tp> & __y) [inline]`

Return new complex value x minus y .

Definition at line 367 of file complex.

2.11.2.35 `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator-= (const complex<_Up > & __z) [inherited]`

Subtract z from this complex number.

Definition at line 280 of file complex.

2.11.2.36 `template<typename _Tp > complex<_Tp> std::operator/ (const complex<_Tp > & __x, const complex<_Tp > & __y) [inline]`

Return new complex value x divided by y .

Definition at line 409 of file complex.

2.11.2.37 `template<typename _Tp > complex<_Tp> std::operator/ (const complex<_Tp > & __x, const _Tp & __y) [inline]`

Return new complex value x divided by y .

Definition at line 418 of file complex.

2.11.2.38 `template<typename _Tp > complex<_Tp> std::operator/ (const _Tp & __x, const complex<_Tp > & __y) [inline]`

Return new complex value x divided by y .

Definition at line 427 of file complex.

2.11.2.39 `template<typename _Tp > complex<_Tp > & std::complex<_Tp >::operator/= (const _Tp & __t) [inherited]`

Divide this complex number by t .

Definition at line 248 of file complex.

2.11.2.40 `template<typename _Tp > template<typename _Up > complex<_Tp > & std::complex<_Tp >::operator/= (const complex<_Up > & __z) [inherited]`

Divide this complex number by z .

Definition at line 305 of file complex.

References `std::norm()`.

2.11.2.41 `template<typename _Tp, typename _CharT, class _Traits
> basic_ostream<_CharT, _Traits>& std::operator<< (
basic_ostream<_CharT, _Traits> & __os, const complex<_Tp>
& __x)`

Insertion operator for complex values.

Definition at line 519 of file complex.

References `std::ios_base::flags()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::ios_base::precision()`, and `std::basic_ostringstream<_CharT, _Traits, _Alloc>::str()`.

2.11.2.42 `template<typename _Tp> complex<_Tp> & std::complex<_Tp
>::operator=(const _Tp & __t) [inherited]`

Assign this complex number to scalar t .

Definition at line 228 of file complex.

2.11.2.43 `template<typename _Tp> template<typename _Up> complex<
_Tp> & std::complex<_Tp>::operator=(const complex<_Up>
& __z) [inherited]`

Assign this complex number to complex z .

Definition at line 258 of file complex.

2.11.2.44 `template<typename _Tp> bool std::operator==(const complex<
_Tp> & __x, const _Tp & __y) [inline]`

Return true if x is equal to y .

Definition at line 456 of file complex.

2.11.2.45 `template<typename _Tp> bool std::operator==(const complex<
_Tp> & __x, const complex<_Tp> & __y) [inline]`

Return true if x is equal to y .

Definition at line 451 of file complex.

2.11.2.46 `template<typename _Tp > bool std::operator==(const _Tp & __x,
const complex< _Tp > & __y) [inline]`

Return true if x is equal to y .

Definition at line 461 of file complex.

2.11.2.47 `template<typename _Tp , typename _CharT , class _Traits
> basic_istream< _CharT, _Traits>& std::operator>> (
basic_istream< _CharT, _Traits > & __is, complex< _Tp > & __x)`

Extraction operator for complex values.

Definition at line 486 of file complex.

References `std::ios_base::failbit`, `std::basic_istream< _CharT, _Traits >::putback()`,
and `std::basic_ios< _CharT, _Traits >::setstate()`.

2.11.2.48 `template<typename _Tp > complex< _Tp > std::polar (const _Tp
& __rho, const _Tp & __theta = 0) [inline]`

Return complex with magnitude ρ and angle θ .

Definition at line 662 of file complex.

References `std::cos()`, and `std::sin()`.

Referenced by `std::pow()`.

2.11.2.49 `template<typename _Tp > complex< _Tp > std::pow (const _Tp &
__x, const complex< _Tp > & __y) [inline]`

Return x to the y 'th power.

Definition at line 1029 of file complex.

References `std::log()`, `std::polar()`, and `std::pow()`.

2.11.2.50 `template<typename _Tp > complex< _Tp > std::pow (const
complex< _Tp > & __x, const _Tp & __y)`

Return x to the y 'th power.

Definition at line 984 of file complex.

References `std::exp()`, `std::log()`, and `std::polar()`.

Referenced by `std::gamma_distribution< _RealType >::operator()()`, and `std::pow()`.

2.11.2.51 `template<typename _Tp > complex< _Tp > std::pow (const
complex< _Tp > & __x, const complex< _Tp > & __y)
[inline]`

Return x to the y 'th power.

Definition at line 1023 of file `complex`.

2.11.2.52 `template<typename _Tp > complex< _Tp > std::sin (const
complex< _Tp > & __z) [inline]`

Return complex sine of z .

Definition at line 817 of file `complex`.

Referenced by `std::polar()`.

2.11.2.53 `template<typename _Tp > complex< _Tp > std::sinh (const
complex< _Tp > & __z) [inline]`

Return complex hyperbolic sine of z .

Definition at line 847 of file `complex`.

2.11.2.54 `template<typename _Tp > complex< _Tp > std::sqrt (const
complex< _Tp > & __z) [inline]`

Return complex square root of z .

Definition at line 891 of file `complex`.

Referenced by `std::normal_distribution< _RealType >::operator()()`, and
`std::student_t_distribution< _RealType >::operator()()`.

2.11.2.55 `template<typename _Tp > complex< _Tp > std::tan (const
complex< _Tp > & __z) [inline]`

Return complex tangent of z .

Definition at line 918 of file `complex`.

2.11.2.56 `template<typename _Tp > complex< _Tp > std::tanh (const
complex< _Tp > & __z) [inline]`

Return complex hyperbolic tangent of z .

Definition at line 946 of file complex.

2.12 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class [std::condition_variable](#)
condition_variable
- class [std::condition_variable_any](#)
condition_variable_any

Enumerations

- enum [std::cv_status](#) { **no_timeout**, **timeout** }

2.12.1 Detailed Description

Classes for [condition_variable](#) support.

2.12.2 Enumeration Type Documentation

2.12.2.1 enum std::cv_status

`cv_status`

Definition at line 54 of file `condition_variable`.

2.13 Futures

Collaboration diagram for Futures:



Classes

- class `std::__basic_future<_Res>`
Common implementation for future and `shared_future`.
- struct `std::__future_base`
Base class and enclosing scope.
- struct `std::__future_base::Result<_Res &>`
Partial specialization for reference types.
- struct `std::__future_base::Result<void>`
Explicit specialization for void.
- class `std::future<_Res>`
Primary template for future.
- class `std::future<_Res &>`
Partial specialization for `future<R&>`
- class `std::future<void>`
Explicit specialization for `future<void>`
- class `std::future_error`
Exception type thrown by futures.
- class `std::packaged_task<_Res(_ArgTypes...)>`
`packaged_task`

- class `std::promise< _Res >`
Primary template for `promise`.
- class `std::promise< _Res & >`
Partial specialization for `promise<R&>`
- class `std::promise< void >`
Explicit specialization for `promise<void>`
- class `std::shared_future< _Res >`
Primary template for `shared_future`.
- class `std::shared_future< _Res & >`
Partial specialization for `shared_future<R&>`
- class `std::shared_future< void >`
Explicit specialization for `shared_future<void>`

Enumerations

- enum `std::future_errc` { `broken_promise`, `future_already_retrieved`, `promise_already_satisfied`, `no_state` }
- enum `launch` { `any`, `async`, `sync` }

Functions

- `std::__basic_future::__basic_future` (const `shared_future< _Res > &`)
- `std::__basic_future::__basic_future` (`shared_future< _Res > &&`)
- `std::__basic_future::__basic_future` (`future< _Res > &&`)
- static `_Setter< void, void > std::__future_base::State::__setter` (`promise< void > *__prom`)
- `template<typename _Fn, typename... _Args>`
`enable_if<!is_same< typename decay< _Fn >::type, launch >::value, future<`
`decltype(std::declval< _Fn >)(std::declval< _Args >)...)> >::type std::async`
`(_Fn &&__fn, _Args &&...__args)`
- `template<typename _Fn, typename... _Args>`
`future< typename result_of< _Fn(_Args...)>::type > std::async` (`launch __-`
`policy, _Fn &&__fn, _Args &&...__args)`
- error_code `std::make_error_code` (`future_errc __errc`)
- error_condition `std::make_error_condition` (`future_errc __errc`)

- void **std::promise**< **void** >::**set_value** ()
- template<typename _Res , typename... _ArgTypes>
void **std::swap** (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y)
- template<typename _Res >
void **std::swap** (promise< _Res > &__x, promise< _Res > &__y)

Variables

- const error_category *const [std::future_category](#)

2.13.1 Detailed Description

Classes for futures support.

2.13.2 Enumeration Type Documentation

2.13.2.1 enum std::future_errc

Error code for futures.

Definition at line 59 of file future.

2.13.3 Variable Documentation

2.13.3.1 const error_category* const std::future_category

Points to a statically-allocated object derived from [error_category](#).

2.14 I/O

Classes

- class [__gnu_cxx::stdio_filebuf](#)< _CharT, _Traits >
*Provides a layer of compatibility for C/POSIX.
This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.*
- class [__gnu_cxx::stdio_sync_filebuf](#)< _CharT, _Traits >

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE's. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

- class `std::basic_filebuf<_CharT, _Traits>`

The actual work of input and output (for files).

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's FILE streams.

- class `std::basic_fstream<_CharT, _Traits>`

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

- class `std::basic_ifstream<_CharT, _Traits>`

Controlling input for files.

This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

- class `std::basic_ios<_CharT, _Traits>`

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

- class `std::basic_iostream<_CharT, _Traits>`

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

- class `std::basic_istream<_CharT, _Traits>`

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.

- class `std::basic_istream<_CharT, _Traits, _Alloc>`

Controlling input for `std::string`.

This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::basic_ofstream<_CharT, _Traits>`

Controlling output for files.

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

- class `std::basic_ostream< _CharT, _Traits >`

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.

- class `std::basic_ostringstream< _CharT, _Traits, _Alloc >`

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::basic_streambuf< _CharT, _Traits >`

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

- class `std::basic_stringbuf< _CharT, _Traits, _Alloc >`

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).

- class `std::basic_stringstream< _CharT, _Traits, _Alloc >`

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

- class `std::ios_base`

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Typedefs

- typedef `basic_filebuf< char >` `std::filebuf`
- typedef `basic_fstream< char >` `std::fstream`

- `typedef basic_ifstream< char > std::ifstream`
- `typedef basic_ios< char > std::ios`
- `typedef basic_iostream< char > std::iostream`
- `typedef basic_istream< char > std::istream`
- `typedef basic_istreamstream< char > std::istreamstream`
- `typedef basic_ofstream< char > std::ofstream`
- `typedef basic_ostream< char > std::ostream`
- `typedef basic_ostringstream< char > std::ostringstream`
- `typedef basic_streambuf< char > std::streambuf`
- `typedef basic_stringbuf< char > std::stringbuf`
- `typedef basic_stringstream< char > std::stringstream`
- `typedef basic_filebuf< wchar_t > std::wfilebuf`
- `typedef basic_fstream< wchar_t > std::wfstream`
- `typedef basic_ifstream< wchar_t > std::wifstream`
- `typedef basic_ios< wchar_t > std::wios`
- `typedef basic_iostream< wchar_t > std::wiostream`
- `typedef basic_istream< wchar_t > std::wistream`
- `typedef basic_istreamstream< wchar_t > std::wistreamstream`
- `typedef basic_ofstream< wchar_t > std::wofstream`
- `typedef basic_ostream< wchar_t > std::wostream`
- `typedef basic_ostringstream< wchar_t > std::wostringstream`
- `typedef basic_streambuf< wchar_t > std::wstreambuf`
- `typedef basic_stringbuf< wchar_t > std::wstringbuf`
- `typedef basic_stringstream< wchar_t > std::wstringstream`

2.14.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt>

2.14.2 Typedef Documentation

2.14.2.1 `typedef basic_filebuf<char> std::filebuf`

One of the [I/O](#) typedefs.

Definition at line 137 of file iosfwd.

2.14.2.2 `typedef basic_fstream<char> std::fstream`

One of the [I/O](#) typedefs.

Definition at line 140 of file iosfwd.

2.14.2.3 `typedef basic_ifstream<char> std::ifstream`

One of the [I/O](#) typedefs.

Definition at line 138 of file iosfwd.

2.14.2.4 `typedef basic_ios<char> std::ios`

One of the [I/O](#) typedefs.

Definition at line 123 of file iosfwd.

2.14.2.5 `typedef basic_iostream<char> std::iostream`

One of the [I/O](#) typedefs.

Definition at line 132 of file iosfwd.

2.14.2.6 `typedef basic_istream<char> std::istream`

One of the [I/O](#) typedefs.

Definition at line 130 of file iosfwd.

2.14.2.7 `typedef basic_istreamstream<char> std::istreamstream`

One of the [I/O](#) typedefs.

Definition at line 134 of file iosfwd.

2.14.2.8 typedef basic_ofstream<char> std::ofstream

One of the [I/O](#) typedefs.

Definition at line 139 of file iosfwd.

2.14.2.9 typedef basic_ostream<char> std::ostream

One of the [I/O](#) typedefs.

Definition at line 131 of file iosfwd.

2.14.2.10 typedef basic_ostringstream<char> std::ostringstream

One of the [I/O](#) typedefs.

Definition at line 135 of file iosfwd.

2.14.2.11 typedef basic_streambuf<char> std::streambuf

One of the [I/O](#) typedefs.

Definition at line 129 of file iosfwd.

2.14.2.12 typedef basic_stringbuf<char> std::stringbuf

One of the [I/O](#) typedefs.

Definition at line 133 of file iosfwd.

2.14.2.13 typedef basic_stringstream<char> std::stringstream

One of the [I/O](#) typedefs.

Definition at line 136 of file iosfwd.

2.14.2.14 typedef basic_filebuf<wchar_t> std::wfilebuf

One of the [I/O](#) typedefs.

Definition at line 152 of file iosfwd.

2.14.2.15 typedef basic_fstream<wchar_t> std::wfstream

One of the [I/O](#) typedefs.

Definition at line 155 of file iosfwd.

2.14.2.16 typedef basic_ifstream<wchar_t> std::wifstream

One of the [I/O](#) typedefs.

Definition at line 153 of file iosfwd.

2.14.2.17 typedef basic_ios<wchar_t> std::wios

One of the [I/O](#) typedefs.

Definition at line 143 of file iosfwd.

2.14.2.18 typedef basic_iostream<wchar_t> std::wiostream

One of the [I/O](#) typedefs.

Definition at line 147 of file iosfwd.

2.14.2.19 typedef basic_istream<wchar_t> std::wistream

One of the [I/O](#) typedefs.

Definition at line 145 of file iosfwd.

2.14.2.20 typedef basic_istreamstream<wchar_t> std::wistreamstream

One of the [I/O](#) typedefs.

Definition at line 149 of file iosfwd.

2.14.2.21 typedef basic_ofstream<wchar_t> std::wofstream

One of the [I/O](#) typedefs.

Definition at line 154 of file iosfwd.

2.14.2.22 typedef basic_ostream<wchar_t> std::wostream

One of the [I/O](#) typedefs.

Definition at line 146 of file iosfwd.

2.14.2.23 typedef basic_ostringstream<wchar_t> std::wostringstream

One of the [I/O](#) typedefs.

Definition at line 150 of file iosfwd.

2.14.2.24 typedef basic_streambuf<wchar_t> std::wstreambuf

One of the [I/O](#) typedefs.

Definition at line 144 of file iosfwd.

2.14.2.25 typedef basic_stringbuf<wchar_t> std::wstringbuf

One of the [I/O](#) typedefs.

Definition at line 148 of file iosfwd.

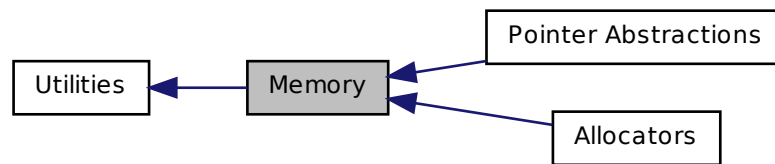
2.14.2.26 typedef basic_stringstream<wchar_t> std::wstringstream

One of the [I/O](#) typedefs.

Definition at line 151 of file iosfwd.

2.15 Memory

Collaboration diagram for Memory:



Modules

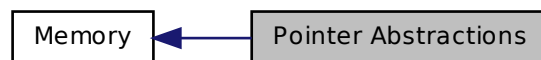
- [Pointer Abstractions](#)
- [Allocators](#)

2.15.1 Detailed Description

Components for memory allocation, deallocation, and management.

2.16 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



Classes

- struct `std::default_delete< _Tp >`
Primary template, `default_delete`.
- struct `std::default_delete< _Tp[]>`
Specialization, `default_delete`.
- class `std::enable_shared_from_this< _Tp >`
Base class allowing use of member function `shared_from_this`.
- struct `std::hash< shared_ptr< _Tp > >`
`std::hash` specialization for `shared_ptr`.
- struct `std::hash< unique_ptr< _Tp, _Dp > >`
`std::hash` specialization for `unique_ptr`.
- struct `std::owner_less< shared_ptr< _Tp > >`
Partial specialization of `owner_less` for `shared_ptr`.
- struct `std::owner_less< weak_ptr< _Tp > >`
Partial specialization of `owner_less` for `weak_ptr`.
- class `std::shared_ptr< _Tp >`
A smart pointer with reference-counted copy semantics.
- class `std::unique_ptr< _Tp, _Dp >`
20.7.12.2 `unique_ptr` for single objects.
- class `std::unique_ptr< _Tp[], _Dp >`
20.7.12.3 `unique_ptr` for array objects with a runtime length
- class `std::weak_ptr< _Tp >`
A smart pointer with weak semantics.

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
`shared_ptr< _Tp > std::allocate_shared (_Alloc __a, _Args &&...__args)`
- template<typename _Tp, typename _Tp1 >
`shared_ptr< _Tp > std::const_pointer_cast (const shared_ptr< _Tp1 > &__r)`

- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::dynamic_pointer_cast (const shared_ptr< _Tp1 >`
`&__r)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`
`_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename... _Args>`
`shared_ptr< _Tp > std::make_shared (_Args &&...__args)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp >`
`bool std::operator!= (nullptr_t, const shared_ptr< _Tp > &__b)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr<`
`_Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Tp >`
`bool std::operator!= (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr<`
`_Up, _Ep > &__y)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch,`
`_Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_`
`ptr< _Up, _Ep > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Tp >`
`bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr<`
`_Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp >`
`bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__b)`

- `template<typename _Tp, typename _Dp >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Dp >`
`void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`

2.16.1 Detailed Description

Smart pointers, etc.

2.16.2 Function Documentation

2.16.2.1 `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr<_Tp> std::allocate_shared (_Alloc __a, _Args &&...
__args) [inline]`

Create an object that is owned by a [shared_ptr](#).

Parameters

`__a` An allocator.

`__args` Arguments for the `_Tp` object's constructor.

Returns

A [shared_ptr](#) that owns the newly created object.

Exceptions

An exception thrown from `_Alloc::allocate` or from the constructor of `_Tp`.

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.
Definition at line 520 of file `shared_ptr.h`.

2.16.2.2 `template<typename _Del , typename _Tp , _Lock_policy _Lp>
_Del* std::get_deleter (const __shared_ptr< _Tp, _Lp > & __p)
[inline]`

2.2.3.10 `shared_ptr` `get_deleter` (experimental)

Definition at line 74 of file `shared_ptr.h`.

2.16.2.3 `template<typename _Tp , typename... _Args> shared_ptr<_Tp>
std::make_shared (_Args &&... __args) [inline]`

Create an object that is owned by a `shared_ptr`.

Parameters

`__args` Arguments for the `_Tp` object's constructor.

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

`std::bad_alloc`, or an exception thrown from the constructor of `_Tp`.

Definition at line 535 of file `shared_ptr.h`.

2.16.2.4 `template<typename _Ch , typename _Tr , typename _Tp
, _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr>&
std::operator<< (std::basic_ostream<_Ch, _Tr > & __os, const
__shared_ptr< _Tp, _Lp > & __p) [inline]`

2.2.3.7 `shared_ptr` I/O

Definition at line 64 of file `shared_ptr.h`.

2.17 Mutexes

Collaboration diagram for Mutexes:



Classes

- struct `std::adopt_lock_t`
Assume the calling thread has already obtained mutex ownership /// and manage it.
- struct `std::defer_lock_t`
Do not acquire ownership of the mutex.
- class `std::lock_guard<_Mutex>`
Scoped lock idiom.
- class `std::mutex`
mutex
- struct `std::once_flag`
once_flag
- class `std::recursive_mutex`
recursive_mutex
- class `std::recursive_timed_mutex`
recursive_timed_mutex
- class `std::timed_mutex`
timed_mutex
- struct `std::try_to_lock_t`
Try to acquire ownership of the mutex without blocking.

- class `std::unique_lock<_Mutex>`
unique_lock

Functions

- mutex & `std::__get_once_mutex()`
- void `std::__once_proxy()`
- void `std::__set_once_function_lock_ptr` (unique_lock< mutex > *)
- template<typename _Callable, typename... _Args>
void `std::call_once` (once_flag &__once, _Callable &&__f, _Args &&...__args)
- template<typename _L1, typename _L2, typename... _L3>
void `std::lock` (_L1 &, _L2 &, _L3 &...)
- template<typename _Mutex>
void `std::swap` (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y)
- template<typename _Lock1, typename _Lock2, typename... _Lock3>
int `std::try_lock` (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)

Variables

- function< void()> `std::__once_function`
- const adopt_lock_t `std::adopt_lock`
- const defer_lock_t `std::defer_lock`
- const try_to_lock_t `std::try_to_lock`

2.17.1 Detailed Description

Classes for mutex support.

2.17.2 Function Documentation

- 2.17.2.1** template<typename _Callable, typename... _Args> void
std::call_once (once_flag & __once, _Callable && __f, _Args &&...
__args)

call_once

Definition at line 721 of file mutex.

2.17.2.2 `template<typename _L1, typename _L2, typename... _L3> void
std::lock (_L1 &, _L2 &, _L3 & ...)`

lock

2.17.2.3 `template<typename _Lock1, typename _Lock2, typename... _Lock3>
int std::try_lock (_Lock1 & __l1, _Lock2 & __l2, _Lock3 &... __l3
)`

Generic try_lock.

Parameters

`__l1` Meets Mutex requirements ([try_lock\(\)](#) may throw).

`__l2` Meets Mutex requirements ([try_lock\(\)](#) may throw).

`__l3` Meets Mutex requirements ([try_lock\(\)](#) may throw).

Returns

Returns -1 if all [try_lock\(\)](#) calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

Either all arguments are locked, or none will be.

Sequentially calls [try_lock\(\)](#) on each argument.

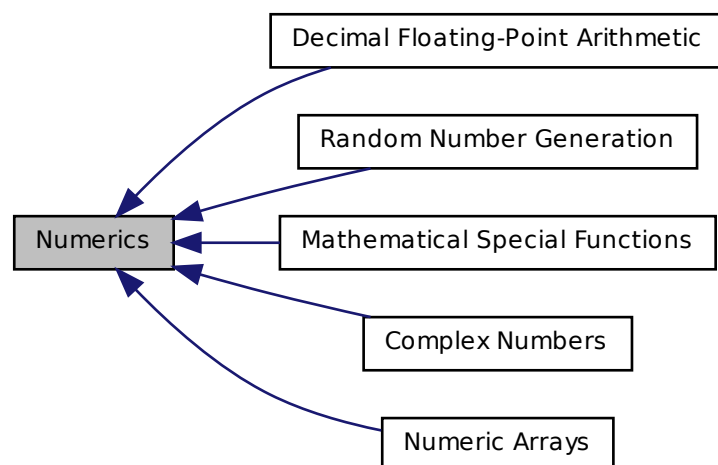
Definition at line 663 of file mutex.

References `std::try_lock()`.

Referenced by `std::try_lock()`.

2.18 Numerics

Collaboration diagram for Numerics:



Modules

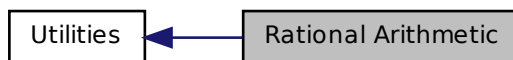
- [Complex Numbers](#)
- [Numeric Arrays](#)
- [Mathematical Special Functions](#)
- [Decimal Floating-Point Arithmetic](#)
- [Random Number Generation](#)

2.18.1 Detailed Description

Components for performing numeric operations. Includes support for for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and special math functions.

2.19 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



Classes

- struct `std::ratio< _Num, _Den >`
Provides compile-time rational arithmetic.
- struct `std::ratio_add< _R1, _R2 >`
ratio_add
- struct `std::ratio_divide< _R1, _R2 >`
ratio_divide
- struct `std::ratio_equal< _R1, _R2 >`
ratio_equal
- struct `std::ratio_multiply< _R1, _R2 >`
ratio_multiply
- struct `std::ratio_not_equal< _R1, _R2 >`
ratio_not_equal
- struct `std::ratio_subtract< _R1, _R2 >`
ratio_subtract

Typedefs

- typedef `ratio< __safe_add< __safe_multiply< _R1::num,(_R2::den/__gcd)>::value, __safe_multiply< _R2::num,(_R1::den/__gcd)>::value >::value,`

```
__safe_multiply< _R1::den,(_R2::den/__gcd)>::value > std::ratio_add::type
```

- `typedef ratio_multiply< _R1, ratio< _R2::den, _R2::num > >::type`
std::ratio_divide::type
- `typedef ratio< __safe_multiply<(_R1::num/__gcd1),(_R2::num/_-`
`_gcd2)>::value, __safe_multiply<(_R1::den/__gcd2),(_R2::den/_-`
`gcd1)>::value > std::ratio_multiply::type`

Functions

- **std::ratio::static_assert** (`_Num >= __INTMAX_MAX__` && `_Den >= __-`
`INTMAX_MAX__, "out of range"`)

Variables

- static const intmax_t **std::ratio::den**
- static const intmax_t **std::ratio::num**
- static const intmax_t **std::__safe_multiply::value**
- static const intmax_t **std::__safe_add::value**

2.19.1 Detailed Description

Compile time representation of finite rational numbers.

2.20 Threads

Collaboration diagram for Threads:



Classes

- struct `std::hash< thread::id >`

std::hash specialization for thread::id.

- class `std::thread`

thread

Namespaces

- namespace `std::this_thread`

Functions

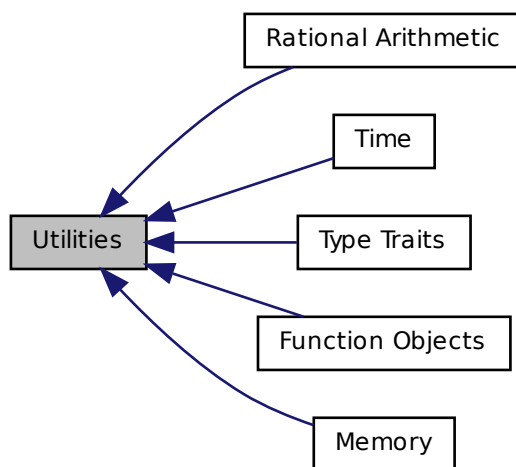
- bool **std::operator!=** (thread::id __x, thread::id __y)
- template<class _CharT, class _Traits >
basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _CharT, _Traits > &__out, thread::id __id)
- bool **std::operator<=** (thread::id __x, thread::id __y)
- bool **std::operator>** (thread::id __x, thread::id __y)
- bool **std::operator>=** (thread::id __x, thread::id __y)
- void **std::swap** (thread &__x, thread &__y)

2.20.1 Detailed Description

Classes for thread support.

2.21 Utilities

Collaboration diagram for Utilities:



Modules

- [Time](#)
- [Memory](#)
- [Rational Arithmetic](#)
- [Type Traits](#)
- [Function Objects](#)

2.21.1 Detailed Description

Components deemed generally useful. Includes pair, tuple, forward/move helpers, ratio, function object, metaprogramming and type traits, time, date, and memory functions.

2.22 Numeric Arrays

Collaboration diagram for Numeric Arrays:



Classes

- class `std::gslice`
Class defining multi-dimensional subset of an array.
- class `std::gslice_array< _Tp >`
Reference to multi-dimensional subset of an array.
- class `std::indirect_array< _Tp >`
Reference to arbitrary subset of an array.
- class `std::mask_array< _Tp >`
Reference to selected subset of an array.
- class `std::slice`
Class defining one-dimensional subset of an array.
- class `std::slice_array< _Tp >`
Reference to one-dimensional subset of an array.
- class `std::valarray< _Tp >`
Smart array designed to support numeric processing.

Defines

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`

- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

Functions

- `std::gslice::gslice ()`
- `std::gslice::gslice (size_t, const valarray< size_t > &, const valarray< size_t > &)`
- `std::gslice::gslice (const gslice &)`
- `std::gslice_array::gslice_array (const gslice_array &)`
- `std::indirect_array::indirect_array (const indirect_array &)`
- `std::mask_array::mask_array (const mask_array &)`
- `std::slice::slice ()`
- `std::slice::slice (size_t, size_t, size_t)`
- `std::slice_array::slice_array (const slice_array &)`
- `std::valarray::valarray (const slice_array< _Tp > &)`
- `std::valarray::valarray (const gslice_array< _Tp > &)`
- `std::valarray::valarray (const valarray &)`
- `std::valarray::valarray (const mask_array< _Tp > &)`
- `std::valarray::valarray (const indirect_array< _Tp > &)`
- `std::valarray::valarray ()`
- `template<typename _Tp>`
`std::valarray::valarray (const _Tp *__restrict __p, size_t __n)`
- `std::valarray::valarray (initializer_list< _Tp >)`
- `template<class _Dom >`
`std::valarray::valarray (const _Expr< _Dom, _Tp > &__e)`
- `std::valarray::valarray (size_t)`
- `std::valarray::valarray (const _Tp &, size_t)`
- `std::gslice::~gslice ()`
- `_Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray::apply (_Tp func(_Tp)) const`
- `_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray::apply (_Tp func(const _Tp &)) const`
- `template<class _Tp >`
`_Tp * std::begin (valarray< _Tp > &__va)`

- `template<class _Tp >`
`const _Tp * std::begin (const valarray< _Tp > &__va)`
- `valarray< _Tp > std::valarray::cshift (int) const`
- `template<class _Tp >`
`const _Tp * std::end (const valarray< _Tp > &__va)`
- `template<class _Tp >`
`_Tp * std::end (valarray< _Tp > &__va)`
- `_Tp std::valarray::max () const`
- `_Tp std::valarray::min () const`
- `_UnaryOp< __logical_not >::_Rt std::valarray::operator! () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const
valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const _-
Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-
name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const
valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename
__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _-
Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename
__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _-
Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename
__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &__t,
const valarray< _Tp > &__v)`
- `void std::gslice_array::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator%= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator%= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator%= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator%= (const _Expr< _Dom, _Tp >
&)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const _Tp`
`&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const _`
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator&= (const _Expr< _Dom, _Tp >`
`&)`
- `void std::gslice_array::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator&= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator&= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator&= (const valarray< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array::operator*= (const valarray< _Tp > &) const`

- `template<class _Dom >`
`void std::gslice_array::operator*= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator*= (const valarray< _Tp > &)`
- `valarray< _Tp > & std::valarray::operator*= (const _Tp &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator*= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _-`
`_fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `_UnaryOp< __unary_plus >::Rt std::valarray::operator+ () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `void std::gslice_array::operator+= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator+= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator+= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator+= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator+= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _-`
`_fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `_UnaryOp< __negate >::Rt std::valarray::operator- () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __minus, _Tp >::result_type > std::operator- (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<class _Dom >`
`void std::gslice_array::operator-= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator-= (const valarray< _Tp > &) const`

- `valarray<_Tp> & std::valarray::operator= (const valarray<_Tp> &)`
- `valarray<_Tp> & std::valarray::operator= (const _Tp &)`
- `template<class _Dom>`
`valarray<_Tp> & std::valarray::operator= (const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp>`
`_Expr<_BinClos<__divides, _ValArray, _ValArray, _Tp, _Tp>, typename _-`
`_fun<__divides, _Tp>::result_type> std::operator/ (const valarray<_Tp>`
`&__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`
`_Expr<_BinClos<__divides, _ValArray, _Constant, _Tp, _Tp>, typename _-`
`_fun<__divides, _Tp>::result_type> std::operator/ (const valarray<_Tp>`
`&__v, const _Tp &__t)`
- `template<typename _Tp>`
`_Expr<_BinClos<__divides, _Constant, _ValArray, _Tp, _Tp>, typename _-`
`_fun<__divides, _Tp>::result_type> std::operator/ (const _Tp &__t, const`
`valarray<_Tp> &__v)`
- `template<class _Dom>`
`void std::gslice_array::operator= (const _Expr<_Dom, _Tp> &) const`
- `void std::gslice_array::operator= (const valarray<_Tp> &) const`
- `valarray<_Tp> & std::valarray::operator= (const _Tp &)`
- `valarray<_Tp> & std::valarray::operator= (const valarray<_Tp> &)`
- `template<class _Dom>`
`valarray<_Tp> & std::valarray::operator= (const _Expr<_Dom, _Tp>`
`&)`
- `template<typename _Tp>`
`_Expr<_BinClos<__less, _ValArray, _ValArray, _Tp, _Tp>, typename _-`
`_fun<__less, _Tp>::result_type> std::operator< (const valarray<_Tp>`
`&__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`
`_Expr<_BinClos<__less, _ValArray, _Constant, _Tp, _Tp>, typename _-`
`_fun<__less, _Tp>::result_type> std::operator< (const valarray<_Tp>`
`&__v, const _Tp &__t)`
- `template<typename _Tp>`
`_Expr<_BinClos<__less, _Constant, _ValArray, _Tp, _Tp>, typename _-`
`_fun<__less, _Tp>::result_type> std::operator< (const _Tp &__t, const`
`valarray<_Tp> &__v)`
- `template<typename _Tp>`
`_Expr<_BinClos<__shift_left, _ValArray, _ValArray, _Tp, _Tp>, typename`
`__fun<__shift_left, _Tp>::result_type> std::operator<< (const valarray<`
`_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`
`_Expr<_BinClos<__shift_left, _ValArray, _Constant, _Tp, _Tp>, typename`
`__fun<__shift_left, _Tp>::result_type> std::operator<< (const valarray<`
`_Tp> &__v, const _Tp &__t)`

- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator<= (const _Expr< _Dom, _Tp`
`> &)`
- `void std::gslice_array::operator<= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array::operator<= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator<= (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator<= (const valarray< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `void std::indirect_array::operator= (const _Tp &) const`
- `valarray< _Tp > & std::valarray::operator= (const valarray< _Tp > &)`
- `void std::slice_array::operator= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `gslice & std::gslice::operator= (const gslice &)`
- `void std::mask_array::operator= (const _Tp &) const`
- `gslice_array & std::gslice_array::operator= (const gslice_array &)`
- `template<class _Dom >`
`void std::gslice_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator= (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator= (const _Tp &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator= (const _Expr< _Dom, _Tp > &)`
- `valarray< _Tp > & std::valarray::operator= (const slice_array< _Tp > &)`
- `indirect_array & std::indirect_array::operator= (const indirect_array &)`
- `void std::indirect_array::operator= (const valarray< _Tp > &) const`
- `mask_array & std::mask_array::operator= (const mask_array &)`

- `template<class _Ex >`
`void std::mask_array::operator= (const _Expr< _Ex, _Tp > &__e) const`
- `void std::mask_array::operator= (const valarray< _Tp > &) const`
- `slice_array & std::slice_array::operator= (const slice_array &)`
- `void std::slice_array::operator= (const _Tp &) const`
- `valarray< _Tp > & std::valarray::operator= (const mask_array< _Tp > &)`
- `template<class _Dom >`
`void std::slice_array::operator= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator= (const indirect_array< _Tp > &)`
- `void std::gslice_array::operator= (const _Tp &) const`
- `valarray< _Tp > & std::valarray::operator= (const gslice_array< _Tp > &)`
- `valarray & std::valarray::operator= (initializer_list< _Tp >)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`
`> &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _`
`__fun< __greater, _Tp >::result_type > std::operator> (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > std::operator>=` (const
`_Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>>` (const _Tp &__t,
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>>` (const valarray<
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > std::operator>>` (const valarray<
`_Tp > &__v, const _Tp &__t)`
- `template<class _Dom >`
`void std::gslice_array::operator>>=` (const _Expr< _Dom, _Tp > &) const
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator>>=` (const _Expr< _Dom, _Tp
`> &)`
- `void std::gslice_array::operator>>=` (const valarray< _Tp > &) const
- `valarray< _Tp > & std::valarray::operator>>=` (const valarray< _Tp > &)
- `valarray< _Tp > & std::valarray::operator>>=` (const _Tp &)
- `slice_array< _Tp > std::valarray::operator[]` (slice)
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray::operator[]` (const
`gslice &) const`
- `valarray< _Tp > std::valarray::operator[]` (const valarray< bool > &) const
- `mask_array< _Tp > std::valarray::operator[]` (const valarray< bool > &)
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray::operator[]` (const
`valarray< size_t > &) const`
- `indirect_array< _Tp > std::valarray::operator[]` (const valarray< size_t > &)
- `_Tp & std::valarray::operator[]` (size_t)
- `gslice_array< _Tp > std::valarray::operator[]` (const gslice &)
- `const _Tp & std::valarray::operator[]` (size_t) const
- `_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray::operator[]` (slice)
`const`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^` (const
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^` (const
`valarray< _Tp > &__v, const _Tp &__t)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const _Tp`
`&__t, const valarray< _Tp > &__v)`
- `valarray< _Tp > & std::valarray::operator^ = (const _Tp &)`
- `valarray< _Tp > & std::valarray::operator^ = (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator^ = (const _Expr< _Dom, _Tp >`
`&)`
- `template<class _Dom >`
`void std::gslice_array::operator^ = (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array::operator^ = (const valarray< _Tp > &) const`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`
`Tp > &__v, const _Tp &__t)`
- `valarray< _Tp > & std::valarray::operator| = (const valarray< _Tp > &)`
- `template<class _Dom >`
`void std::gslice_array::operator| = (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray::operator| = (const _Expr< _Dom, _Tp >`
`&)`
- `void std::gslice_array::operator| = (const valarray< _Tp > &) const`
- `valarray< _Tp > & std::valarray::operator| = (const _Tp &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`
`Tp > &__v, const _Tp &__t)`

- `_UnaryOp< __bitwise_not >::_Rt std::valarray::operator~ () const`
- `void std::valarray::resize (size_t __size, _Tp __c=_Tp())`
- `valarray< _Tp > std::valarray::shift (int) const`
- `size_t std::slice::size () const`
- `valarray< size_t > std::gslice::size () const`
- `size_t std::valarray::size () const`
- `size_t std::gslice::start () const`
- `size_t std::slice::start () const`
- `size_t std::slice::stride () const`
- `valarray< size_t > std::gslice::stride () const`
- `_Tp std::valarray::sum () const`

2.22.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

2.22.2 Function Documentation

2.22.2.1 `std::gslice::gslice () [inline, inherited]`

Construct an empty slice.

Definition at line 147 of file `gslice.h`.

2.22.2.2 `std::gslice::gslice (size_t __o, const valarray< size_t > & __l, const valarray< size_t > & __s) [inline, inherited]`

Construct a slice.

Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

Parameters

- o* Offset in array of first element.
- l* Array of dimension lengths.
- s* Array of dimension strides between array elements.

Definition at line 151 of file `gslice.h`.

2.22.2.3 `std::gslice::gslice (const gslice & __g) [inline, inherited]`

Copy constructor.

Definition at line 156 of file `gslice.h`.

2.22.2.4 `template<typename _Tp> std::gslice_array<_Tp>::gslice_array (`
`const gslice_array<_Tp> & __a) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 142 of file gslice_array.h.

2.22.2.5 `template<typename _Tp> std::indirect_array<_Tp`
`>::indirect_array (const indirect_array<_Tp> & __a)`
`[inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 142 of file indirect_array.h.

2.22.2.6 `template<typename _Tp> std::mask_array<_Tp>::mask_array (`
`const mask_array<_Tp> & a) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 138 of file mask_array.h.

2.22.2.7 `std::slice::slice () [inline, inherited]`

Construct an empty slice.

Definition at line 89 of file slice_array.h.

2.22.2.8 `std::slice::slice (size_t __o, size_t __d, size_t __s) [inline,`
`inherited]`

Construct a slice.

Parameters

o Offset in array of first element.

d Number of elements in slice.

s Stride between array elements.

Definition at line 93 of file slice_array.h.

2.22.2.9 `template<typename _Tp> std::slice_array<_Tp>::slice_array (`
`const slice_array<_Tp> & a) [inline, inherited]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 206 of file slice_array.h.

2.22.2.10 `template<typename _Tp> std::valarray<_Tp>::valarray (const slice_array<_Tp> & __sa) [inline, inherited]`

Construct an array with the same size and values in *sa*.

Definition at line 609 of file valarray.

2.22.2.11 `template<typename _Tp> std::valarray<_Tp>::valarray (const gslice_array<_Tp> & __ga) [inline, inherited]`

Construct an array with the same size and values in *ga*.

Definition at line 618 of file valarray.

2.22.2.12 `template<typename _Tp> std::valarray<_Tp>::valarray (const valarray<_Tp> & __v) [inline, inherited]`

Copy constructor.

Definition at line 602 of file valarray.

2.22.2.13 `template<typename _Tp> std::valarray<_Tp>::valarray (const mask_array<_Tp> & __ma) [inline, inherited]`

Construct an array with the same size and values in *ma*.

Definition at line 629 of file valarray.

2.22.2.14 `template<typename _Tp> std::valarray<_Tp>::valarray (const indirect_array<_Tp> & __ia) [inline, inherited]`

Construct an array with the same size and values in *ia*.

Definition at line 638 of file valarray.

2.22.2.15 `template<typename _Tp> std::valarray<_Tp>::valarray () [inline, inherited]`

Construct an empty array.

Definition at line 577 of file valarray.

2.22.2.16 `template<typename _Tp> std::valarray<_Tp>::valarray (`
`initializer_list<_Tp> __l) [inline, inherited]`

Construct an array with an [initializer_list](#) of values.

Definition at line 648 of file valarray.

2.22.2.17 `template<typename _Tp> std::valarray<_Tp>::valarray (size_t`
`__n) [inline, explicit, inherited]`

Construct an array with n elements.

Definition at line 581 of file valarray.

2.22.2.18 `template<typename _Tp> std::valarray<_Tp>::valarray (const`
`_Tp & __t, size_t __n) [inline, inherited]`

Construct an array with n elements initialized to t .

Definition at line 587 of file valarray.

2.22.2.19 `std::gslice::~gslice () [inline, inherited]`

Destructor.

Definition at line 161 of file gslice.h.

2.22.2.20 `template<class _Tp> _Expr<_ValFunClos<_ValArray, _Tp>, _Tp`
`> std::valarray<_Tp>::apply (_Tp func_Tp) const [inline,`
`inherited]`

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying `func` to the corresponding element of this array. The new array has the same size as this one.

Parameters

func Function of Tp returning Tp to apply.

Returns

New valarray with transformed elements.

Definition at line 971 of file valarray.

2.22.2.21 `template<class _Tp> _Expr<_RefFuncClos<_ValArray, _Tp>, _Tp> std::valarray<_Tp>::apply (_Tp funcconst _Tp &) const`
[inline, inherited]

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

Parameters

func Function of const Tp& returning Tp to apply.

Returns

New valarray with transformed elements.

Definition at line 979 of file valarray.

2.22.2.22 `template<class _Tp> _Tp* std::begin (valarray<_Tp> & __va)`
[inline]

Return an iterator pointing to the first element of the valarray.

Parameters

va valarray.

Definition at line 1118 of file valarray.

2.22.2.23 `template<class _Tp> const _Tp* std::begin (const valarray<_Tp> & __va)`
[inline]

Return an iterator pointing to the first element of the const valarray.

Parameters

va valarray.

Definition at line 1128 of file valarray.

2.22.2.24 `template<class _Tp> valarray<_Tp> std::valarray<_Tp>::cshift (int __n) const`
[inline, inherited]

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index i , the new position is $(i - n) \% \text{size}()$. The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

Parameters

n Number of element positions to rotate.

Returns

New valarray with elements in shifted positions.

Definition at line 897 of file valarray.

2.22.2.25 `template<class _Tp> const _Tp* std::end (const valarray< _Tp > & __va) [inline]`

Return an iterator pointing to one past the last element of the const valarray.

Parameters

va valarray.

Definition at line 1148 of file valarray.

References `std::valarray< _Tp >::size()`.

2.22.2.26 `template<class _Tp> _Tp* std::end (valarray< _Tp > & __va) [inline]`

Return an iterator pointing to one past the last element of the valarray.

Parameters

va valarray.

Definition at line 1138 of file valarray.

References `std::valarray< _Tp >::size()`.

2.22.2.27 `template<typename _Tp > _Tp std::valarray< _Tp >::max ()`
`const [inline, inherited]`

Return the maximum element using operator<().

Definition at line 963 of file valarray.

References std::max_element().

2.22.2.28 `template<typename _Tp > _Tp std::valarray< _Tp >::min ()`
`const [inline, inherited]`

Return the minimum element using operator<().

Definition at line 955 of file valarray.

References std::min_element().

2.22.2.29 `template<typename _Tp > valarray< _Tp >::template _UnaryOp<`
`__logical_not >::Rt std::valarray< _Tp >::operator! () const`
`[inline, inherited]`

Return a new valarray by applying unary ! to each element.

Definition at line 998 of file valarray.

2.22.2.30 `template<typename _Tp > void std::gslice_array< _Tp`
`>::operator%=(const valarray< _Tp > & __v) const [inline,`
`inherited]`

Modulo slice elements by corresponding elements of v.

Definition at line 201 of file gslice_array.h.

2.22.2.31 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp`
`>::operator%=(const _Tp & __t) [inline, inherited]`

Set each element e of array to e % t.

Definition at line 1025 of file valarray.

2.22.2.32 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp`
`>::operator%=(const valarray< _Tp > & __v) [inline,`
`inherited]`

Modulo elements of array by corresponding elements of v.

Definition at line 1025 of file valarray.

2.22.2.33 `template<typename _Tp> void std::gslice_array<_Tp>::operator&= (const valarray<_Tp> & __v) const [inline, inherited]`

Logical and slice elements with corresponding elements of v .

Definition at line 205 of file gslice_array.h.

2.22.2.34 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator&= (const _Tp & __t) [inline, inherited]`

Set each element e of array to $e \& t$.

Definition at line 1027 of file valarray.

2.22.2.35 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator&= (const valarray<_Tp> & __v) [inline, inherited]`

Logical and corresponding elements of v with elements of array.

Definition at line 1027 of file valarray.

2.22.2.36 `template<typename _Tp> void std::gslice_array<_Tp>::operator*= (const valarray<_Tp> & __v) const [inline, inherited]`

Multiply slice elements by corresponding elements of v .

Definition at line 199 of file gslice_array.h.

2.22.2.37 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator*= (const valarray<_Tp> & __v) [inline, inherited]`

Multiply elements of array by corresponding elements of v .

Definition at line 1023 of file valarray.

2.22.2.38 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator*=(const _Tp & __t) [inline, inherited]`

Multiply each element of array by *t*.

Definition at line 1023 of file valarray.

2.22.2.39 `template<typename _Tp > valarray< _Tp >::template _UnaryOp< __unary_plus >::_Rt std::valarray< _Tp >::operator+() const [inline, inherited]`

Return a new valarray by applying unary + to each element.

Definition at line 995 of file valarray.

2.22.2.40 `template<typename _Tp > void std::gslice_array< _Tp >::operator+=(const valarray< _Tp > & __v) const [inline, inherited]`

Add corresponding elements of *v* to slice elements.

Definition at line 202 of file gslice_array.h.

2.22.2.41 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator+=(const _Tp & __t) [inline, inherited]`

Add *t* to each element of array.

Definition at line 1021 of file valarray.

2.22.2.42 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp >::operator+=(const valarray< _Tp > & __v) [inline, inherited]`

Add corresponding elements of *v* to elements of array.

Definition at line 1021 of file valarray.

2.22.2.43 `template<typename _Tp > valarray< _Tp >::template _UnaryOp< __negate >::_Rt std::valarray< _Tp >::operator-() const [inline, inherited]`

Return a new valarray by applying unary - to each element.

Definition at line 996 of file valarray.

2.22.2.44 `template<typename _Tp > void std::gslice_array< _Tp
>::operator-= (const valarray< _Tp > & __v) const [inline,
inherited]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 203 of file `gslice_array.h`.

2.22.2.45 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator-= (const valarray< _Tp > & __v) [inline,
inherited]`

Subtract corresponding elements of *v* from elements of array.

Definition at line 1022 of file `valarray`.

2.22.2.46 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator-= (const _Tp & __t) [inline, inherited]`

Subtract *t* to each element of array.

Definition at line 1022 of file `valarray`.

2.22.2.47 `template<typename _Tp > void std::gslice_array< _Tp
>::operator/= (const valarray< _Tp > & __v) const [inline,
inherited]`

Divide slice elements by corresponding elements of *v*.

Definition at line 200 of file `gslice_array.h`.

2.22.2.48 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator/= (const _Tp & __t) [inline, inherited]`

Divide each element of array by *t*.

Definition at line 1024 of file `valarray`.

2.22.2.49 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator/= (const valarray< _Tp > & __v) [inline,
inherited]`

Divide elements of array by corresponding elements of *v*.

Definition at line 1024 of file `valarray`.

2.22.2.50 `template<typename _Tp > void std::gslice_array< _Tp
>::operator<<= (const valarray< _Tp > & __v) const
[inline, inherited]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 207 of file `gslice_array.h`.

2.22.2.51 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator<<= (const _Tp & __t) [inline, inherited]`

Left shift each element *e* of array by *t* bits.

Definition at line 1029 of file `valarray`.

2.22.2.52 `template<class _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator<<= (const valarray< _Tp > & __v) [inline,
inherited]`

Left shift elements of array by corresponding elements of *v*.

Definition at line 1029 of file `valarray`.

2.22.2.53 `template<typename _Tp > void std::indirect_array< _Tp
>::operator= (const _Tp & __t) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 162 of file `indirect_array.h`.

2.22.2.54 `template<typename _Tp> valarray< _Tp > & std::valarray< _Tp
>::operator= (const valarray< _Tp > & __v) [inline,
inherited]`

Assign elements to an array.

Assign elements of array to values in *v*. Results are undefined if *v* does not have the same size as this array.

Parameters

v Valarray to get values from.

Definition at line 669 of file `valarray`.

2.22.2.55 `template<typename _Tp> void std::slice_array<_Tp>::operator=(const valarray<_Tp> & __v) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 228 of file slice_array.h.

2.22.2.56 `gslice & std::gslice::operator=(const gslice & __g) [inline, inherited]`

Assignment operator.

Definition at line 168 of file gslice.h.

2.22.2.57 `template<typename _Tp> void std::mask_array<_Tp>::operator=(const _Tp & __t) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 157 of file mask_array.h.

2.22.2.58 `template<typename _Tp> gslice_array<_Tp> & std::gslice_array<_Tp>::operator=(const gslice_array<_Tp> & __a) [inline, inherited]`

Assignment operator. Assigns slice elements to corresponding /// elements of *a*.

Definition at line 147 of file gslice_array.h.

References `std::valarray<_Tp>::size()`.

2.22.2.59 `template<typename _Tp> void std::gslice_array<_Tp>::operator=(const valarray<_Tp> & __v) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 165 of file gslice_array.h.

References `std::valarray<_Tp>::size()`.

2.22.2.60 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=(const _Tp & __t) [inline, inherited]`

Assign elements to a value.

Assign all elements of array to *t*.

Parameters

t Value for elements.

Definition at line 717 of file valarray.

2.22.2.61 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=(const slice_array<_Tp> & __sa) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

Parameters

sa Array slice to get values from.

Definition at line 725 of file valarray.

2.22.2.62 `template<typename _Tp> indirect_array<_Tp> & std::indirect_array<_Tp>::operator=(const indirect_array<_Tp> & __a) [inline, inherited]`

Assignment operator. Assigns elements to corresponding elements /// of *a*.

Definition at line 153 of file indirect_array.h.

2.22.2.63 `template<typename _Tp> void std::indirect_array<_Tp>::operator=(const valarray<_Tp> & __v) const [inline, inherited]`

Assign slice elements to corresponding elements of *v*.

Definition at line 167 of file indirect_array.h.

2.22.2.64 `template<typename _Tp> mask_array<_Tp> & std::mask_array<_Tp>::operator=(const mask_array<_Tp> & __a) [inline, inherited]`

Assignment operator. Assigns elements to corresponding elements /// of *a*.

Definition at line 148 of file mask_array.h.

2.22.2.65 `template<typename _Tp> slice_array<_Tp> & std::slice_array<_Tp>::operator= (const slice_array<_Tp> & __a) [inline, inherited]`

Assignment operator. Assigns slice elements to corresponding /// elements of *a*.

Definition at line 214 of file slice_array.h.

2.22.2.66 `template<typename _Tp> void std::slice_array<_Tp>::operator= (const _Tp & __t) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 223 of file slice_array.h.

2.22.2.67 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator= (const mask_array<_Tp> & __ma) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

Parameters

ma Array slice to get values from.

Definition at line 745 of file valarray.

2.22.2.68 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator= (const indirect_array<_Tp> & __ia) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

Parameters

ia Array slice to get values from.

Definition at line 755 of file valarray.

2.22.2.69 `template<typename _Tp> void std::gslice_array<_Tp>::operator=(const _Tp & __t) const [inline, inherited]`

Assign all slice elements to *t*.

Definition at line 157 of file `gslice_array.h`.

References `std::valarray<_Tp>::size()`.

2.22.2.70 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=(const gslice_array<_Tp> & __ga) [inline, inherited]`

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

Parameters

ga Array slice to get values from.

Definition at line 735 of file `valarray`.

References `std::valarray<_Tp>::size()`.

2.22.2.71 `template<typename _Tp> valarray<_Tp> & std::valarray<_Tp>::operator=(initializer_list<_Tp> __l) [inline, inherited]`

Assign elements to an [initializer_list](#).

Assign elements of array to values in *l*. Results are undefined if *l* does not have the same size as this array.

Parameters

l [initializer_list](#) to get values from.

Definition at line 693 of file `valarray`.

2.22.2.72 `template<typename _Tp> void std::gslice_array<_Tp>::operator>>= (const valarray<_Tp> & __v) const [inline, inherited]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 208 of file `gslice_array.h`.

2.22.2.73 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator>>= (const valarray<_Tp> & __v) [inline, inherited]`

Right shift elements of array by corresponding elements of *v*.

Definition at line 1030 of file valarray.

2.22.2.74 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator>>= (const _Tp & __t) [inline, inherited]`

Right shift each element *e* of array by *t* bits.

Definition at line 1030 of file valarray.

2.22.2.75 `template<typename _Tp> slice_array<_Tp> std::valarray<_Tp>::operator[] (slice __s) [inline, inherited]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See also

[slice](#).

Parameters

s The source slice.

Returns

New valarray containing elements in *s*.

Definition at line 782 of file valarray.

2.22.2.76 `template<typename _Tp> _Expr<_GClos<_ValArray, _Tp>, _Tp> & std::valarray<_Tp>::operator[] (const gslice & __gs) const [inline, inherited]`

Return an array subset.

Returns a [slice_array](#) referencing the elements of the array indicated by the slice argument.

See also

[gslice](#).

Parameters

s The source slice.

Returns

Slice_array referencing elements indicated by *s*.

Definition at line 787 of file valarray.

2.22.2.77 `template<typename _Tp > valarray< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > & __m) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

Parameters

m The valarray bitmask.

Returns

New valarray containing elements indicated by *m*.

Definition at line 804 of file valarray.

References std::valarray< _Tp >::size().

2.22.2.78 `template<typename _Tp > mask_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > & __m) [inline, inherited]`

Return a reference to an array subset.

Returns a new [mask_array](#) referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

Parameters

m The valarray bitmask.

Returns

New valarray containing elements indicated by *m*.

Definition at line 816 of file valarray.

References `std::valarray<_Tp>::size()`.

2.22.2.79 `template<typename _Tp > _Expr< _IClos< _ValArray, _Tp >, _Tp
> std::valarray< _Tp >::operator[] (const valarray< size_t > &
__i) const [inline, inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

Parameters

i The valarray element index list.

Returns

New valarray containing elements in *s*.

Definition at line 827 of file valarray.

2.22.2.80 `template<typename _Tp > indirect_array< _Tp > std::valarray<
_Tp >::operator[] (const valarray< size_t > & __i) [inline,
inherited]`

Return a reference to an array subset.

Returns an [indirect_array](#) referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned [indirect_array](#) refers to these elements.

Parameters

i The valarray element index list.

Returns

Indirect_array referencing elements in *i*.

Definition at line 835 of file valarray.

References `std::valarray<_Tp>::size()`.

2.22.2.81 `template<typename _Tp > _Tp & std::valarray< _Tp >::operator[]
(size_t __i) [inline, inherited]`

Return a reference to the *i*'th array element.

Parameters

i Index of element to return.

Returns

Reference to the *i*'th element.

Definition at line 551 of file valarray.

2.22.2.82 `template<typename _Tp > gslice_array< _Tp > std::valarray< _Tp
>::operator[] (const gslice & __gs) [inline, inherited]`

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the *gslice* argument. The new valarray has the same size as the input *gslice*.

See also

[gslice](#).

Parameters

s The source *gslice*.

Returns

New valarray containing elements in *s*.

Definition at line 796 of file valarray.

2.22.2.83 `template<typename _Tp > _Expr< _SClos< _ValArray, _Tp >, _Tp
> std::valarray< _Tp >::operator[] (slice __s) const [inline,
inherited]`

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the *slice* argument. The new valarray has the same size as the input *slice*.

See also

[slice](#).

Parameters

s The source slice.

Returns

New valarray containing elements in *s*.

Definition at line 774 of file valarray.

2.22.2.84 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator^=(const _Tp & __t) [inline, inherited]`

Set each element *e* of array to $e \wedge t$.

Definition at line 1026 of file valarray.

2.22.2.85 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator^=(const valarray<_Tp> & __v) [inline,
inherited]`

Logical xor corresponding elements of *v* with elements of array.

Definition at line 1026 of file valarray.

2.22.2.86 `template<typename _Tp> void std::gslice_array<_Tp>
>::operator^=(const valarray<_Tp> & __v) const [inline,
inherited]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 204 of file gslice_array.h.

2.22.2.87 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>
>::operator|=(const valarray<_Tp> & __v) [inline,
inherited]`

Logical or corresponding elements of *v* with elements of array.

Definition at line 1028 of file valarray.

2.22.2.88 `template<typename _Tp> void std::gslice_array<_Tp>
>::operator|=(const valarray<_Tp> & __v) const [inline,
inherited]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 206 of file `gslice_array.h`.

2.22.2.89 `template<class _Tp> valarray<_Tp> & std::valarray<_Tp>::operator|= (const _Tp & __t) [inline, inherited]`

Set each element *e* of array to *e* | *t*.

Definition at line 1028 of file `valarray`.

2.22.2.90 `template<typename _Tp> valarray<_Tp>::template _UnaryOp<__bitwise_not>::Rt std::valarray<_Tp>::operator~ () const [inline, inherited]`

Return a new `valarray` by applying unary `~` to each element.

Definition at line 997 of file `valarray`.

2.22.2.91 `template<class _Tp> void std::valarray<_Tp>::resize (size_t __size, _Tp __c = _Tp()) [inline, inherited]`

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

Parameters

size New array size.

c New value for all elements.

Definition at line 938 of file `valarray`.

2.22.2.92 `template<class _Tp> valarray<_Tp> std::valarray<_Tp>::shift (int __n) const [inline, inherited]`

Return a shifted array.

A new `valarray` is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is *i* - *n*. The new `valarray` has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, *n*). Negative arguments discard elements from the top of the array.

Parameters

n Number of element positions to shift.

Returns

New valarray with elements in shifted positions.

Definition at line 856 of file valarray.

2.22.2.93 `size_t std::slice::size () const [inline, inherited]`

Return size of slice.

Definition at line 101 of file slice_array.h.

2.22.2.94 `valarray< size_t > std::gslice::size () const [inline, inherited]`

Return array of sizes of slice dimensions.

Definition at line 137 of file gslice.h.

2.22.2.95 `template<class _Tp> size_t std::valarray< _Tp >::size () const [inline, inherited]`

Return the number of elements in array.

Definition at line 843 of file valarray.

Referenced by `std::end()`, `std::valarray< _Tp >::operator=()`, `std::gslice_array< _Tp >::operator=()`, and `std::valarray< _Tp >::operator[]()`.

2.22.2.96 `size_t std::gslice::start () const [inline, inherited]`

Return array offset of first slice element.

Definition at line 133 of file gslice.h.

2.22.2.97 `size_t std::slice::start () const [inline, inherited]`

Return array offset of first slice element.

Definition at line 97 of file slice_array.h.

2.22.2.98 `size_t std::slice::stride () const` **[inline, inherited]**

Return array stride of slice.

Definition at line 105 of file slice_array.h.

2.22.2.99 `valarray< size_t > std::gslice::stride () const` **[inline, inherited]**

Return array of array strides for each dimension.

Definition at line 141 of file gslice.h.

2.22.2.100 `template<class _Tp > _Tp std::valarray< _Tp >::sum () const` **[inline, inherited]**

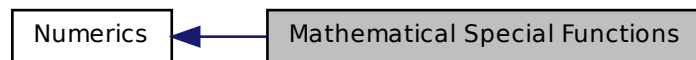
Return the sum of all elements in the array.

Accumulates the sum of all elements into a Tp using +=. The order of adding the elements is unspecified.

Definition at line 848 of file valarray.

2.23 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



Functions

- `template<typename _Tp > __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`

- long double **std::tr1::assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::assoc_legendre** (unsigned int __l, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **std::tr1::assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpx, typename _Tpy >
__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type **std::tr1::beta** (_Tpx __x, _Tpy __y)
- float **std::tr1::betaf** (float __x, float __y)
- long double **std::tr1::betal** (long double __x, long double __y)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::comp_ellint_1** (_Tp __k)
- float **std::tr1::comp_ellint_1f** (float __k)
- long double **std::tr1::comp_ellint_1l** (long double __k)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::comp_ellint_2** (_Tp __k)
- float **std::tr1::comp_ellint_2f** (float __k)
- long double **std::tr1::comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn >
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type **std::tr1::comp_ellint_3** (_Tp __k, _Tpn __nu)
- float **std::tr1::comp_ellint_3f** (float __k, float __nu)
- long double **std::tr1::comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp >
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type **std::tr1::conf_hyperg** (_Tpa __a, _Tpc __c, _Tp __x)
- float **std::tr1::conf_hypergf** (float __a, float __c, float __x)
- long double **std::tr1::conf_hypergl** (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **std::tr1::cyl_bessel_i** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_if** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **std::tr1::cyl_bessel_j** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)

- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (_-`
`Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl_bessel_kf (float __nu, float __x)`
- `long double std::tr1::cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (_-`
`Tpnu __nu, _Tp __x)`
- `float std::tr1::cyl_neumannf (float __nu, float __x)`
- `long double std::tr1::cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _-`
`Tpp __phi)`
- `float std::tr1::ellint_1f (float __k, float __phi)`
- `long double std::tr1::ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _-`
`Tpp __phi)`
- `float std::tr1::ellint_2f (float __k, float __phi)`
- `long double std::tr1::ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp >`
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp`
`__k, _Tpn __nu, _Tpp __phi)`
- `float std::tr1::ellint_3f (float __k, float __nu, float __phi)`
- `long double std::tr1::ellint_3l (long double __k, long double __nu, long double`
`__phi)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)`
- `float std::tr1::expintf (float __x)`
- `long double std::tr1::expintl (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp`
`__x)`
- `float std::tr1::hermitef (unsigned int __n, float __x)`
- `long double std::tr1::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg`
`(_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float std::tr1::hypergf (float __a, float __b, float __c, float __x)`
- `long double std::tr1::hypergl (long double __a, long double __b, long double`
`__c, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _-`
`Tp __x)`

- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::legendre** (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::riemann_zeta** (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_bessel** (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_legendre** (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_neumann** (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

2.23.1 Detailed Description

A collection of advanced mathematical special functions.

2.23.2 Function Documentation

2.23.2.1 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp
__x) [inline]`

5.2.1.1 Associated Laguerre polynomials.

Definition at line 123 of file tr1/cmath.

2.23.2.2 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::assoc_legendre(unsigned int __l, unsigned int __m, _Tp
__x) [inline]`

5.2.1.2 Associated Legendre functions.

Definition at line 140 of file tr1/cmath.

2.23.2.3 `template<typename _Tpx , typename _Tpy >
__gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta(_Tpx
__x, _Tpy __y) [inline]`

5.2.1.3 Beta functions.

Definition at line 157 of file tr1/cmath.

2.23.2.4 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::comp_ellint_1(_Tp __k) [inline]`

5.2.1.4 Complete elliptic integrals of the first kind.

Definition at line 174 of file tr1/cmath.

2.23.2.5 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::comp_ellint_2(_Tp __k) [inline]`

5.2.1.5 Complete elliptic integrals of the second kind.

Definition at line 191 of file tr1/cmath.

2.23.2.6 `template<typename _Tp , typename _Tpn > __gnu_cxx::__promote_
2<_Tp, _Tpn>::__type std::tr1::comp_ellint_3(_Tp __k, _Tpn
__nu) [inline]`

5.2.1.6 Complete elliptic integrals of the third kind.

Definition at line 208 of file tr1/cmath.

2.23.2.7 `template<typename _Tpa , typename _Tpc , typename _Tp
> __gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type
std::tr1::conf_hyperg(_Tpa __a, _Tpc __c, _Tp __x) [inline]`

5.2.1.7 Confluent hypergeometric functions.

Definition at line 225 of file tr1/cmath.

2.23.2.8 `template<typename _Tpnu , typename _Tp >`
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_i`
`(_Tpnu __nu, _Tp __x) [inline]`

5.2.1.8 Regular modified cylindrical Bessel functions.

Definition at line 242 of file tr1/cmath.

2.23.2.9 `template<typename _Tpnu , typename _Tp >`
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j`
`(_Tpnu __nu, _Tp __x) [inline]`

5.2.1.9 Cylindrical Bessel functions (of the first kind).

Definition at line 259 of file tr1/cmath.

2.23.2.10 `template<typename _Tpnu , typename _Tp >`
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type`
`std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x) [inline]`

5.2.1.10 Irregular modified cylindrical Bessel functions.

Definition at line 276 of file tr1/cmath.

2.23.2.11 `template<typename _Tpnu , typename _Tp >`
`__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type`
`std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x) [inline]`

5.2.1.11 Cylindrical Neumann functions.

Definition at line 293 of file tr1/cmath.

2.23.2.12 `template<typename _Tp , typename _Tpp >`
`__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_1 (`
`_Tp __k, _Tpp __phi) [inline]`

5.2.1.12 Incomplete elliptic integrals of the first kind.

Definition at line 310 of file tr1/cmath.

2.23.2.13 `template<typename _Tp , typename _Tpp >`
`__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_2 (`
`_Tp __k, _Tpp __phi) [inline]`

5.2.1.13 Incomplete elliptic integrals of the second kind.

Definition at line 327 of file tr1/cmath.

2.23.2.14 `template<typename _Tp , typename _Tpn , typename _Tpp
> __gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type
std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi) [inline]`

5.2.1.14 Incomplete elliptic integrals of the third kind.

Definition at line 344 of file tr1/cmath.

2.23.2.15 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::expint (_Tp __x) [inline]`

5.2.1.15 Exponential integrals.

Definition at line 361 of file tr1/cmath.

2.23.2.16 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::hermite (unsigned int __n, _Tp __x) [inline]`

5.2.1.16 Hermite polynomials.

Definition at line 378 of file tr1/cmath.

2.23.2.17 `template<typename _Tpa , typename _Tpb , typename _Tpc ,
typename _Tp > __gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc,
_Tp>::__type std::tr1::hyperg (_Tpa __a, _Tpb __b, _Tpc __c,
_Tp __x) [inline]`

5.2.1.17 Hypergeometric functions.

Definition at line 395 of file tr1/cmath.

2.23.2.18 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::laguerre (unsigned int __n, _Tp __x) [inline]`

5.2.1.18 Laguerre polynomials.

Definition at line 412 of file tr1/cmath.

2.23.2.19 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::legendre (unsigned int __n, _Tp __x) [inline]`

5.2.1.19 Legendre polynomials.

Definition at line 429 of file tr1/cmath.

2.23.2.20 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::riemann_zeta (_Tp __x) [inline]`

5.2.1.20 Riemann zeta function.

Definition at line 446 of file tr1/cmath.

2.23.2.21 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::sph_bessel (unsigned int __n, _Tp __x) [inline]`

5.2.1.21 Spherical Bessel functions.

Definition at line 463 of file tr1/cmath.

2.23.2.22 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp
__theta) [inline]`

5.2.1.22 Spherical associated Legendre functions.

Definition at line 480 of file tr1/cmath.

2.23.2.23 `template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type
std::tr1::sph_neumann (unsigned int __n, _Tp __x) [inline]`

5.2.1.23 Spherical Neumann functions.

Definition at line 497 of file tr1/cmath.

2.24 Type Traits

Collaboration diagram for Type Traits:



Classes

- struct `std::__declval_protector< _Tp >`
declval
- struct `std::add_lvalue_reference< _Tp >`
add_lvalue_reference
- struct `std::add_rvalue_reference< _Tp >`
add_rvalue_reference
- struct `std::aligned_storage< _Len, _Align >`
Alignment type.
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
conditional
- class `std::decay< _Tp >`
decay
- struct `std::enable_if< bool, _Tp >`
enable_if
- struct `std::has_nothrow_copy_assign< _Tp >`
has_nothrow_copy_assign
- struct `std::has_nothrow_copy_constructor< _Tp >`
has_nothrow_copy_constructor

- struct `std::has_nothrow_default_constructor< _Tp >`
has_nothrow_default_constructor
- struct `std::has_trivial_copy_assign< _Tp >`
has_trivial_copy_assign
- struct `std::has_trivial_copy_constructor< _Tp >`
has_trivial_copy_constructor
- struct `std::has_trivial_default_constructor< _Tp >`
has_trivial_default_constructor
- struct `std::has_trivial_destructor< _Tp >`
has_trivial_destructor
- struct `std::is_base_of< _Base, _Derived >`
is_base_of
- struct `std::is_constructible< _Tp, _Args >`
is_constructible
- struct `std::is_convertible< _From, _To >`
is_convertible
- struct `std::is_explicitly_convertible< _From, _To >`
is_explicitly_convertible
- struct `std::is_lvalue_reference< typename >`
is_lvalue_reference
- struct `std::is_nothrow_constructible< _Tp, _Args >`
is_nothrow_constructible
- struct `std::is_pod< _Tp >`
is_pod
- struct `std::is_reference< _Tp >`
is_reference
- struct `std::is_rvalue_reference< typename >`
is_rvalue_reference

- struct `std::is_signed< _Tp >`
is_signed
- struct `std::is_standard_layout< _Tp >`
is_standard_layout
- struct `std::is_trivial< _Tp >`
is_trivial
- struct `std::is_unsigned< _Tp >`
is_unsigned
- struct `std::make_signed< _Tp >`
make_signed
- struct `std::make_unsigned< _Tp >`
make_unsigned
- struct `std::remove_reference< _Tp >`
remove_reference
- struct `std::tr1::__is_member_pointer_helper< _Tp >`
is_member_pointer
- struct `std::tr1::add_const< _Tp >`
add_const
- struct `std::tr1::add_cv< _Tp >`
add_cv
- struct `std::tr1::add_pointer< _Tp >`
add_pointer
- struct `std::tr1::add_volatile< _Tp >`
add_volatile
- struct `std::tr1::alignment_of< _Tp >`
alignment_of
- struct `std::tr1::extent< typename, _UInt >`
extent

- struct `std::tr1::has_virtual_destructor< _Tp >`
has_virtual_destructor
- struct `std::tr1::integral_constant< _Tp, __v >`
integral_constant
- struct `std::tr1::is_abstract< _Tp >`
is_abstract
- struct `std::tr1::is_arithmetic< _Tp >`
is_arithmetic
- struct `std::tr1::is_array< typename >`
is_array
- struct `std::tr1::is_class< _Tp >`
is_class
- struct `std::tr1::is_compound< _Tp >`
is_compound
- struct `std::tr1::is_const< typename >`
is_const
- struct `std::tr1::is_empty< _Tp >`
is_empty
- struct `std::tr1::is_enum< _Tp >`
is_enum
- struct `std::tr1::is_floating_point< _Tp >`
is_floating_point
- struct `std::tr1::is_function< typename >`
is_function
- struct `std::tr1::is_fundamental< _Tp >`
is_fundamental
- struct `std::tr1::is_integral< _Tp >`
is_integral

- struct `std::tr1::is_member_function_pointer< _Tp >`
is_member_function_pointer
- struct `std::tr1::is_member_object_pointer< _Tp >`
is_member_object_pointer
- struct `std::tr1::is_object< _Tp >`
is_object
- struct `std::tr1::is_pointer< _Tp >`
is_pointer
- struct `std::tr1::is_polymorphic< _Tp >`
is_polymorphic
- struct `std::tr1::is_same< typename, typename >`
is_same
- struct `std::tr1::is_scalar< _Tp >`
is_scalar
- struct `std::tr1::is_union< _Tp >`
is_union
- struct `std::tr1::is_void< _Tp >`
is_void
- struct `std::tr1::is_volatile< typename >`
is_volatile
- struct `std::tr1::rank< typename >`
rank
- struct `std::tr1::remove_all_extents< _Tp >`
remove_all_extents
- struct `std::tr1::remove_const< _Tp >`
remove_const
- struct `std::tr1::remove_cv< _Tp >`
remove_cv

- struct `std::tr1::remove_extent< _Tp >`
remove_extent
- struct `std::tr1::remove_pointer< _Tp >`
remove_pointer
- struct `std::tr1::remove_volatile< _Tp >`
remove_volatile

Defines

- `#define _DEFINE_SPEC(_Order, _Trait, _Type, _Value)`
- `#define _DEFINE_SPEC_0_HELPER`
- `#define _DEFINE_SPEC_1_HELPER`
- `#define _DEFINE_SPEC_2_HELPER`
- `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`

Typedefs

- `typedef integral_constant< bool, false > std::tr1::false_type`
- `typedef integral_constant< bool, true > std::tr1::true_type`

Functions

- `template<typename _Tp >`
`add_rvalue_reference< _Tp >::type std::declval () noexcept`

Variables

- `static const _Tp std::tr1::integral_constant::value`

2.24.1 Detailed Description

Compile time type transformation and information.

2.24.2 Define Documentation

2.24.2.1 `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`

Use SFINAE to determine if the type `_Tp` has a publicly-accessible member type `_NTYPE`.

Definition at line 703 of file `type_traits`.

2.24.3 Typedef Documentation

2.24.3.1 `typedef integral_constant<bool, false> std::tr1::false_type`

typedef for `false_type`

Definition at line 78 of file `tr1_impl/type_traits`.

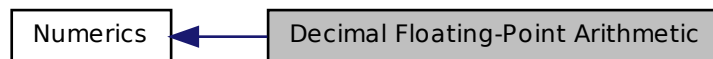
2.24.3.2 `typedef integral_constant<bool, true> std::tr1::true_type`

typedef for `true_type`

Definition at line 75 of file `tr1_impl/type_traits`.

2.25 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



Namespaces

- namespace [std::decimal](#)

2.25.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

2.26 Binder Classes

Collaboration diagram for Binder Classes:



Classes

- class `std::binder1st< _Operation >`
One of the [binder functors](#).
- class `std::binder2nd< _Operation >`
One of the [binder functors](#).
- struct `std::is_bind_expression< _Tp >`
Determines if the given type `_Tp` is a function object should be treated as a sub-expression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].
- struct `std::is_bind_expression< _Bind< _Signature > >`
Class template `_Bind` is always a bind expression.
- struct `std::is_bind_expression< _Bind_result< _Result, _Signature > >`
Class template `_Bind` is always a bind expression.
- struct `std::is_placeholder< _Tp >`
Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].
- struct `std::is_placeholder< _Placeholder< _Num > >`

Namespaces

- namespace [std::placeholders](#)

Functions

- `template<typename _Functor, typename... _ArgTypes>
_Bind_helper< _Functor, _ArgTypes...>::type std::bind (_Functor &&__f, _-
_ArgTypes &&...__args)`
- `template<typename _Operation, typename _Tp >
binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__-
_x)`
- `template<typename _Operation, typename _Tp >
binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__-
_x)`

Variables

- [std::binder1st](#) [std::_GLIBCXX_DEPRECATED_ATTR](#)

2.26.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type [binder1st](#) is constructed from a functor `f` and an argument `x`. Later, `B's operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1, y2, ...`) and will in turn call `f(x, y1), f(x, y2), ...`

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of [binder1st](#).

The type [binder2nd](#) and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>, 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform `1.3 - x` instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>, 5)`.

These become more useful when combined with the composition functions.

2.26.2 Function Documentation

2.26.2.1 `template<typename _Functor , typename... _ArgTypes>
_Bind_helper<_Functor, _ArgTypes...>::type std::bind (_Functor
&& __f, _ArgTypes &&... __args) [inline]`

Function template for `std::bind`.

Definition at line 1430 of file `functional`.

2.26.2.2 `template<typename _Operation , typename _Tp >
binder1st<_Operation> std::bind1st (const _Operation & __fn,
const _Tp & __x) [inline]`

One of the [binder functors](#).

Definition at line 125 of file `binders.h`.

2.26.2.3 `template<typename _Operation , typename _Tp >
binder2nd<_Operation> std::bind2nd (const _Operation & __fn,
const _Tp & __x) [inline]`

One of the [binder functors](#).

Definition at line 160 of file `binders.h`.

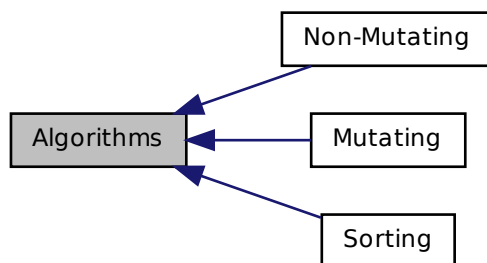
2.26.3 Variable Documentation

2.26.3.1 `std::binder2nd std::_GLIBCXX_DEPRECATED_ATTR`

One of the [binder functors](#).

2.27 Algorithms

Collaboration diagram for Algorithms:



Modules

- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

2.27.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

2.28 Mutating

Collaboration diagram for Mutating:



Functions

- `template<typename _II, typename _OI >`
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator`
`__result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`
`value)`
- `template<typename _OI, typename _Size, typename _Tp >`
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator`
`__gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator`
`__gen)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate`
`__pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`
`_OI std::move (_II __first, _II __last, _OI __result)`

- `template<typename _Tp >`
`std::remove_reference< _Tp >::type && std::move (_Tp &&__t)`
- `template<typename _BI1 , typename _BI2 >`
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`
- `template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , type-`
`name _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator`
`__first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __-`
`out_false, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator`
`__last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::random_shuffle (_RandomAccessIterator __first, __-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, __-`
`RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator , typename _Tp >`
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Tp >`
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __-`
`last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`
- `template<typename _ForwardIterator , typename _Tp >`
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp`
`&__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _-`
`Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __-`
`last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator , typename _Predicate , typename _Tp >`
`void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _-`
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`

- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator std::reverse_copy (_BidirectionalIterator __first, _-`
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __-`
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _-`
`ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _-`
`ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _BinaryOperation >`
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_-`
`op)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last,`
`_BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, _BinaryPredicate __binary_pred)`

2.28.1 Function Documentation

2.28.1.1 `template<typename _II, typename _OI> _OI std::copy (_II __first, _II __last, _OI __result) [inline]`

Copies the range [first,last) into result.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

result + (first - last)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 442 of file `stl_algobase.h`.

2.28.1.2 `template<typename _BI1, typename _BI2> _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result) [inline]`

Copies the range [first,last) into result.

Parameters

first A bidirectional iterator.

last A bidirectional iterator.

result A bidirectional iterator.

Returns

result - (first - last)

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are

passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use copy instead. Note that the start of the output range may overlap [first,last).

Definition at line 611 of file stl_algobase.h.

2.28.1.3 `template<typename _InputIterator , typename _OutputIterator ,
typename _Predicate > _OutputIterator std::copy_if (_InputIterator
__first, _InputIterator __last, _OutputIterator __result, _Predicate
__pred)`

Copy the elements of a sequence for which a predicate is true.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

pred A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [first,last) for which `pred` returns true to the range beginning at `result`.

`copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 949 of file stl_algo.h.

2.28.1.4 `template<typename _InputIterator , typename _Size , typename
_OutputIterator > _OutputIterator std::copy_n (_InputIterator
__first, _Size __n, _OutputIterator __result) [inline]`

Copies the range [first,first+n) into [result,result+n).

Parameters

first An input iterator.

n The number of elements to copy.

result An output iterator.

Returns

result+n.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 1006 of file `stl_algo.h`.

References `std::__iterator_category()`.

2.28.1.5 `template<typename _ForwardIterator, typename _Tp> void std::fill
(_ForwardIterator __first, _ForwardIterator __last, const _Tp &
__value) [inline]`

Fills the range `[first,last)` with copies of `value`.

Parameters

first A forward iterator.

last A forward iterator.

value A reference-to-const of arbitrary type.

Returns

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 713 of file `stl_algobase.h`.

2.28.1.6 `template<typename _OI, typename _Size, typename _Tp> _OI
std::fill_n (_OI __first, _Size __n, const _Tp & __value)
[inline]`

Fills the range `[first,first+n)` with copies of `value`.

Parameters

first An output iterator.

n The count of copies to perform.

value A reference-to-const of arbitrary type.

Returns

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `@ wmemset`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 773 of file `stl_algobase.h`.

2.28.1.7 `template<typename _ForwardIterator, typename _Generator> void
std::generate (_ForwardIterator __first, _ForwardIterator __last,
_Generator __gen)`

Assign the result of a function object to each value in a sequence.

Parameters

first A forward iterator.

last A forward iterator.

gen A function object taking no arguments and returning `std::iterator_traits<_ForwardIterator>::value_type`

Returns

`generate()` returns no value.

Performs the assignment `*i = gen ()` for each `i` in the range `[first,last)`.

Definition at line 4802 of file `stl_algo.h`.

2.28.1.8 `template<typename _OutputIterator, typename _Size, typename
_Generator> _OutputIterator std::generate_n (_OutputIterator
__first, _Size __n, _Generator __gen)`

Assign the result of a function object to each value in a sequence.

Parameters

first A forward iterator.

n The length of the sequence.

gen A function object taking no arguments and returning `std::iterator_traits<_ForwardIterator>::value_type`

Returns

The end of the sequence, `first+n`

Performs the assignment `*i = gen()` for each `i` in the range `[first, first+n)`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 4833 of file `stl_algo.h`.

2.28.1.9 `template<typename _InputIterator, typename _Predicate> bool
std::is_partitioned (_InputIterator __first, _InputIterator __last,
_Predicate __pred) [inline]`

Checks whether the sequence is partitioned.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

True if the range `[first, last)` is partitioned by `pred`, i.e. if all elements that satisfy `pred` appear before those that do not.

Definition at line 801 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

2.28.1.10 `template<typename _ForwardIterator1, typename
_ForwardIterator2> void std::iter_swap (_ForwardIterator1 __a,
_ForwardIterator2 __b) [inline]`

Swaps the contents of two iterators.

Parameters

a An iterator.

b Another iterator.

Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 116 of file `stl_algobase.h`.

Referenced by `std::__merge_without_buffer()`, `std::__move_median_first()`, `std::__partition()`, `std::__reverse()`, `std::__rotate()`, `std::__unguarded_partition()`, `std::next_permutation()`, `std::prev_permutation()`, `std::random_shuffle()`, `std::shuffle()`, and `std::swap_ranges()`.

2.28.1.11 `template<typename _II, typename _OI> _OI std::move (_II
__first, _II __last, _OI __result) [inline]`

Moves the range `[first,last)` into `result`.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 475 of file `stl_algobase.h`.

2.28.1.12 `template<typename _Tp> std::remove_reference<_Tp>::type&&
std::move (_Tp && __t) [inline]`

Move a value.

Parameters

__t A thing of arbitrary type.

Returns

Same, moved.

Definition at line 86 of file `move.h`.

2.28.1.13 `template<typename _BI1 , typename _BI2 > _BI2
std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
[inline]`

Moves the range [first,last) into result.

Parameters

first A bidirectional iterator.
last A bidirectional iterator.
result A bidirectional iterator.

Returns

result - (first - last)

The function has the same effect as move, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to memmove whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use move instead. Note that the start of the output range may overlap [first,last).

Definition at line 647 of file stl_algobase.h.

2.28.1.14 `template<typename _ForwardIterator , typename _Predicate
> _ForwardIterator std::partition (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred) [inline]`

Move elements for which a predicate is true to the beginning of a sequence.

Parameters

first A forward iterator.
last A forward iterator.
pred A predicate functor.

Returns

An iterator middle such that pred(i) is true for each iterator i in the range [first,middle) and false for each i in the range [middle,last).

pred must not modify its operand. partition() does not preserve the relative ordering of elements in each group, use stable_partition() if this is needed.

Definition at line 5005 of file stl_algo.h.

References std::__iterator_category(), and std::__partition().

2.28.1.15 `template<typename _InputIterator , typename _OutputIterator1
, typename _OutputIterator2 , typename _Predicate >
pair<_OutputIterator1, _OutputIterator2> std::partition_copy (`
`_InputIterator __first, _InputIterator __last, _OutputIterator1`
`__out_true, _OutputIterator2 __out_false, _Predicate __pred)`

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

Parameters

first An input iterator.
last An input iterator.
out_true An output iterator.
out_false An output iterator.
pred A predicate.

Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range [first,last) for which *pred* returns true to the range beginning at *out_true* and each element for which *pred* returns false to *out_false*.

Definition at line 1035 of file `stl_algo.h`.

2.28.1.16 `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first,`
`_FowardIterator __last, _Predicate __pred)`

Find the partition point of a partitioned range.

Parameters

first An iterator.
last Another iterator.
pred A predicate.

Returns

An iterator *mid* such that `all_of(first, mid, pred)` and `none_of(mid, last, pred)` are both true.

Definition at line 819 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.28.1.17 `template<typename _RandomAccessIterator > void
std::random_shuffle (_RandomAccessIterator __first,
_RandomAccessIterator __last) [inline]`

Randomly shuffle the elements of a sequence.

Parameters

first A forward iterator.

last A forward iterator.

Returns

Nothing.

Reorder the elements in the range [first,last) using a random distribution, so that every possible ordering of the sequence is equally likely.

Definition at line 4941 of file stl_algo.h.

References std::iter_swap().

2.28.1.18 `template<typename _RandomAccessIterator , typename
_RandomNumberGenerator > void std::random_shuffle (
_RandomAccessIterator __first, _RandomAccessIterator __last,
_RandomNumberGenerator && __rand)`

Shuffle the elements of a sequence using a random number generator.

Parameters

first A forward iterator.

last A forward iterator.

rand The RNG functor or function.

Returns

Nothing.

Reorders the elements in the range [first,last) using *rand* to provide a random distribution. Calling *rand*(N) for a positive integer N should return a randomly chosen integer from the range [0,N).

Definition at line 4969 of file stl_algo.h.

References std::iter_swap().

2.28.1.19 `template<typename _ForwardIterator, typename _Tp >
_ForwardIterator std::remove (_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __value)`

Remove elements from a sequence.

Parameters

first An input iterator.

last An input iterator.

value The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

All elements equal to *value* are removed from the range [first,last).

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and *last* are still present, but their value is unspecified.

Definition at line 1084 of file `stl_algo.h`.

2.28.1.20 `template<typename _InputIterator, typename _OutputIterator,
typename _Tp > _OutputIterator std::remove_copy (_InputIterator
__first, _InputIterator __last, _OutputIterator __result, const _Tp
& __value)`

Copy a sequence, removing elements of a given value.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

value The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [first,last) not equal to *value* to the range beginning at *result*. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 872 of file `stl_algo.h`.

2.28.1.21 `template<typename _InputIterator , typename _OutputIterator
, typename _Predicate > _OutputIterator std::remove_copy_if (
_InputIterator __first, _InputIterator __last, _OutputIterator
__result, _Predicate __pred)`

Copy a sequence, removing elements for which a predicate is true.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

pred A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` for which `pred` returns false to the range beginning at `result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 910 of file `stl_algo.h`.

2.28.1.22 `template<typename _ForwardIterator , typename _Predicate >
_ForwardIterator std::remove_if (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred)`

Remove elements from a sequence using a predicate.

Parameters

first A forward iterator.

last A forward iterator.

pred A predicate.

Returns

An iterator designating the end of the resulting sequence.

All elements for which `pred` returns true are removed from the range `[first,last)`.

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1127 of file `stl_algo.h`.

2.28.1.23 `template<typename _ForwardIterator, typename _Tp> void
std::replace (_ForwardIterator __first, _ForwardIterator __last,
const _Tp & __old_value, const _Tp & __new_value)`

Replace each occurrence of one value in a sequence with another value.

Parameters

first A forward iterator.

last A forward iterator.

old_value The value to be replaced.

new_value The replacement value.

Returns

`replace()` returns no value.

For each iterator `i` in the range `[first,last)` if `*i == old_value` then the assignment `*i = new_value` is performed.

Definition at line 4738 of file `stl_algo.h`.

2.28.1.24 `template<typename _InputIterator, typename _OutputIterator
, typename _Predicate, typename _Tp> _OutputIterator
std::replace_copy_if (_InputIterator __first, _InputIterator
__last, _OutputIterator __result, _Predicate __pred, const _Tp &
__new_value)`

Copy a sequence, replacing each value for which a predicate returns true with another value.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

pred A predicate.

new_value The replacement value.

Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the range `[first,last)` to the range `[result,result+(last-first))` replacing elements for which `pred` returns true with `new_value`.

Definition at line 3779 of file `stl_algo.h`.

2.28.1.25 `template<typename _ForwardIterator , typename _Predicate ,
typename _Tp > void std::replace_if (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred, const _Tp &
__new_value)`

Replace each value in a sequence for which a predicate returns true with another value.

Parameters

first A forward iterator.

last A forward iterator.

pred A predicate.

new_value The replacement value.

Returns

`replace_if()` returns no value.

For each iterator `i` in the range `[first,last)` if `pred(*i)` is true then the assignment `*i = new_value` is performed.

Definition at line 4770 of file `stl_algo.h`.

2.28.1.26 `template<typename _BidirectionalIterator > void std::reverse (
_BidirectionalIterator __first, _BidirectionalIterator __last)
[inline]`

Reverse a sequence.

Parameters

first A bidirectional iterator.

last A bidirectional iterator.

Returns

`reverse()` returns no value.

Reverses the order of the elements in the range `[first,last)`, so that the first element becomes the last etc. For every i such that $0 \leq i \leq (\text{last} - \text{first})/2$, `reverse()` swaps `*(first+i)` and `*(last-(i+1))`

Definition at line 1435 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__reverse()`.

Referenced by `std::next_permutation()`, and `std::prev_permutation()`.

2.28.1.27 `template<typename _BidirectionalIterator, typename
_OutputIterator > _OutputIterator std::reverse_copy (
_BidirectionalIterator __first, _BidirectionalIterator __last,
_OutputIterator __result)`

Copy a sequence, reversing its elements.

Parameters

first A bidirectional iterator.

last A bidirectional iterator.

result An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range `[first,last)` to the range `[result,result+(last-first))` such that the order of the elements is reversed. For every i such that $0 \leq i \leq (\text{last} - \text{first})$, `reverse_copy()` performs the assignment `*(result+(last-first)-i) = *(first+i)`. The ranges `[first,last)` and `[result,result+(last-first))` must not overlap.

Definition at line 1462 of file `stl_algo.h`.

2.28.1.28 `template<typename _ForwardIterator > void std::rotate
(_ForwardIterator __first, _ForwardIterator __middle,
_ForwardIterator __last) [inline]`

Rotate the elements of a sequence.

Parameters

first A forward iterator.

middle A forward iterator.

last A forward iterator.

Returns

Nothing.

Rotates the elements of the range `[first,last)` by `(middle-first)` positions so that the element at `middle` is moved to `first`, the element at `middle+1` is moved to `+1` and so on for each element in the range `[first,last)`.

This effectively swaps the ranges `[first,middle)` and `[middle,last)`.

Performs `*(first+(n+(last-middle))%(last-first))=*(first+n)` for each `n` in the range `[0,last-first)`.

Definition at line 1666 of file `stl_algo.h`.

References `std::__rotate()`.

Referenced by `std::__inplace_stable_partition()`, `std::__merge_without_buffer()`, `std::__rotate_adaptive()`, and `std::__stable_partition_adaptive()`.

2.28.1.29 `template<typename _ForwardIterator, typename _OutputIterator
> _OutputIterator std::rotate_copy (_ForwardIterator
__first, _ForwardIterator __middle, _ForwardIterator __last,
_OutputIterator __result)`

Copy a sequence, rotating its elements.

Parameters

first A forward iterator.

middle A forward iterator.

last A forward iterator.

result An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[first,last)` to the range beginning at

Returns

, rotating the copied elements by `(middle-first)` positions so that the element at `middle` is moved to `result`, the element at `middle+1` is moved to `+1` and so on for each element in the range `[first,last)`.

Performs `*(result+(n+(last-middle))%(last-first))=*(first+n)` for each `n` in the range `[0,last-first)`.

Definition at line 1700 of file `stl_algo.h`.

2.28.1.30 `template<typename _RandomAccessIterator , typename
_UniformRandomNumberGenerator > void std::shuffle (`
`_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_UniformRandomNumberGenerator && __g)`

Shuffle the elements of a sequence using a uniform random number generator.

Parameters

first A forward iterator.

last A forward iterator.

g A UniformRandomNumberGenerator (26.5.1.3).

Returns

Nothing.

Reorders the elements in the range [first,last) using *g* to provide random numbers.

Definition at line 4128 of file stl_algo.h.

References std::iter_swap().

2.28.1.31 `template<typename _ForwardIterator , typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first,`
`_ForwardIterator __last, _Predicate __pred)`

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

Parameters

first A forward iterator.

last A forward iterator.

pred A predicate functor.

Returns

An iterator *middle* such that *pred*(*i*) is true for each iterator *i* in the range [first,middle) and false for each *i* in the range [middle,last).

Performs the same function as *partition*() with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements *x* and *y* in the range [first,last) such that *pred*(*x*)==*pred*(*y*) will have the same relative ordering after calling *stable_partition*() .

Definition at line 1858 of file stl_algo.h.

References `std::__inplace_stable_partition()`, `std::__stable_partition_adaptive()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::requested_size()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.28.1.32 `template<typename _Tp> void std::swap (_Tp & __a, _Tp & __b) [inline]`

Swaps two values.

Parameters

__a A thing of arbitrary type.
__b Another thing of arbitrary type.

Returns

Nothing.

Definition at line 123 of file `move.h`.

2.28.1.33 `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

Swap the elements of two sequences.

Parameters

first1 A forward iterator.
last1 A forward iterator.
first2 A forward iterator.

Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 157 of file `stl_algobase.h`.

References `std::iter_swap()`.

Referenced by `std::__rotate()`.

2.28.1.34 `template<typename _InputIterator1 , typename _InputIterator2
, typename _OutputIterator , typename _BinaryOperation
> _OutputIterator std::transform (_InputIterator1 __first1,
_InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator
__result, _BinaryOperation __binary_op)`

Perform an operation on corresponding elements of two sequences.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

result An output iterator.

binary_op A binary operator.

Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates $*(result+N)=binary_op(*(first1+N),*(first2+N))$ for each N in the range $[0, last1-first1)$.

`binary_op` must not alter either of its arguments.

Definition at line 4706 of file `stl_algo.h`.

2.28.1.35 `template<typename _InputIterator , typename _OutputIterator ,
typename _UnaryOperation > _OutputIterator std::transform (_InputIterator __first,
_InputIterator __last, _OutputIterator
__result, _UnaryOperation __unary_op)`

Perform an operation on a sequence.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

unary_op A unary operator.

Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates $*(result+N)=unary_op(*(first+N))$ for each N in the range $[0, last-first)$.

`unary_op` must not alter its argument.

Definition at line 4670 of file `stl_algo.h`.

2.28.1.36 `template<typename _ForwardIterator, typename _BinaryPredicate
> _ForwardIterator std::unique (_ForwardIterator __first,
_ForwardIterator __last, _BinaryPredicate __binary_pred)`

Remove consecutive values from a sequence using a predicate.

Parameters

first A forward iterator.

last A forward iterator.

binary_pred A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1207 of file `stl_algo.h`.

2.28.1.37 `template<typename _ForwardIterator > _ForwardIterator
std::unique (_ForwardIterator __first, _ForwardIterator __last)`

Remove consecutive duplicate values from a sequence.

Parameters

first A forward iterator.

last A forward iterator.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is

unchanged. Elements between the end of the resulting sequence and `last` are still present, but their value is unspecified.

Definition at line 1167 of file `stl_algo.h`.

2.28.1.38 `template<typename _InputIterator , typename _OutputIterator
> _OutputIterator std::unique_copy (_InputIterator __first,
_InputIterator __last, _OutputIterator __result) [inline]`

Copy a sequence, removing consecutive duplicate values.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` to the range beginning at `result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 4870 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

2.28.1.39 `template<typename _InputIterator , typename _OutputIterator ,
typename _BinaryPredicate > _OutputIterator std::unique_copy (`
`_InputIterator __first, _InputIterator __last, _OutputIterator`
`__result, _BinaryPredicate __binary_pred) [inline]`

Copy a sequence, removing consecutive values using a predicate.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

binary_pred A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[first,last)` to the range beginning at `result`, except that only the first element is copied from groups of consecutive elements for which `binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require Copy-Constructible and Assignable?

Definition at line 4910 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

2.29 Non-Mutating

Collaboration diagram for Non-Mutating:



Functions

- `template<typename _ForwardIterator >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __`
`first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator`
`__first, _InputIterator __last, _Predicate __pred)`
- `template<typename _II1, typename _II2 >`
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _`
`BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp`
`&__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1`
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _`
`BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1`
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1,`
`_ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1,`
`_ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _`
`Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _`
`Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function`
`__f)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_`
`pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2)`

- `template<typename _InputIterator, typename _Predicate >`
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

2.29.1 Function Documentation

2.29.1.1 `template<typename _ForwardIterator > _ForwardIterator` `std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`

Find two adjacent values in a sequence that are equal.

Parameters

first A forward iterator.

last A forward iterator.

Returns

The first iterator *i* such that *i* and *i+1* are both valid iterators in [*first*,*last*) and such that **i == *(i+1)*, or *last* if no such iterator exists.

Definition at line 4317 of file `stl_algo.h`.

2.29.1.2 `template<typename _ForwardIterator, typename _BinaryPredicate >` `_ForwardIterator std::adjacent_find (_ForwardIterator __first,` `_ForwardIterator __last, _BinaryPredicate __binary_pred)`

Find two adjacent values in a sequence using a predicate.

Parameters

first A forward iterator.

last A forward iterator.

binary_pred A binary predicate.

Returns

The first iterator *i* such that *i* and *i*+1 are both valid iterators in [*first*,*last*) and such that `binary_pred(*i,*i+1)` is true, or *last* if no such iterator exists.

Definition at line 4349 of file `stl_algo.h`.

```
2.29.1.3  template<typename _InputIterator , typename _Predicate > bool
           std::all_of( _InputIterator __first, _InputIterator __last, _Predicate
                      __pred ) [inline]
```

Checks that a predicate is true for all the elements of a sequence.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

True if the check is true, false otherwise.

Returns true if *pred* is true for each element in the range [*first*,*last*), and false otherwise.

Definition at line 728 of file `stl_algo.h`.

References `std::find_if_not()`.

```
2.29.1.4  template<typename _InputIterator , typename _Predicate >
           bool std::any_of( _InputIterator __first, _InputIterator __last,
                          _Predicate __pred ) [inline]
```

Checks that a predicate is false for at least an element of a sequence.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range [first,last) such that *pred* is true, and false otherwise.

Definition at line 762 of file `stl_algo.h`.

References `std::none_of()`.

2.29.1.5 `template<typename _InputIterator , typename _Tp >
iterator_traits<_InputIterator>::difference_type std::count (`
`_InputIterator __first, _InputIterator __last, const _Tp & __value)`

Count the number of copies of a value in a sequence.

Parameters

first An input iterator.

last An input iterator.

value The value to be counted.

Returns

The number of iterators *i* in the range [first,last) for which `*i == value`

Definition at line 4381 of file `stl_algo.h`.

2.29.1.6 `template<typename _InputIterator , typename _Predicate >
iterator_traits<_InputIterator>::difference_type std::count_if (`
`_InputIterator __first, _InputIterator __last, _Predicate __pred)`

Count the elements of a sequence for which a predicate is true.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

The number of iterators *i* in the range [first,last) for which `pred(*i)` is true.

Definition at line 4406 of file `stl_algo.h`.

2.29.1.7 `template<typename _II1, typename _II2 > bool std::equal (_II1
__first1, _II1 __last1, _II2 __first2) [inline]`

Tests a range for element-wise equality.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

Returns

A boolean true or false.

This compares the elements of two ranges using == and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1008 of file stl_algobase.h.

Referenced by std::operator==().

2.29.1.8 `template<typename _IIter1, typename _IIter2, typename
_BinaryPredicate > bool std::equal (_IIter1 __first1, _IIter1 __last1,
_IIter2 __first2, _BinaryPredicate __binary_pred) [inline]`

Tests a range for element-wise equality.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

binary_pred A binary predicate [functor](#).

Returns

A boolean true or false.

This compares the elements of two ranges using the *binary_pred* parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1040 of file stl_algobase.h.

2.29.1.9 `template<typename _InputIterator , typename _Tp > _InputIterator
std::find (_InputIterator __first, _InputIterator __last, const _Tp &
__val) [inline]`

Find the first occurrence of a value in a sequence.

Parameters

first An input iterator.

last An input iterator.

val The value to find.

Returns

The first iterator *i* in the range [*first*,*last*) such that **i == val*, or *last* if no such iterator exists.

Definition at line 4193 of file `stl_algo.h`.

References `std::__find()`, and `std::__iterator_category()`.

2.29.1.10 `template<typename _ForwardIterator1 , typename
_ForwardIterator2 , typename _BinaryPredicate >
_ForwardIterator1 std::find_end (_ForwardIterator1 __first1,
_ForwardIterator1 __last1, _ForwardIterator2 __first2,
_ForwardIterator2 __last2, _BinaryPredicate __comp)
[inline]`

Find last matching subsequence in a sequence using a predicate.

Parameters

first1 Start of range to search.

last1 End of range to search.

first2 Start of sequence to match.

last2 End of sequence to match.

comp The predicate to use.

Returns

The last iterator *i* in the range [*first1*,*last1*-(*last2*-*first2*)) such that `predicate(*(i+N), (first2+N))` is true for each *N* in the range [0,*last2*-*first2*), or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[first,last1)`.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[first1,last1-(last2-first2))`

Definition at line 694 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.29.1.11  template<typename _ForwardIterator1 , typename
              _ForwardIterator2 > _ForwardIterator1 std::find_end (
              _ForwardIterator1 __first1, _ForwardIterator1 __last1,
              _ForwardIterator2 __first2, _ForwardIterator2 __last2 )
              [inline]
```

Find last matching subsequence in a sequence.

Parameters

first1 Start of range to search.

last1 End of range to search.

first2 Start of sequence to match.

last2 End of sequence to match.

Returns

The last iterator `i` in the range `[first1,last1-(last2-first2))` such that `*(i+N) == *(first2+N)` for each `N` in the range `[0,last2-first2)`, or `last1` if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` and returns an iterator to the first element of the sub-sequence, or `last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[first,last1)`.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[first1,last1-(last2-first2))`

Definition at line 647 of file `stl_algo.h`.

References `std::__iterator_category()`.

2.29.1.12 `template<typename _InputIterator , typename _ForwardIterator ,
typename _BinaryPredicate > _InputIterator std::find_first_of (`
`_InputIterator __first1, _InputIterator __last1, _ForwardIterator`
`__first2, _ForwardIterator __last2, _BinaryPredicate __comp)`

Find element from a set in a sequence using a predicate.

Parameters

first1 Start of range to search.
last1 End of range to search.
first2 Start of match candidates.
last2 End of match candidates.
comp Predicate to use.

Returns

The first iterator *i* in the range [*first1*,*last1*) such that `comp(*i, *(i2))` is true and *i2* is an iterator in [*first2*,*last2*), or *last1* if no such iterator exists.

Searches the range [*first1*,*last1*) for an element that is equal to some element in the range [*first2*,*last2*). If found, returns an iterator in the range [*first1*,*last1*), otherwise returns *last1*.

Definition at line 4286 of file `stl_algo.h`.

2.29.1.13 `template<typename _InputIterator , typename _ForwardIterator`
`> _InputIterator std::find_first_of (_InputIterator`
`__first1, _InputIterator __last1, _ForwardIterator __first2,`
`_FowardIterator __last2)`

Find element from a set in a sequence.

Parameters

first1 Start of range to search.
last1 End of range to search.
first2 Start of match candidates.
last2 End of match candidates.

Returns

The first iterator *i* in the range [*first1*,*last1*) such that `*i == *(i2)` such that *i2* is an iterator in [*first2*,*last2*), or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for an element that is equal to some element in the range `[first2,last2)`. If found, returns an iterator in the range `[first1,last1)`, otherwise returns `last1`.

Definition at line 4246 of file `stl_algo.h`.

```
2.29.1.14  template<typename _InputIterator , typename _Predicate >
             _InputIterator std::find_if ( _InputIterator __first, _InputIterator
             __last, _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is true.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

The first iterator `i` in the range `[first,last)` such that `pred(*i)` is true, or `last` if no such iterator exists.

Definition at line 4217 of file `stl_algo.h`.

References `std::__find_if()`, and `std::__iterator_category()`.

```
2.29.1.15  template<typename _InputIterator , typename _Predicate
             > _InputIterator std::find_if_not ( _InputIterator __first,
             _InputIterator __last, _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is false.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

The first iterator `i` in the range `[first,last)` such that `pred(*i)` is false, or `last` if no such iterator exists.

Definition at line 777 of file `stl_algo.h`.

References `std::__find_if_not()`, and `std::__iterator_category()`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

2.29.1.16 `template<typename _InputIterator , typename _Function >
 _Function std::for_each (_InputIterator __first, _InputIterator
 __last, _Function __f)`

Apply a function to every element of a sequence.

Parameters

first An input iterator.

last An input iterator.

f A unary function object.

Returns

\mathbb{f} (std::move(\mathbb{f}) in C++0x).

Applies the function object \mathbb{f} to each element in the range [first,last). \mathbb{f} must not modify the order of the sequence. If \mathbb{f} has a return value it is ignored.

Definition at line 4172 of file stl_algo.h.

2.29.1.17 `template<typename _InputIterator1 , typename _InputIterator2
 , typename _BinaryPredicate > pair<_InputIterator1,
 _InputIterator2> std::mismatch (_InputIterator1 __first1,
 _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate
 __binary_pred)`

Finds the places in ranges which don't match.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

binary_pred A binary predicate [functor](#).

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1183 of file stl_algobase.h.

2.29.1.18 `template<typename _InputIterator1, typename _InputIterator2
> pair<_InputIterator1, _InputIterator2> std::mismatch (`
 `_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`
 `__first2)`

Finds the places in ranges which don't match.

Parameters

first1 An input iterator.

last1 An input iterator.

first2 An input iterator.

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using == and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1145 of file stl_algobase.h.

2.29.1.19 `template<typename _InputIterator, typename _Predicate > bool`
 `std::none_of (_InputIterator __first, _InputIterator __last,`
 `_Predicate __pred) [inline]`

Checks that a predicate is false for all the elements of a sequence.

Parameters

first An input iterator.

last An input iterator.

pred A predicate.

Returns

True if the check is true, false otherwise.

Returns true if *pred* is false for each element in the range [first,last), and false otherwise.

Definition at line 745 of file stl_algo.h.

Referenced by std::any_of(), and std::is_partitioned().

2.29.1.20 `template<typename _ForwardIterator1 , typename
_ForwardIterator2 > _ForwardIterator1 std::search (`
`_ForwardIterator1 __first1, _ForwardIterator1 __last1,`
`_ForwardIterator2 __first2, _ForwardIterator2 __last2)`

Search a sequence for a matching sub-sequence.

Parameters

first1 A forward iterator.

last1 A forward iterator.

first2 A forward iterator.

last2 A forward iterator.

Returns

The first iterator *i* in the range `[first1,last1-(last2-first2))` such that `*(i+N) == *(first2+N)` for each *N* in the range `[0,last2-first2)`, or *last1* if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)` and returns an iterator to the first element of the sub-sequence, or *last1* if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range `[first1,last1)` it must start at a position less than `last1-(last2-first2)` where `last2-first2` is the length of the sub-sequence. This means that the returned iterator *i* will be in the range `[first1,last1-(last2-first2))`

Definition at line 4446 of file `stl_algo.h`.

2.29.1.21 `template<typename _ForwardIterator1 , typename
_ForwardIterator2 , typename _BinaryPredicate >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1,`
`_ForwardIterator1 __last1, _ForwardIterator2 __first2,`
`_ForwardIterator2 __last2, _BinaryPredicate __predicate)`

Search a sequence for a matching sub-sequence using a predicate.

Parameters

first1 A forward iterator.

last1 A forward iterator.

first2 A forward iterator.

last2 A forward iterator.

predicate A binary predicate.

Returns

The first iterator *i* in the range `[first1,last1-(last2-first2))` such that `predicate(*(i+N),*(first2+N))` is true for each *N* in the range `[0,last2-first2)`, or `last1` if no such iterator exists.

Searches the range `[first1,last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[first2,last2)`, using `predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `last1` if no such iterator exists.

See also

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4518 of file `stl_algo.h`.

2.29.1.22 `template<typename _ForwardIterator , typename _Integer ,
typename _Tp > _ForwardIterator std::search_n (_ForwardIterator
__first, _ForwardIterator __last, _Integer __count, const _Tp &
__val)`

Search a sequence for a number of consecutive values.

Parameters

first A forward iterator.

last A forward iterator.

count The number of consecutive values.

val The value to find.

Returns

The first iterator *i* in the range `[first,last-count)` such that `*(i+N) == val` for each *N* in the range `[0,count)`, or `last` if no such iterator exists.

Searches the range `[first,last)` for `count` consecutive elements equal to `val`.

Definition at line 4591 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

2.29.1.23 `template<typename _ForwardIterator , typename _Integer ,
typename _Tp , typename _BinaryPredicate > _ForwardIterator
std::search_n (_ForwardIterator __first, _ForwardIterator
__last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred)`

Search a sequence for a number of consecutive values using a predicate.

Parameters

first A forward iterator.

last A forward iterator.

count The number of consecutive values.

val The value to find.

binary_pred A binary predicate.

Returns

The first iterator *i* in the range `[first,last-count)` such that `binary_pred(*(i+N),val)` is true for each *N* in the range `[0,count)`, or *last* if no such iterator exists.

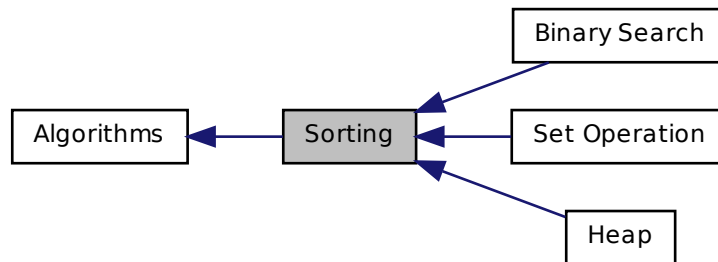
Searches the range `[first,last)` for *count* consecutive elements for which the predicate returns true.

Definition at line 4628 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__search_n()`.

2.30 Sorting

Collaboration diagram for Sorting:



Modules

- [Set Operation](#)
- [Binary Search](#)
- [Heap](#)

Functions

- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`

- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _II1, typename _II2 >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`
`__last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2`
`__last2, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _`
`_Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`
`&__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`
`&__b)`

- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_-`
`ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_-`
`ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator`
`__nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator`
`__nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __-`
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`
`RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __-`
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`
`RandomAccessIterator __result_last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`

2.30.1 Function Documentation

2.30.1.1 `template<typename _BidirectionalIterator > void std::inplace_merge` `(_BidirectionalIterator __first, _BidirectionalIterator __middle,` `_BidirectionalIterator __last)`

Merges two sorted ranges in place.

Parameters

first An iterator.

middle Another iterator.

last Another iterator.

Returns

Nothing.

Merges two sorted and consecutive ranges, `[first,middle)` and `[middle,last)`, and puts the result in `[first,last)`. The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes $(last-first)-1$ comparisons. Otherwise an $N\log N$ algorithm is used, where N is `distance(first,last)`.

Definition at line 3054 of file `stl_algo.h`.

References `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::_Temporary_buffer<_ForwardIterator, _Tp >::begin()`, `std::distance()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::size()`.

2.30.1.2 `template<typename _BidirectionalIterator, typename _Compare` `> void std::inplace_merge (_BidirectionalIterator __first,` `_BidirectionalIterator __middle, _BidirectionalIterator __last,` `_Compare __comp)`

Merges two sorted ranges in place.

Parameters

first An iterator.
middle Another iterator.
last Another iterator.
comp A functor to use for comparisons.

Returns

Nothing.

Merges two sorted and consecutive ranges, [first,middle) and [middle,last), and puts the result in [first,last). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (last-first)-1 comparisons. Otherwise an NlogN algorithm is used, where N is distance(first,last).

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 3109 of file stl_algo.h.

References std::__merge_adaptive(), std::__merge_without_buffer(), std::_Temporary_buffer<_ForwardIterator, _Tp >::begin(), std::distance(), and std::_Temporary_buffer<_ForwardIterator, _Tp >::size().

2.30.1.3 `template<typename _ForwardIterator> bool std::is_sorted (` `_ForwardIterator __first, _ForwardIterator __last) [inline]`

Determines whether the elements of a sequence are sorted.

Parameters

first An iterator.
last Another iterator.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3809 of file stl_algo.h.

References std::is_sorted_until().

2.30.1.4 `template<typename _ForwardIterator, typename _Compare> bool
std::is_sorted (_ForwardIterator __first, _ForwardIterator __last,
_Compare __comp) [inline]`

Determines whether the elements of a sequence are sorted according to a comparison functor.

Parameters

first An iterator.

last Another iterator.

comp A comparison functor.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3823 of file `stl_algo.h`.

References `std::is_sorted_until()`.

2.30.1.5 `template<typename _ForwardIterator> _ForwardIterator
std::is_sorted_until (_ForwardIterator __first, _ForwardIterator
__last)`

Determines the end of a sorted sequence.

Parameters

first An iterator.

last Another iterator.

Returns

An iterator pointing to the last iterator *i* in `[first, last)` for which the range `[first, i)` is sorted.

Definition at line 3837 of file `stl_algo.h`.

2.30.1.6 `template<typename _ForwardIterator, typename _Compare>
_ForwardIterator std::is_sorted_until (_ForwardIterator __first,
_ForwardIterator __last, _Compare __comp)`

Determines the end of a sorted sequence using comparison functor.

Parameters

first An iterator.
last Another iterator.
comp A comparison functor.

Returns

An iterator pointing to the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is sorted.

Definition at line 3866 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

2.30.1.7 `template<typename _II1 , typename _II2 > bool
 std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2
 __first2, _II2 __last2) [inline]`

Performs **dictionary** comparison on ranges.

Parameters

first1 An input iterator.
last1 An input iterator.
first2 An input iterator.
last2 An input iterator.

Returns

A boolean true or false.

*Returns true if the sequence of elements defined by the range [*first1*,*last1*) is lexicographically less than the sequence of elements defined by the range [*first2*,*last2*). Returns false otherwise.* (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1071 of file `stl_algobase.h`.

2.30.1.8 `template<typename _II1 , typename _II2 , typename _Compare >
 bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2
 __first2, _II2 __last2, _Compare __comp)`

Performs **dictionary** comparison on ranges.

Parameters

first1 An input iterator.
last1 An input iterator.
first2 An input iterator.
last2 An input iterator.
comp A [comparison functor](#).

Returns

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1105 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

2.30.1.9 `template<typename _Tp > const _Tp & std::max (const _Tp & __a,
const _Tp & __b) [inline]`

This does what you think it does.

Parameters

a A thing of arbitrary type.
b Another thing of arbitrary type.

Returns

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 208 of file `stl_algobase.h`.

Referenced by `std::_Deque_base< _Tp, _Alloc >::_M_initialize_map()`, and `std::deque< _Tp, _Alloc >::_M_reallocate_map()`.

2.30.1.10 `template<typename _Tp , typename _Compare > const _Tp &
std::max (const _Tp & __a, const _Tp & __b, _Compare __comp
) [inline]`

This does what you think it does.

Parameters

- a* A thing of arbitrary type.
b Another thing of arbitrary type.
comp A [comparison functor](#).

Returns

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 252 of file `stl_algobase.h`.

2.30.1.11 `template<typename _ForwardIterator > _ForwardIterator
std::max_element (_ForwardIterator __first, _ForwardIterator
__last)`

Return the maximum element in a range.

Parameters

- first* Start of range.
last End of range.

Returns

Iterator referencing the first instance of the largest value.

Definition at line 6028 of file `stl_algo.h`.

2.30.1.12 `template<typename _ForwardIterator , typename _Compare >
_ForwardIterator std::max_element (_ForwardIterator __first,
_ForwardIterator __last, _Compare __comp)`

Return the maximum element in a range using comparison functor.

Parameters

- first* Start of range.
last End of range.
comp Comparison functor.

Returns

Iterator referencing the first instance of the largest value according to *comp*.

Definition at line 6056 of file stl_algo.h.

Referenced by std::valarray< _Tp >::max().

2.30.1.13 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

Merges two sorted ranges.

Parameters

first1 An iterator.

first2 Another iterator.

last1 Another iterator.

last2 Another iterator.

result An iterator pointing to the end of the merged range.

Returns

An iterator pointing to the first element *not less than val*.

Merges the ranges [first1,last1) and [first2,last2) into the sorted range [result, result + (last1-first1) + (last2-first2)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 5271 of file stl_algo.h.

2.30.1.14 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare > _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

Merges two sorted ranges.

Parameters

first1 An iterator.

first2 Another iterator.

last1 Another iterator.

last2 Another iterator.

result An iterator pointing to the end of the merged range.

comp A functor to use for comparisons.

Returns

An iterator pointing to the first element "not less than" *val*.

Merges the ranges [first1,last1) and [first2,last2) into the sorted range [result, result + (last1-first1) + (last2-first2)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 5334 of file stl_algo.h.

2.30.1.15 `template<typename _Tp, typename _Compare> const _Tp &
std::min (const _Tp & __a, const _Tp & __b, _Compare __comp)
[inline]`

This does what you think it does.

Parameters

a A thing of arbitrary type.

b Another thing of arbitrary type.

comp A [comparison functor](#).

Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 231 of file stl_algobase.h.

2.30.1.16 `template<typename _Tp> const _Tp & std::min (const _Tp & __a,
const _Tp & __b) [inline]`

This does what you think it does.

Parameters

- a* A thing of arbitrary type.
- b* Another thing of arbitrary type.

Returns

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 185 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_profile::__report()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::basic_string<char>::compare()`, `std::generate_canonical()`, `__gnu_cxx::random_sample_n()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`, `std::basic_string<_CharT, _Traits, _Alloc>::rfind()`, `std::basic_filebuf<_CharT, _Traits>::underflow()`, `std::basic_streambuf<_CharT, _Traits>::xsgetn()`, and `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

2.30.1.17 `template<typename _ForwardIterator> _ForwardIterator
std::min_element (_ForwardIterator __first, _ForwardIterator
__last)`

Return the minimum element in a range.

Parameters

- first* Start of range.
- last* End of range.

Returns

Iterator referencing the first instance of the smallest value.

Definition at line 5972 of file `stl_algo.h`.

2.30.1.18 `template<typename _ForwardIterator, typename _Compare>
_ForwardIterator std::min_element (_ForwardIterator __first,
_ForwardIterator __last, _Compare __comp)`

Return the minimum element in a range using comparison functor.

Parameters

first Start of range.
last End of range.
comp Comparison functor.

Returns

Iterator referencing the first instance of the smallest value according to comp.

Definition at line 6000 of file stl_algo.h.

Referenced by std::valarray<_Tp>::min().

2.30.1.19 `template<typename _Tp, typename _Compare> pair< const _Tp
&, const _Tp & > std::minmax (const _Tp & __a, const _Tp &
__b, _Compare __comp) [inline]`

Determines min and max at once as an ordered pair.

Parameters

a A thing of arbitrary type.
b Another thing of arbitrary type.
comp A [comparison functor](#).

Returns

A pair(b, a) if b is smaller than a, pair(a, b) otherwise.

Definition at line 3914 of file stl_algo.h.

2.30.1.20 `template<typename _Tp> pair< const _Tp &, const _Tp & >
std::minmax (const _Tp & __a, const _Tp & __b) [inline]`

Determines min and max at once as an ordered pair.

Parameters

a A thing of arbitrary type.
b Another thing of arbitrary type.

Returns

A pair(b, a) if b is smaller than a, pair(a, b) otherwise.

Definition at line 3895 of file stl_algo.h.

2.30.1.21 `template<typename _ForwardIterator > pair<_ForwardIterator, _ForwardIterator> std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

first Start of range.

last End of range.

Returns

make_pair(m, M), where m is the first iterator i in [first, last) such that no other element in the range is smaller, and where M is the last iterator i in [first, last) such that no other element in the range is larger.

Definition at line 3933 of file stl_algo.h.

References std::make_pair().

2.30.1.22 `template<typename _ForwardIterator , typename _Compare > pair<_ForwardIterator, _ForwardIterator> std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

first Start of range.

last End of range.

comp Comparison functor.

Returns

make_pair(m, M), where m is the first iterator i in [first, last) such that no other element in the range is smaller, and where M is the last iterator i in [first, last) such that no other element in the range is larger.

Definition at line 4009 of file stl_algo.h.

References std::make_pair().

2.30.1.23 `template<typename _BidirectionalIterator > bool
std::next_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last)`

Permute range into the next *dictionary* ordering.

Parameters

first Start of range.

last End of range.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3511 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

2.30.1.24 `template<typename _BidirectionalIterator , typename _Compare
> bool std::next_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last, _Compare __comp)`

Permute range into the next *dictionary* ordering using comparison functor.

Parameters

first Start of range.

last End of range.

comp A comparison functor.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range `[first,last)` as a set of *dictionary* sorted sequences ordered by *comp*. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3568 of file `stl_algo.h`.

References `std::iter_swap()`, and `std::reverse()`.

2.30.1.25 `template<typename _RandomAccessIterator > void std::nth_element
(_RandomAccessIterator __first, _RandomAccessIterator __nth,
_RandomAccessIterator __last) [inline]`

Sort a sequence just enough to find a particular position.

Parameters

first An iterator.

nth Another iterator.

last Another iterator.

Returns

Nothing.

Rearranges the elements in the range [first,last) so that *nth is the same element that would have been in that position had the whole sequence been sorted. whole sequence been sorted. The elements either side of *nth are not completely sorted, but for any iterator in the range [first,nth) and any iterator in the range [nth,last) it holds that *j<*i is false.

Definition at line 5116 of file stl_algo.h.

References std::__lg().

2.30.1.26 `template<typename _RandomAccessIterator , typename _Compare
> void std::nth_element (_RandomAccessIterator __first,
_RandomAccessIterator __nth, _RandomAccessIterator __last,
_Compare __comp) [inline]`

Sort a sequence just enough to find a particular position using a predicate for comparison.

Parameters

first An iterator.

nth Another iterator.

last Another iterator.

comp A comparison functor.

Returns

Nothing.

Rearranges the elements in the range `[first,last)` so that `*nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*nth` are not completely sorted, but for any iterator in the range `[first,nth)` and any iterator in the range `[nth,last)` it holds that `comp(*j,*i)` is false.

Definition at line 5155 of file `stl_algo.h`.

References `std::__lg()`.

2.30.1.27 `template<typename _RandomAccessIterator > void std::partial_sort
(_RandomAccessIterator __first, _RandomAccessIterator
__middle, _RandomAccessIterator __last) [inline]`

Sort the smallest elements of a sequence.

Parameters

first An iterator.

middle Another iterator.

last Another iterator.

Returns

Nothing.

Sorts the smallest (middle-first) elements in the range `[first,last)` and moves them to the range `[first,middle)`. The order of the remaining elements in the range `[middle,last)` is undefined. After the sort if `i` and `j` are iterators in the range `[first,middle)` such that `i` precedes `j` and `i` is an iterator in the range `[middle,last)` then `*j < *i` and `*k < *i` are both false.

Definition at line 5039 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

2.30.1.28 `template<typename _RandomAccessIterator, typename _Compare
> void std::partial_sort (_RandomAccessIterator __first,
_RandomAccessIterator __middle, _RandomAccessIterator __last,
_Compare __comp) [inline]`

Sort the smallest elements of a sequence using a predicate for comparison.

Parameters

first An iterator.

middle Another iterator.

last Another iterator.

comp A comparison functor.

Returns

Nothing.

Sorts the smallest (middle-first) elements in the range `[first,last)` and moves them to the range `[first,middle)`. The order of the remaining elements in the range `[middle,last)` is undefined. After the sort if `i` and `j` are iterators in the range `[first,middle)` such that `j` precedes `i` and `i` is an iterator in the range `[middle,last)` then `*comp(j,*i)` and `comp(*j,*i)` are both false.

Definition at line 5078 of file `stl_algo.h`.

References `std::__heap_select()`, and `std::sort_heap()`.

2.30.1.29 `template<typename _InputIterator , typename
_RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::partial_sort_copy (_InputIterator
__first, _InputIterator __last, _RandomAccessIterator __result_first,
_RandomAccessIterator __result_last, _Compare __comp)`

Copy the smallest elements of a sequence using a predicate for comparison.

Parameters

first An input iterator.

last Another input iterator.

result_first A random-access iterator.

result_last Another random-access iterator.

comp A comparison functor.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest `N` values from the range `[first,last)` to the range beginning at `result_first`, where the number of elements to be copied, `N`, is the smaller of `(last-first)` and `(result_last-result_first)`. After the sort if `i` and `j` are iterators in the range `[result_first,result_first+N)` such that `j` precedes `i` then `comp(*j,*i)` is false. The value returned is `result_first+N`.

Definition at line 2006 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

2.30.1.30 `template<typename _InputIterator , typename
_RandomAccessIterator > _RandomAccessIterator
std::partial_sort_copy (_InputIterator __first,
_InputIterator __last, _RandomAccessIterator __result_first,
_RandomAccessIterator __result_last)`

Copy the smallest elements of a sequence.

Parameters

first An iterator.

last Another iterator.

result_first A random-access iterator.

result_last Another random-access iterator.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range [first,last) to the range beginning at *result_first*, where the number of elements to be copied, N, is the smaller of (last-first) and (result_last-result_first). After the sort if *i* and *j* are iterators in the range [result_first,result_first+N) such that *i* precedes *j* then **j < *i* is false. The value returned is *result_first*+N.

Definition at line 1940 of file `stl_algo.h`.

References `std::make_heap()`, and `std::sort_heap()`.

2.30.1.31 `template<typename _BidirectionalIterator , typename _Compare
> bool std::prev_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last, _Compare __comp)`

Permute range into the previous *dictionary* ordering using comparison functor.

Parameters

first Start of range.

last End of range.

comp A comparison functor.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range [first,last) as a set of *dictionary* sorted sequences ordered by *comp*. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3681 of file stl_algo.h.

References std::iter_swap(), and std::reverse().

2.30.1.32 `template<typename _BidirectionalIterator > bool
std::prev_permutation (_BidirectionalIterator __first,
_BidirectionalIterator __last)`

Permute range into the previous *dictionary* ordering.

Parameters

first Start of range.

last End of range.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3624 of file stl_algo.h.

References std::iter_swap(), and std::reverse().

2.30.1.33 `template<typename _RandomAccessIterator , typename
_Compare > void std::sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp) [inline]`

Sort the elements of a sequence using a predicate for comparison.

Parameters

first An iterator.

last Another iterator.

comp A comparison functor.

Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `comp(*(i+1),*i)` is false for every iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5229 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

```
2.30.1.34  template<typename _RandomAccessIterator > void std::sort (  
            _RandomAccessIterator __first, _RandomAccessIterator __last )  
            [inline]
```

Sort the elements of a sequence.

Parameters

first An iterator.

last Another iterator.

Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `*(i+1)<*i` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5193 of file `stl_algo.h`.

References `std::__final_insertion_sort()`, `std::__introsort_loop()`, and `std::__lg()`.

```
2.30.1.35  template<typename _RandomAccessIterator , typename _Compare  
            > void std::stable_sort ( _RandomAccessIterator __first,  
            _RandomAccessIterator __last, _Compare __comp ) [inline]
```

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

Parameters

first An iterator.

last Another iterator.

comp A comparison functor.

Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `comp(*(i+1),*i)` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `comp(x,y)` is false and `comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5435 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.30.1.36 `template<typename _RandomAccessIterator> void std::stable_sort
(_RandomAccessIterator __first, _RandomAccessIterator __last)
[inline]`

Sort the elements of a sequence, preserving the relative order of equivalent elements.

Parameters

first An iterator.

last Another iterator.

Returns

Nothing.

Sorts the elements in the range `[first,last)` in ascending order, such that `*(i+1)<*i` is false for each iterator `i` in the range `[first,last-1)`.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[first,last)` such that `x<y` is false and `y<x` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5393 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`, `std::_Temporary_buffer<_ForwardIterator, _Tp>::begin()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp>::size()`.

2.31 Set Operation

Collaboration diagram for Set Operation:



Functions

- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`
`_Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1`
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`
`result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1`
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`
`result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`
`OutputIterator __result)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`
`OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`

2.31.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

2.31.2 Function Documentation

2.31.2.1 `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2)`

Determines whether all elements of a sequence exists in a range.

Parameters

first1 Start of search range.
last1 End of search range.
first2 Start of sequence
last2 End of sequence.

Returns

True if each element in [first2,last2) is contained in order within [first1,last1). False otherwise.

This operation expects both [first1,last1) and [first2,last2) to be sorted. Searches for the presence of each element in [first2,last2) within [first1,last1). The iterators over each range only move forward, so this is a linear algorithm. If an element in [first2,last2) is not found before the search iterator reaches *last2*, false is returned.

Definition at line 3407 of file `stl_algo.h`.

2.31.2.2 `template<typename _InputIterator1 , typename _InputIterator2 ,
 typename _Compare > bool std::includes (_InputIterator1 __first1,
 _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
 __last2, _Compare __comp)`

Determines whether all elements of a sequence exists in a range using comparison.

Parameters

first1 Start of search range.
last1 End of search range.
first2 Start of sequence
last2 End of sequence.
comp Comparison function to use.

Returns

True if each element in [first2,last2) is contained in order within [first1,last1) according to comp. False otherwise.

This operation expects both [first1,last1) and [first2,last2) to be sorted. Searches for the presence of each element in [first2,last2) within [first1,last1), using comp to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [first2,last2) is not found before the search iterator reaches *last2*, false is returned.

Definition at line 3457 of file stl_algo.h.

2.31.2.3 `template<typename _InputIterator1 , typename _InputIterator2 ,
 typename _OutputIterator , typename _Compare > _OutputIterator
 std::set_difference (_InputIterator1 __first1, _InputIterator1
 __last1, _InputIterator2 __first2, _InputIterator2 __last2,
 _OutputIterator __result, _Compare __comp)`

Return the difference of two sorted ranges using comparison functor.

Parameters

first1 Start of first range.
last1 End of first range.
first2 Start of second range.
last2 End of second range.
comp The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to *comp*, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to *comp*, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5792 of file `stl_algo.h`.

2.31.2.4 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`

Return the difference of two sorted ranges.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5731 of file `stl_algo.h`.

2.31.2.5 `template<typename _InputIterator1 , typename _InputIterator2 ,
typename _OutputIterator > _OutputIterator std::set_intersection (
_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
__first2, _InputIterator2 __last2, _OutputIterator __result)`

Return the intersection of two sorted ranges.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5616 of file `stl_algo.h`.

2.31.2.6 `template<typename _InputIterator1 , typename _InputIterator2 ,
typename _OutputIterator , typename _Compare > _OutputIterator
std::set_intersection (_InputIterator1 __first1, _InputIterator1
__last1, _InputIterator2 __first2, _InputIterator2 __last2,
_OutputIterator __result, _Compare __comp)`

Return the intersection of two sorted ranges using comparison functor.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

comp The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that iterator advances. If an element is contained in both ranges according to *comp*, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5673 of file `stl_algo.h`.

2.31.2.7 `template<typename _InputIterator1 , typename
_InputIterator2 , typename _OutputIterator > _OutputIterator
std::set_symmetric_difference (_InputIterator1 __first1,
_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
__last2, _OutputIterator __result)`

Return the symmetric difference of two sorted ranges.

Parameters

first1 Start of first range.
last1 End of first range.
first2 Start of second range.
last2 End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5850 of file `stl_algo.h`.

2.31.2.8 `template<typename _InputIterator1 , typename _InputIterator2 ,
typename _OutputIterator , typename _Compare > _OutputIterator
std::set_symmetric_difference (_InputIterator1 __first1,
_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
__last2, _OutputIterator __result, _Compare __comp)`

Return the symmetric difference of two sorted ranges using comparison functor.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

comp The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that element is copied and the iterator advances. If an element is contained in both ranges according to *comp*, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5916 of file `stl_algo.h`.

2.31.2.9 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::set_union (`
 `_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`
 `__first2, _InputIterator2 __last2, _OutputIterator __result)`

Return the union of two sorted ranges.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5482 of file `stl_algo.h`.

2.31.2.10 `template<typename _InputIterator1 , typename _InputIterator2 ,
 typename _OutputIterator , typename _Compare > _OutputIterator
 std::set_union (_InputIterator1 __first1, _InputIterator1 __last1,
 _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator
 __result, _Compare __comp)`

Return the union of two sorted ranges using a comparison functor.

Parameters

first1 Start of first range.

last1 End of first range.

first2 Start of second range.

last2 End of second range.

comp The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to *comp*, that element is copied and the iterator advanced. If an equivalent element according to *comp* is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5549 of file `stl_algo.h`.

2.32 Binary Search

Collaboration diagram for Binary Search:



Functions

- `template<typename _ForwardIterator, typename _Tp >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator`
`__first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator`
`__first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val)`

2.32.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by equal_range is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions

the range.

2.32.2 Function Documentation

2.32.2.1 `template<typename _ForwardIterator, typename _Tp> bool
std::binary_search (_ForwardIterator __first, _ForwardIterator
__last, const _Tp & __val)`

Determines whether an element exists in a range.

Parameters

first An iterator.

last Another iterator.

val The search term.

Returns

True if *val* (or its equivalent) is in [*first*,*last*].

Note that this does not actually return an iterator to *val*. For that, use `std::find` or a container's specialized find member functions.

Definition at line 2661 of file `stl_algo.h`.

References `std::lower_bound()`.

2.32.2.2 `template<typename _ForwardIterator, typename _Tp, typename
_Compare> bool std::binary_search (_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __val, _Compare __comp)`

Determines whether an element exists in a range.

Parameters

first An iterator.

last Another iterator.

val The search term.

comp A functor to use for comparisons.

Returns

True if *val* (or its equivalent) is in [*first*,*last*].

Note that this does not actually return an iterator to *val*. For that, use `std::find` or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2694 of file `stl_algo.h`.

References `std::lower_bound()`.

2.32.2.3 `template<typename _ForwardIterator, typename _Tp, typename
_Compare > pair<_ForwardIterator, _ForwardIterator>
std::equal_range (_ForwardIterator __first, _ForwardIterator
__last, const _Tp & __val, _Compare __comp)`

Finds the largest subrange in which *val* could be inserted at any place in it without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

comp A functor to use for comparisons.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(first, last, val, comp),
               upper_bound(first, last, val, comp))
```

but does not actually call those functions.

Definition at line 2601 of file `stl_algo.h`.

References `std::advance()`, `std::distance()`, `std::lower_bound()`, and `std::upper_bound()`.

2.32.2.4 `template<typename _ForwardIterator, typename _Tp >
pair<_ForwardIterator, _ForwardIterator> std::equal_range (
_ForwardIterator __first, _ForwardIterator __last, const _Tp &
__val)`

Finds the largest subrange in which *val* could be inserted at any place in it without changing the ordering.

Parameters

first An iterator.
last Another iterator.
val The search term.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(first, last, val),
               upper_bound(first, last, val))
```

but does not actually call those functions.

Definition at line 2539 of file stl_algo.h.

References std::advance(), std::distance(), std::lower_bound(), and std::upper_bound().

2.32.2.5 `template<typename _ForwardIterator, typename _Tp, typename
 _Compare > _ForwardIterator std::lower_bound (_ForwardIterator
 __first, _ForwardIterator __last, const _Tp & __val, _Compare
 __comp)`

Finds the first position in which *val* could be inserted without changing the ordering.

Parameters

first An iterator.
last Another iterator.
val The search term.
comp A functor to use for comparisons.

Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2390 of file stl_algo.h.

References std::advance(), and std::distance().

2.32.2.6 `template<typename _ForwardIterator, typename _Tp >
_ForwardIterator std::lower_bound (_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __val)`

Finds the first position in which *val* could be inserted without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

Definition at line 934 of file `stl_algobase.h`.

References `std::advance()`, and `std::distance()`.

Referenced by `std::__merge_adaptive()`, `std::__merge_without_buffer()`, `std::binary_search()`, and `std::equal_range()`.

2.32.2.7 `template<typename _ForwardIterator, typename _Tp, typename
_Compare > _ForwardIterator std::upper_bound (_ForwardIterator
__first, _ForwardIterator __last, const _Tp & __val, _Compare
__comp)`

Finds the last position in which *val* could be inserted without changing the ordering.

Parameters

first An iterator.

last Another iterator.

val The search term.

comp A functor to use for comparisons.

Returns

An iterator pointing to the first element greater than *val*, or `end()` if no elements are greater than *val*.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2486 of file stl_algo.h.

References std::advance(), and std::distance().

Referenced by std::__merge_adaptive(), std::__merge_without_buffer(), and std::equal_range().

2.32.2.8 `template<typename _ForwardIterator, typename _Tp >
_ForwardIterator std::upper_bound (_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __val)`

Finds the last position in which *val* could be inserted without changing the ordering.

Parameters

first An iterator.
last Another iterator.
val The search term.

Returns

An iterator pointing to the first element greater than *val*, or [end\(\)](#) if no elements are greater than *val*.

Definition at line 2437 of file stl_algo.h.

References std::advance(), and std::distance().

2.33 Allocators

Collaboration diagram for Allocators:



Classes

- class [__gnu_cxx::__mt_alloc<_Tp, _Poolp >](#)

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

- class `__gnu_cxx::__pool_alloc< _Tp >`
Allocator using a memory pool with a single lock.
- class `__gnu_cxx::__ExtPtr_allocator< _Tp >`
*An example allocator which uses a non-standard pointer type.
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).*
- class `__gnu_cxx::array_allocator< _Tp, _Array >`
An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.
- class `__gnu_cxx::bitmap_allocator< _Tp >`
Bitmap Allocator, primary template.
- class `__gnu_cxx::debug_allocator< _Alloc >`
*A meta-allocator with debugging bits, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls operator new
 - all deallocation calls operator delete.
- class `__gnu_cxx::malloc_allocator< _Tp >`
*An allocator that uses malloc.
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls malloc
 - all deallocation calls free.
- class `__gnu_cxx::new_allocator< _Tp >`
*An allocator that uses global new, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls operator new
 - all deallocation calls operator delete.
- class `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`
*Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templated code.
Note: Deallocate not allowed to throw.*

- class `std::allocator<_Tp>`

The standard allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

2.33.1 Detailed Description

Classes encapsulating memory operations.

2.34 Atomics

Classes

- struct `std::atomic<_Tp>`

atomic /// 29.4.3, Generic atomic type, primary class template.

- struct `std::atomic<_Tp*>`

Partial specialization for pointer types.

- struct `std::atomic<bool>`

Explicit specialization for bool.

- struct `std::atomic<char>`

Explicit specialization for char.

- struct `std::atomic<char16_t>`

Explicit specialization for char16_t.

- struct `std::atomic<char32_t>`

Explicit specialization for char32_t.

- struct `std::atomic<int>`

Explicit specialization for int.

- struct `std::atomic<long>`

Explicit specialization for long.

- struct `std::atomic<long long>`

Explicit specialization for long long.

- struct `std::atomic<short>`

Explicit specialization for short.

- struct `std::atomic< signed char >`

Explicit specialization for signed char.

- struct `std::atomic< unsigned char >`

Explicit specialization for unsigned char.

- struct `std::atomic< unsigned int >`

Explicit specialization for unsigned int.

- struct `std::atomic< unsigned long >`

Explicit specialization for unsigned long.

- struct `std::atomic< unsigned long long >`

Explicit specialization for unsigned long long.

- struct `std::atomic< unsigned short >`

Explicit specialization for unsigned short.

- struct `std::atomic< void * >`

Explicit specialization for void.*

- struct `std::atomic< wchar_t >`

Explicit specialization for wchar_t.

Defines

- #define `_ATOMIC_CMPEXCHNG__(__a, __e, __m, __x)`
- #define `_ATOMIC_LOAD__(__a, __x)`
- #define `_ATOMIC_MODIFY__(__a, __o, __m, __x)`
- #define `_ATOMIC_STORE__(__a, __m, __x)`
- #define `_GLIBCXX_ATOMIC_NAMESPACE`
- #define `_GLIBCXX_ATOMIC_PROPERTY`
- #define `ATOMIC_ADDRESS_LOCK_FREE`
- #define `ATOMIC_FLAG_INIT`
- #define `ATOMIC_INTEGRAL_LOCK_FREE`

Typedefs

- typedef struct std::__atomic_flag_base **std::__atomic_flag_base**
- typedef __atomic_base< char > [atomic_char](#)
- typedef __atomic_base< char16_t > [atomic_char16_t](#)
- typedef __atomic_base< char32_t > [atomic_char32_t](#)
- typedef __atomic_base< int > [atomic_int](#)
- typedef [atomic_short](#) **std::atomic_int_fast16_t**
- typedef [atomic_int](#) **std::atomic_int_fast32_t**
- typedef [atomic_llong](#) **std::atomic_int_fast64_t**
- typedef [atomic_schar](#) **std::atomic_int_fast8_t**
- typedef [atomic_short](#) **std::atomic_int_least16_t**
- typedef [atomic_int](#) **std::atomic_int_least32_t**
- typedef [atomic_llong](#) **std::atomic_int_least64_t**
- typedef [atomic_schar](#) **std::atomic_int_least8_t**
- typedef [atomic_llong](#) **std::atomic_intmax_t**
- typedef [atomic_long](#) **std::atomic_intptr_t**
- typedef __atomic_base< long long > [atomic_llong](#)
- typedef __atomic_base< long > [atomic_long](#)
- typedef [atomic_long](#) **std::atomic_ptrdiff_t**
- typedef __atomic_base< signed char > [atomic_schar](#)
- typedef __atomic_base< short > [atomic_short](#)
- typedef [atomic_ulong](#) **std::atomic_size_t**
- typedef [atomic_long](#) **std::atomic_ssize_t**
- typedef __atomic_base< unsigned char > [atomic_uchar](#)
- typedef __atomic_base< unsigned int > [atomic_uint](#)
- typedef [atomic_ushort](#) **std::atomic_uint_fast16_t**
- typedef [atomic_uint](#) **std::atomic_uint_fast32_t**
- typedef [atomic_ullong](#) **std::atomic_uint_fast64_t**
- typedef [atomic_uchar](#) **std::atomic_uint_fast8_t**
- typedef [atomic_ushort](#) **std::atomic_uint_least16_t**
- typedef [atomic_uint](#) **std::atomic_uint_least32_t**
- typedef [atomic_ullong](#) **std::atomic_uint_least64_t**
- typedef [atomic_uchar](#) **std::atomic_uint_least8_t**
- typedef [atomic_ullong](#) **std::atomic_uintmax_t**
- typedef [atomic_ulong](#) **std::atomic_uintptr_t**
- typedef __atomic_base< unsigned long long > [atomic_ullong](#)
- typedef __atomic_base< unsigned long > [atomic_ulong](#)
- typedef __atomic_base< unsigned short > [atomic_ushort](#)
- typedef __atomic_base< wchar_t > [atomic_wchar_t](#)
- typedef enum [std::memory_order](#) **std::memory_order**

Enumerations

- enum `std::memory_order` {
`memory_order_relaxed`, `memory_order_consume`, `memory_order_-`
`acquire`, `memory_order_release`,
`memory_order_acq_rel`, `memory_order_seq_cst` }

Functions

- void `std::atomic_flag_wait_explicit` (__atomic_flag_base *, memory_order)
`_GLIBCXX_NOTHROW`
- `std::atomic` __attribute__((const)) __atomic_flag_base * __atomic_flag_for_
address(const void *__z) `_GLIBCXX_NOTHROW`
- memory_order `std::calculate_memory_order` (memory_order __m)
- template<typename _ITp >
bool `std::atomic_compare_exchange_strong` (__atomic_base< _ITp > *__a,
_ITp *__i1, _ITp __i2)
- bool `std::atomic_compare_exchange_strong` (atomic_address *__a, void **_
v1, void *__v2)
- bool `std::atomic_compare_exchange_strong` (atomic_bool *__a, bool *__i1,
bool __i2)
- template<typename _ITp >
bool `std::atomic_compare_exchange_strong_explicit` (__atomic_base< _ITp
> *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)
- bool `std::atomic_compare_exchange_strong_explicit` (atomic_address *__a,
void **__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool `std::atomic_compare_exchange_strong_explicit` (atomic_bool *__a,
bool *__i1, bool __i2, memory_order __m1, memory_order __m2)
- template<typename _ITp >
bool `std::atomic_compare_exchange_weak` (__atomic_base< _ITp > *__a,
_ITp *__i1, _ITp __i2)
- bool `std::atomic_compare_exchange_weak` (atomic_address *__a, void **_
v1, void *__v2)
- bool `std::atomic_compare_exchange_weak` (atomic_bool *__a, bool *__i1,
bool __i2)
- template<typename _ITp >
bool `std::atomic_compare_exchange_weak_explicit` (__atomic_base< _ITp
> *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2)
- bool `std::atomic_compare_exchange_weak_explicit` (atomic_bool *__a, bool
__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool `std::atomic_compare_exchange_weak_explicit` (atomic_address *__a,
void **__v1, void *__v2, memory_order __m1, memory_order __m2)

- `void * std::atomic_exchange (atomic_address *__a, void *__v)`
- `template<typename _ITp >`
`_ITp std::atomic_exchange (__atomic_base<_ITp> *__a, _ITp __i)`
- `bool std::atomic_exchange (atomic_bool *__a, bool __i)`
- `template<typename _ITp >`
`_ITp std::atomic_exchange_explicit (__atomic_base<_ITp> *__a, _ITp __i,`
`memory_order __m)`
- `void * std::atomic_exchange_explicit (atomic_address *__a, void *__v,`
`memory_order __m)`
- `bool std::atomic_exchange_explicit (atomic_bool *__a, bool __i, memory_`
`order __m)`
- `void * std::atomic_fetch_add (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add (__atomic_base<_ITp> *__a, _ITp __i)`
- `void * std::atomic_fetch_add_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add_explicit (__atomic_base<_ITp> *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and (__atomic_base<_ITp> *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and_explicit (__atomic_base<_ITp> *__a, _ITp __i,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or (__atomic_base<_ITp> *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or_explicit (__atomic_base<_ITp> *__a, _ITp __i,`
`memory_order __m)`
- `void * std::atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub (__atomic_base<_ITp> *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub_explicit (__atomic_base<_ITp> *__a, _ITp __i,`
`memory_order __m)`
- `void * std::atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d,`
`memory_order __m)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor (__atomic_base<_ITp> *__a, _ITp __i)`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor_explicit (__atomic_base<_ITp> *__a, _ITp __i,`
`memory_order __m)`
- `void std::atomic_flag_clear (__atomic_flag_base *__a)`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m)`

- `void std::atomic_flag_clear_explicit (__atomic_flag_base *, memory_order) _GLIBCXX_NOTHROW`
- `bool std::atomic_flag_test_and_set (__atomic_flag_base *__a)`
- `bool std::atomic_flag_test_and_set_explicit (__atomic_flag_base *, memory_order) _GLIBCXX_NOTHROW`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m)`
- `template<typename _ITp >`
`bool std::atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_is_lock_free (const atomic_address *__a)`
- `bool std::atomic_is_lock_free (const atomic_bool *__a)`
- `void * std::atomic_load (const atomic_address *__a)`
- `template<typename _ITp >`
`_ITp std::atomic_load (const __atomic_base< _ITp > *__a)`
- `bool std::atomic_load (const atomic_bool *__a)`
- `template<typename _ITp >`
`_ITp std::atomic_load_explicit (const __atomic_base< _ITp > *__a, memory_order __m)`
- `void * std::atomic_load_explicit (const atomic_address *__a, memory_order __m)`
- `bool std::atomic_load_explicit (const atomic_bool *__a, memory_order __m)`
- `void std::atomic_store (atomic_address *__a, void *__v)`
- `template<typename _ITp >`
`void std::atomic_store (__atomic_base< _ITp > *__a, _ITp __i)`
- `void std::atomic_store (atomic_bool *__a, bool __i)`
- `void std::atomic_store_explicit (atomic_bool *__a, bool __i, memory_order __m)`
- `template<typename _ITp >`
`void std::atomic_store_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory_order __m)`
- `void std::atomic_store_explicit (atomic_address *__a, void *__v, memory_order __m)`
- `bool std::atomic< _Tp * >::compare_exchange_strong (_Tp * &, _Tp *, memory_order, memory_order)`
- `bool std::atomic< _Tp * >::compare_exchange_strong (_Tp * &, _Tp *, memory_order=memory_order_seq_cst)`
- `bool std::atomic< _Tp * >::compare_exchange_weak (_Tp * &, _Tp *, memory_order, memory_order)`
- `bool std::atomic< _Tp * >::compare_exchange_weak (_Tp * &, _Tp *, memory_order=memory_order_seq_cst)`
- `_Tp * std::atomic< _Tp * >::fetch_add (ptrdiff_t, memory_order=memory_order_seq_cst)`
- `_Tp * std::atomic< _Tp * >::fetch_sub (ptrdiff_t, memory_order=memory_order_seq_cst)`

- `template<typename _Tp >`
 `_Tp std::kill_dependency (_Tp __y)`

2.34.1 Detailed Description

Components for performing atomic operations.

2.34.2 Define Documentation

2.34.2.1 `#define _GLIBCXX_ATOMIC_PROPERTY`

29.2 Lock-free Property

Definition at line 68 of file `atomic_base.h`.

2.34.3 Typedef Documentation

2.34.3.1 `typedef __atomic_base<char> atomic_char`

`atomic_char`

Definition at line 71 of file `atomicfwd_cxx.h`.

2.34.3.2 `typedef __atomic_base<char16_t> atomic_char16_t`

`atomic_char16_t`

Definition at line 107 of file `atomicfwd_cxx.h`.

2.34.3.3 `typedef __atomic_base<char32_t> atomic_char32_t`

`atomic_char32_t`

Definition at line 110 of file `atomicfwd_cxx.h`.

2.34.3.4 `typedef __atomic_base<int> atomic_int`

`atomic_int`

Definition at line 86 of file `atomicfwd_cxx.h`.

2.34.3.5 typedef __atomic_base<long long> atomic_llong

atomic_llong

Definition at line 98 of file atomicfwd_cxx.h.

2.34.3.6 typedef __atomic_base<long> atomic_long

atomic_long

Definition at line 92 of file atomicfwd_cxx.h.

2.34.3.7 typedef __atomic_base<signed char> atomic_schar

atomic_schar

Definition at line 74 of file atomicfwd_cxx.h.

2.34.3.8 typedef __atomic_base<short> atomic_short

atomic_short

Definition at line 80 of file atomicfwd_cxx.h.

2.34.3.9 typedef __atomic_base<unsigned char> atomic_uchar

atomic_uchar

Definition at line 77 of file atomicfwd_cxx.h.

2.34.3.10 typedef __atomic_base<unsigned int> atomic_uint

atomic_uint

Definition at line 89 of file atomicfwd_cxx.h.

2.34.3.11 typedef __atomic_base<unsigned long long> atomic_ullong

atomic_ullong

Definition at line 101 of file atomicfwd_cxx.h.

2.34.3.12 typedef __atomic_base<unsigned long> atomic_ulong

atomic_ulong

Definition at line 95 of file atomicfwd_cxx.h.

2.34.3.13 typedef __atomic_base<unsigned short> atomic_ushort

atomic_ushort

Definition at line 83 of file atomicfwd_cxx.h.

2.34.3.14 typedef __atomic_base<wchar_t> atomic_wchar_t

atomic_wchar_t

Definition at line 104 of file atomicfwd_cxx.h.

2.34.3.15 typedef enum std::memory_order std::memory_order

Enumeration for memory_order.

2.34.4 Enumeration Type Documentation**2.34.4.1 enum std::memory_order**

Enumeration for memory_order.

Definition at line 47 of file atomic_base.h.

2.34.5 Function Documentation**2.34.5.1 template<typename _Tp > _Tp std::kill_dependency (_Tp __y)
[inline]**

kill_dependency

Definition at line 54 of file atomic.

2.35 Hashes

Collaboration diagram for Hashes:



Classes

- struct `std::hash<_Tp>`
Primary class template hash.
- struct `std::hash<_Tp*>`
Partial specializations for pointer types.

Defines

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

Functions

- `size_t std::_Fnv_hash_bytes` (const void *__ptr, size_t __len, size_t __seed)
- `size_t std::_Hash_bytes` (const void *__ptr, size_t __len, size_t __seed)

2.35.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

2.36 Locales

Classes

- class `std::codecvt<_InternT, _ExternT, _StateT>`

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

- class `std::ctype< _CharT >`

Primary class template `ctype` facet.

This template class defines classification and conversion functions for character sets. It wraps `cctype` functionality. `Ctype` gets used by streams for many I/O operations.

- class `std::ctype< char >`

The `ctype<char>` specialization.

This class defines classification and conversion functions for the `char` type. It gets used by `char` streams for many I/O operations. The `char` specialization provides a number of optimizations as well.

- class `std::ctype< wchar_t >`

The `ctype<wchar_t>` specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.

- class `std::locale`

Container class for localization functionality.

The `locale` class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

- class `std::locale::facet`

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

- class `std::locale::id`

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

- class `std::messages< _CharT >`

Primary class template `messages`.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

- struct `std::messages_base`

Messages facet base class providing catalog typedef.

- class `std::money_base`
Money format ordering data.
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.
- class `std::money_get< _CharT, _InIter >`
Primary class template `money_get`.
This facet encapsulates the code to parse and return a monetary amount from a string.
- class `std::money_put< _CharT, _OutIter >`
Primary class template `money_put`.
This facet encapsulates the code to format and output a monetary amount.
- class `std::moneypunct< _CharT, _Intl >`
Primary class template `moneypunct`.
This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.
- class `std::num_get< _CharT, _InIter >`
Primary class template `num_get`.
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.
- class `std::num_put< _CharT, _OutIter >`
Primary class template `num_put`.
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.
- class `std::numpunct< _CharT >`
Primary class template `numpunct`.
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.
- class `std::time_base`
Time format ordering data.
This class provides an enum representing different orderings of time: day, month, and year.
- class `std::time_get< _CharT, _InIter >`
Primary class template `time_get`.
This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

- class `std::time_put<_CharT, _OutIter >`

Primary class template *time_put*.

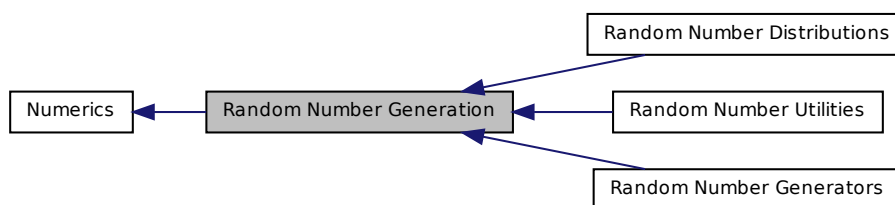
This facet encapsulates the code to format and output dates and times according to formats used by *strftime()*.

2.36.1 Detailed Description

Classes and functions for internationalization and localization.

2.37 Random Number Generation

Collaboration diagram for Random Number Generation:



Namespaces

- namespace `std::__detail`

Modules

- [Random Number Generators](#)
- [Random Number Distributions](#)
- [Random Number Utilities](#)

Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >
_RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`

2.37.1 Detailed Description

A facility for generating random numbers on selected distributions.

2.37.2 Function Documentation

- 2.37.2.1** `template<typename _RealType, size_t __bits, typename
_UniformRandomNumberGenerator > _RealType
std::generate_canonical (_UniformRandomNumberGenerator & __g
)`

A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Definition at line 2802 of file random.tcc.

References `std::log()`, and `std::min()`.

2.38 Regular Expressions

Classes

- class `std::basic_regex< _Ch_type, _Rx_traits >`
- class `std::match_results< _Bi_iter, _Allocator >`
The results of a match or search operation.
- class `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- struct `std::regex_traits< _Ch_type >`
Describes aspects of a regular expression.
- class `std::sub_match< _BiIter >`

Typedefs

- `typedef match_results< const char * > std::cmatch`
- `typedef regex_iterator< const char * > std::cregex_iterator`

- typedef regex_token_iterator< const char * > [std::cregex_token_iterator](#)
- typedef sub_match< const char * > [std::csub_match](#)
- typedef basic_regex< char > [std::regex](#)
- typedef match_results< string::const_iterator > [std::smatch](#)
- typedef regex_iterator< string::const_iterator > [std::sregex_iterator](#)
- typedef regex_token_iterator< string::const_iterator > [std::sregex_token_iterator](#)
- typedef sub_match< string::const_iterator > [std::ssub_match](#)
- typedef match_results< const wchar_t * > [std::wcmatch](#)
- typedef regex_iterator< const wchar_t * > [std::wcregex_iterator](#)
- typedef regex_token_iterator< const wchar_t * > [std::wcregex_token_iterator](#)
- typedef sub_match< const wchar_t * > [std::wcsub_match](#)
- typedef basic_regex< wchar_t > [std::wregex](#)
- typedef match_results< wstring::const_iterator > [std::wsmatch](#)
- typedef regex_iterator< wstring::const_iterator > [std::wsregex_iterator](#)
- typedef regex_token_iterator< wstring::const_iterator > [std::wsregex_token_iterator](#)
- typedef sub_match< wstring::const_iterator > [std::wssub_match](#)

Functions

- template<typename _Bi_iter >
const sub_match< _Bi_iter > & [std::__unmatched_sub](#) ()
- bool [std::regex_traits::isctype](#) (_Ch_type __c, char_class_type __f) const
- template<typename _Biter >
bool [std::operator!=](#) (const sub_match< _Biter > &__lhs, const sub_match< _Biter > &__rhs)
- template<typename _Bi_iter >
bool [std::operator!=](#) (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
bool [std::operator!=](#) (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >
bool [std::operator!=](#) (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
bool [std::operator!=](#) (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter, class _Allocator >
bool [std::operator!=](#) (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &_`
`__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _BiIter >`
`bool std::operator< (const sub_match< _BiIter > &__lhs, const sub_match<`
`_BiIter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &_`
`__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator< (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *_`
`__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream<`
`_Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const basic_`
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_`
`alloc > &__rhs)`
- `template<typename _BiIter >`
`bool std::operator<= (const sub_match< _BiIter > &__lhs, const sub_match<`
`_BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const &__rhs)`

- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const basic_-`
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_-`
`alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const`
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`bool std::operator== (const match_results< _Bi_iter, _Allocator > &__m1,`
`const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _BiIter >`
`bool std::operator== (const sub_match< _BiIter > &__lhs, const sub_match<`
`_BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator> (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const basic_string<`

- ```
typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &_
_rhs)
```
- `template<typename _Bi_iter >`  
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_`  
`traits< _Bi_iter >::value_type const *__rhs)`
  - `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *_`  
`__lhs, const sub_match< _Bi_iter > &__rhs)`
  - `template<typename _Bi_iter >`  
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
  - `template<typename _BiIter >`  
`bool std::operator> (const sub_match< _BiIter > &__lhs, const sub_match<`  
`_BiIter > &__rhs)`
  - `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const basic_`  
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_`  
`alloc > &__rhs)`
  - `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`  
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
  - `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool std::operator>= (const basic_string< typename iterator_traits< _Bi_iter`  
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`  
`&__rhs)`
  - `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`  
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
  - `template<typename _Bi_iter >`  
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`  
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
  - `template<typename _BiIter >`  
`bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match<`  
`_BiIter > &__rhs)`
  - `template<typename _Bi_iter >`  
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`  
`iterator_traits< _Bi_iter >::value_type const *__rhs)`
  - `template<typename _Bi_iter, typename _Allocator >`  
`void std::swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results<`  
`_Bi_iter, _Allocator > &__rhs)`
  - `template<typename _Ch_type, typename _Rx_traits >`  
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _`  
`Ch_type, _Rx_traits > &__rhs)`
  - `int std::regex_traits::value (_Ch_type __ch, int __radix) const`

## Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex\_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _`  
`Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`  
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex\_match (_Bi_iter __first, _Bi_iter __last, const basic_`  
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Allocator, typename _Rx_traits >`  
`bool std::regex\_match (const _Ch_type *__s, match_results< const _Ch_type`  
`*, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type,`  
`typename _Rx_traits >`  
`bool std::regex\_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc`  
`> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _`  
`Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_`  
`type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_`  
`constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`  
`bool std::regex\_match (const _Ch_type *__s, const basic_regex< _Ch_`  
`type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_`  
`constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_`  
`traits >`  
`bool std::regex\_match (const basic_string< _Ch_type, _Ch_traits, _Str_`  
`allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_`  
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_`  
`iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, const basic_`  
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Allocator, class _Rx_traits >`  
`bool std::regex\_search (const _Ch_type *__s, match_results< const _Ch_type *`  
`, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_`  
`constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`bool std::regex\_search (const _Ch_type *__s, const basic_regex< _Ch_`

- ```
type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
constants::match_default)
```
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_`
`allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_`
`constants::match_flag_type __flags=regex_constants::match_default)`
 - `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type,`
`typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_`
`alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_`
`traits, _Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex<`
`_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_`
`constants::match_default)`
 - `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __`
`last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string<`
`_Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_`
`constants::match_default)`
 - `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > std::regex_replace (const basic_string< _Ch_`
`type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const`
`basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __`
`flags=regex_constants::match_default)`

2.38.1 Detailed Description

A facility for performing regular expression pattern matching.

2.38.2 Typedef Documentation

2.38.2.1 `typedef regex_token_iterator<const char*>` `std::cregex_token_iterator`

Token iterator for C-style NULL-terminated strings.

Definition at line 2415 of file `regex.h`.

2.38.2.2 `typedef sub_match<const char*> std::csub_match`

Standard regex submatch over a C-style null-terminated string.

Definition at line 847 of file `regex.h`.

2.38.2.3 typedef basic_regex<char> std::regex

Standard regular expressions.

Definition at line 722 of file regex.h.

**2.38.2.4 typedef regex_token_iterator<string::const_iterator>
std::sregex_token_iterator**

Token iterator for standard strings.

Definition at line 2417 of file regex.h.

2.38.2.5 typedef sub_match<string::const_iterator> std::ssub_match

Standard regex submatch over a standard string.

Definition at line 849 of file regex.h.

**2.38.2.6 typedef regex_token_iterator<const wchar_t*>
std::wregex_token_iterator**

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2420 of file regex.h.

2.38.2.7 typedef sub_match<const wchar_t*> std::wcsub_match

Regex submatch over a C-style null-terminated wide string.

Definition at line 852 of file regex.h.

2.38.2.8 typedef basic_regex<wchar_t> std::wregex

Standard wide-character regular expressions.

Definition at line 725 of file regex.h.

**2.38.2.9 typedef regex_token_iterator<wstring::const_iterator>
std::wsregex_token_iterator**

Token iterator for standard wide-character strings.

Definition at line 2422 of file regex.h.

2.38.2.10 `typedef sub_match<wstring::const_iterator> std::wssub_match`

Regex submatch over a standard wide string.

Definition at line 854 of file regex.h.

2.38.3 Function Documentation**2.38.3.1** `template<typename _Ch_type > bool std::regex_traits<_Ch_type>::isctype (_Ch_type __c, char_class_type __f) const [inherited]`

Determines if `c` is a member of an identified class.

Parameters

c a character.

f a class type (as returned from `lookup_classname`).

Returns

true if the character `c` is a member of the classification represented by `f`, false otherwise.

Exceptions

[*std::bad_cast*](#) if the current locale does not have a ctype facet.

Definition at line 283 of file regex.h.

References `std::__ctype_abstract_base<_CharT>::is()`, `std::regex_traits<_Ch_type>::lookup_classname()`, `std::use_facet()`, and `std::__ctype_abstract_base<_CharT>::widen()`.

2.38.3.2 `template<typename _BiIter > bool std::operator!=(const sub_match<_BiIter> & __lhs, const sub_match<_BiIter> & __rhs) [inline]`

Tests the inequivalence of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 879 of file regex.h.

References `std::sub_match<_BiIter>::compare()`.

2.38.3.3 `template<typename _Bi_iter > bool std::operator!= (typename
iterator_traits<_Bi_iter>::value_type const & __lhs, const
sub_match<_Bi_iter> & __rhs) [inline]`

Tests the inequivalence of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1268 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.4 `template<typename _Bi_iter > bool std::operator!= (const
sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>
>::value_type const & __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1342 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.5 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator!= (const basic_string< typename
iterator_traits<_Bi_iter>::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match<_Bi_iter> & __rhs) [inline]`

Tests the inequivalence of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 955 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.6 `template<typename _Bi_iter > bool std::operator!= (typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the inequivalence of an iterator value and a regular expression submatch.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1120 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.7 `template<typename _Bi_iter, class _Allocator > bool std::operator!=
(const match_results< _Bi_iter, _Allocator > & __m1, const
match_results< _Bi_iter, _Allocator > & __m2) [inline]`

Compares two [match_results](#) for inequality.

Returns

true if the two objects do not refer to the same match, false otherwise.

Definition at line 1779 of file regex.h.

2.38.3.8 `template<typename _Bi_iter > bool std::operator!=(const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const * __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A pointer to a string.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1194 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.9 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator!=(const sub_match< _Bi_iter >
& __lhs, const basic_string< typename iterator_traits< _Bi_iter
>::value_type, _Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* is not equivalent to *rhs*, false otherwise.

Definition at line 1036 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.10 `template<typename _BiIter > bool std::operator< (const
sub_match< _BiIter > & __lhs, const sub_match< _BiIter > &
__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 891 of file regex.h.

```
2.38.3.11 template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator< ( const sub_match< _Bi_iter > & __lhs, const
basic_string< typename iterator_traits< _Bi_iter >::value_type,
_Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1050 of file regex.h.

```
2.38.3.12 template<typename _Bi_iter > bool std::operator< ( typename
iterator_traits< _Bi_iter >::value_type const & __lhs, const
sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1280 of file regex.h.

2.38.3.13 `template<typename _Bi_iter, typename _Ch_traits, typename
_Ch_alloc > bool std::operator< (const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 968 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.14 `template<typename _Bi_iter > bool std::operator< (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1354 of file regex.h.

2.38.3.15 `template<typename _Bi_iter > bool std::operator< (typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1132 of file regex.h.

```
2.38.3.16  template<typename _Bi_iter > bool std::operator< ( const  
            sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter  
            >::value_type const * __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* precedes *rhs*, false otherwise.

Definition at line 1206 of file regex.h.

```
2.38.3.17  template<typename _Ch_type, typename _Ch_traits, typename  
            _Bi_iter > basic_ostream< _Ch_type, _Ch_traits>& std::operator<<  
            ( basic_ostream< _Ch_type, _Ch_traits > & __os, const  
            sub_match< _Bi_iter > & __m ) [inline]
```

Inserts a matched string into an output stream.

Parameters

os The output stream.

m A submatch string.

Returns

the output stream with the submatch string inserted.

Definition at line 1405 of file regex.h.

2.38.3.18 `template<typename _Bi_iter > bool std::operator<= (typename
iterator_traits< _Bi_iter >::value_type const & __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1316 of file regex.h.

2.38.3.19 `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator<= (const sub_match< _Bi_iter > & __lhs, const
basic_string< typename iterator_traits< _Bi_iter >::value_type,
_Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1092 of file regex.h.

2.38.3.20 `template<typename _BiIter > bool std::operator<= (const
sub_match< _BiIter > & __lhs, const sub_match< _BiIter > &
__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 903 of file regex.h.

2.38.3.21 `template<typename _Bi_iter > bool std::operator<= (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1390 of file regex.h.

2.38.3.22 `template<typename _Bi_iter > bool std::operator<= (typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1168 of file regex.h.

2.38.3.23 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator<= (const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1007 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.24 `template<typename _Bi_iter > bool std::operator<= (const
sub_match<_Bi_iter > & __lhs, typename iterator_traits<_Bi_iter
>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* does not succeed *rhs*, false otherwise.

Definition at line 1242 of file regex.h.

2.38.3.25 `template<typename _Bi_iter > bool std::operator== (typename
iterator_traits<_Bi_iter >::value_type const & __lhs, const
sub_match<_Bi_iter > & __rhs) [inline]`

Tests the equivalence of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1255 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.26 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator==(const sub_match< _Bi_iter >
& __lhs, const basic_string< typename iterator_traits< _Bi_iter
>::value_type, _Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1021 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.27 `template<typename _Bi_iter > bool std::operator==(const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1329 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.28 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator==(const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the equivalence of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 940 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.29 `template<typename _Bi_iter > bool std::operator==(typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the equivalence of a C string and a regular expression submatch.

Parameters

lhs A C string.

rhs A regular expression submatch.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1107 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.30 `template<typename _Bi_iter , typename _Allocator > bool
std::operator==(const match_results< _Bi_iter, _Allocator > &
__m1, const match_results< _Bi_iter, _Allocator > & __m2)
[inline]`

Compares two [match_results](#) for equality.

Returns

true if the two objects refer to the same match, false otherwise.

Todo

Implement this function.

2.38.3.31 `template<typename _Bilter > bool std::operator==(const
sub_match<_Bilter > & __lhs, const sub_match<_Bilter > &
__rhs) [inline]`

Tests the equivalence of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 867 of file regex.h.

References `std::sub_match<_Bilter >::compare()`.

2.38.3.32 `template<typename _Bi_iter > bool std::operator==(const
sub_match<_Bi_iter > & __lhs, typename iterator_traits<_Bi_iter
>::value_type const * __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A pointer to a string?

Returns

true if *lhs* is equivalent to *rhs*, false otherwise.

Definition at line 1181 of file regex.h.

References `std::sub_match<_Bilter >::str()`.

2.38.3.33 `template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool std::operator>(const sub_match<_Bi_iter > & __lhs, const
basic_string< typename iterator_traits<_Bi_iter >::value_type,
_Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1064 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.34 `template<typename _Bi_iter > bool std::operator> (typename
iterator_traits<_Bi_iter >::value_type const & __lhs, const
sub_match<_Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1292 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.35 `template<typename _Bi_iter > bool std::operator> (const
sub_match<_Bi_iter > & __lhs, typename iterator_traits<_Bi_iter
>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1366 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.36 `template<typename _Bi_iter, typename _Ch_traits, typename
_Ch_alloc > bool std::operator> (const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 981 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.37 `template<typename _Bi_iter > bool std::operator> (typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1144 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.38 `template<typename _BiIter > bool std::operator> (const
sub_match< _BiIter > & __lhs, const sub_match< _BiIter > &
__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 927 of file regex.h.

References `std::sub_match<_BiIter>::compare()`.

2.38.3.39 `template<typename _Bi_iter > bool std::operator> (const
sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>
>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* succeeds *rhs*, false otherwise.

Definition at line 1218 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.40 `template<typename _Bi_iter > bool std::operator>= (typename
iterator_traits<_Bi_iter>::value_type const & __lhs, const
sub_match<_Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1304 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.41 `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >
bool std::operator>= (const sub_match< _Bi_iter > & __lhs, const
basic_string< typename iterator_traits< _Bi_iter >::value_type,
_Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1078 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.42 `template<typename _Bi_iter > bool std::operator>= (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A const string reference.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1378 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.43 `template<typename _Bi_iter > bool std::operator>= (typename
iterator_traits< _Bi_iter >::value_type const * __lhs, const
sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1156 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.44 `template<typename _Bi_iter , typename _Ch_traits , typename
_Ch_alloc > bool std::operator>= (const basic_string< typename
iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &
__lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

lhs A string.

rhs A regular expression submatch.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 994 of file regex.h.

References `std::sub_match<_BiIter>::str()`.

2.38.3.45 `template<typename _BiIter > bool std::operator>= (const
sub_match< _BiIter > & __lhs, const sub_match< _BiIter > &
__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

lhs First regular expression submatch.

rhs Second regular expression submatch.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 915 of file regex.h.

References `std::sub_match<_BiIter>::compare()`.

2.38.3.46 `template<typename _Bi_iter > bool std::operator>= (const
sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter
>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

lhs A regular expression submatch.

rhs A string.

Returns

true if *lhs* does not precede *rhs*, false otherwise.

Definition at line 1230 of file regex.h.

References `std::sub_match< _BiIter >::str()`.

2.38.3.47 `template<typename _Ch_traits , typename _Ch_alloc , typename
_Allocator , typename _Ch_type , typename _Rx_traits >
bool std::regex_match (const basic_string< _Ch_type,
_Ch_traits, _Ch_alloc > & __s, match_results< typename
basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,
_Allocator > & __m, const basic_regex< _Ch_type, _Rx_traits
> & __re, regex_constants::match_flag_type __flags =
regex_constants::match_default) [inline]`

Determines if there is a match between the regular expression *e* and a string.

Parameters

s The string to match.

m The match results.

re The regular expression.

flags Controls how the regular expression is matched.

Return values

true A match exists.

false Otherwise.

Exceptions

an exception of type `regex_error`.

Definition at line 1903 of file regex.h.

References `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

2.38.3.48 `template<typename _Ch_type, typename _Allocator, typename _Rx_traits> bool std::regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Allocator > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f = regex_constants::match_default) [inline]`

Determines if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

- `s` The C-style null-terminated string to match.
- `m` The match results.
- `re` The regular expression.
- `f` Controls how the regular expression is matched.

Return values

- `true` A match exists.
- `false` Otherwise.

Exceptions

- `an` exception of type `regex_error`.

Definition at line 1879 of file regex.h.

References `std::regex_match()`.

2.38.3.49 `template<typename _Ch_type, class _Rx_traits> bool std::regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f = regex_constants::match_default) [inline]`

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

- `s` The C-style null-terminated string to match.

re The regular expression.

f Controls how the regular expression is matched.

Return values

true A match exists.

false Otherwise.

Exceptions

an exception of type [regex_error](#).

Definition at line 1926 of file regex.h.

References `std::regex_match()`.

```
2.38.3.50 template<typename _Ch_traits , typename _Str_allocator ,
typename _Ch_type , typename _Rx_traits > bool std::regex_match
( const basic_string< _Ch_type, _Ch_traits, _Str_allocator
> & __s, const basic_regex< _Ch_type, _Rx_traits >
& __re, regex_constants::match_flag_type __flags =
regex_constants::match_default ) [inline]
```

Indicates if there is a match between the regular expression *e* and a string.

Parameters

s [IN] The string to match.

re [IN] The regular expression.

flags [IN] Controls how the regular expression is matched.

Return values

true A match exists.

false Otherwise.

Exceptions

an exception of type [regex_error](#).

Definition at line 1948 of file regex.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

2.38.3.51 `template<typename _Bi_iter , typename _Allocator , typename
_Ch_type , typename _Rx_traits > bool std::regex_match
(_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter,
_Allocator > & __m, const basic_regex< _Ch_type, _Rx_traits
> & __re, regex_constants::match_flag_type __flags =
regex_constants::match_default)`

Determines if there is a match between the regular expression *e* and all of the character sequence [first, last).

Parameters

- s* Start of the character sequence to match.
- e* One-past-the-end of the character sequence to match.
- m* The match results.
- re* The regular expression.
- flags* Controls how the regular expression is matched.

Return values

- true* A match exists.
- false* Otherwise.

Exceptions

- an* exception of type `regex_error`.

Todo

- Implement this function.

Definition at line 1823 of file `regex.h`.

Referenced by `std::regex_match()`.

2.38.3.52 `template<typename _Bi_iter , typename _Ch_type , typename
_Rx_traits > bool std::regex_match (_Bi_iter __first,
_Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits
> & __re, regex_constants::match_flag_type __flags =
regex_constants::match_default)`

Indicates if there is a match between the regular expression *e* and all of the character sequence [first, last).

Parameters

- first* Beginning of the character sequence to match.

last One-past-the-end of the character sequence to match.

re The regular expression.

flags Controls how the regular expression is matched.

Return values

true A match exists.

false Otherwise.

Exceptions

an exception of type `regex_error`.

Definition at line 1854 of file `regex.h`.

References `std::regex_match()`.

```
2.38.3.53 template<typename _Rx_traits , typename _Ch_type  
> basic_string<_Ch_type> std::regex_replace ( const  
basic_string<_Ch_type> & __s, const basic_regex<  
_Ch_type, _Rx_traits> & __e, const basic_string<_Ch_type  
> & __fmt, regex_constants::match_flag_type __flags =  
regex_constants::match_default ) [inline]
```

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Parameters

s

e

fmt

flags

Returns

a copy of string *s* with replacements.

Exceptions

an exception of type `regex_error`.

Definition at line 2127 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_replace()`.

```

2.38.3.54 template<typename _Out_iter , typename _Bi_iter , typename
        _Rx_traits , typename _Ch_type > _Out_iter std::regex_replace
        ( _Out_iter __out, _Bi_iter __first, _Bi_iter __last, const
        basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string<
        _Ch_type > & __fmt, regex_constants::match_flag_type __flags =
        regex_constants::match_default ) [inline]

```

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Parameters

out
first
last
e
fmt
flags

Returns

out

Exceptions

an exception of type [regex_error](#).

Todo

Implement this function.

Definition at line 2107 of file regex.h.

Referenced by `std::regex_replace()`.

```

2.38.3.55 template<typename _Ch_traits , typename _Ch_alloc , typename
        _Allocator , typename _Ch_type , typename _Rx_traits >
        bool std::regex_search ( const basic_string< _Ch_type,
        _Ch_traits, _Ch_alloc > & __s, match_results< typename
        basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,
        _Allocator > & __m, const basic_regex< _Ch_type,
        _Rx_traits > & __e, regex_constants::match_flag_type __f =
        regex_constants::match_default ) [inline]

```

Searches for a regular expression within a string.

Parameters

s [IN] A C++ string to search for the regex.
m [OUT] The set of regex matches.
e [IN] The regex to search for in *s*.
f [IN] The search flags.

Return values

true A match was found within the string.
false No match was found within the string, the content of *m* is undefined.

Exceptions

an exception of type `regex_error`.

Definition at line 2081 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_search()`.

```
2.38.3.56  template<typename _Bi_iter , typename _Allocator , typename
           _Ch_type , typename _Rx_traits > bool std::regex_search
           ( _Bi_iter __first, _Bi_iter __last, match_results<_Bi_iter,
           _Allocator > & __m, const basic_regex<_Ch_type, _Rx_traits
           > & __re, regex_constants::match_flag_type __flags =
           regex_constants::match_default ) [inline]
```

Searches for a regular expression within a range.

Parameters

first [IN] The start of the string to search.
last [IN] One-past-the-end of the string to search.
m [OUT] The match results.
re [IN] The regular expression to search for.
flags [IN] Search policy flags.

Return values

true A match was found within the string.
false No match was found within the string, the content of *m* is undefined.

Exceptions

an exception of type `regex_error`.

Todo

Implement this function.

Definition at line 1973 of file regex.h.

Referenced by std::regex_search().

```
2.38.3.57  template<typename _Ch_type , typename _Rx_traits > bool
std::regex_search ( const _Ch_type * __s, const basic_regex<
_Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type
__f= regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

Parameters

s [IN] The C-string to search.
e [IN] The regular expression to search for.
f [IN] Search policy flags.

Return values

true A match was found within the string.
false No match was found within the string.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Exceptions

an exception of type [regex_error](#).

Definition at line 2038 of file regex.h.

References std::regex_search().

```
2.38.3.58  template<typename _Ch_type , class _Allocator , class _Rx_traits >
bool std::regex_search ( const _Ch_type * __s, match_results< const
_Ch_type *, _Allocator > & __m, const basic_regex< _Ch_type,
_Rx_traits > & __e, regex_constants::match_flag_type __f=
regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

Parameters

s [IN] A C-string to search for the regex.

m [OUT] The set of regex matches.

e [IN] The regex to search for in *s*.

f [IN] The search flags.

Return values

true A match was found within the string.

false No match was found within the string, the content of *m* is undefined.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Exceptions

an exception of type [regex_error](#).

Definition at line 2018 of file regex.h.

References `std::regex_search()`.

```
2.38.3.59 template<typename _Bi_iter , typename _Ch_type , typename
    _Rx_traits > bool std::regex_search ( _Bi_iter __first,
    _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits
    > & __re, regex_constants::match_flag_type __flags =
    regex_constants::match_default ) [inline]
```

Searches for a regular expression within a range.

Parameters

first [IN] The start of the string to search.

last [IN] One-past-the-end of the string to search.

re [IN] The regular expression to search for.

flags [IN] Search policy flags.

Return values

true A match was found within the string.

false No match was found within the string.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Exceptions

an exception of type `regex_error`.

Definition at line 1994 of file `regex.h`.

References `std::regex_search()`.

```
2.38.3.60 template<typename _Ch_traits , typename _String_allocator ,
typename _Ch_type , typename _Rx_traits > bool std::regex_search
( const basic_string< _Ch_type, _Ch_traits, _String_allocator
> & __s, const basic_regex< _Ch_type, _Rx_traits >
& __e, regex_constants::match_flag_type __flags =
regex_constants::match_default ) [inline]
```

Searches for a regular expression within a string.

Parameters

s [IN] The string to search.

e [IN] The regular expression to search for.

flags [IN] Search policy flags.

Return values

true A match was found within the string.

false No match was found within the string.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Exceptions

an exception of type `regex_error`.

Definition at line 2058 of file `regex.h`.

References `std::regex_search()`.

2.38.3.61 `template<typename _Ch_type, typename _Rx_traits > void
std::swap (basic_regex< _Ch_type, _Rx_traits > & __lhs,
basic_regex< _Ch_type, _Rx_traits > & __rhs) [inline]`

Swaps the contents of two regular expression objects.

Parameters

lhs First regular expression.

rhs Second regular expression.

Definition at line 737 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

2.38.3.62 `template<typename _Bi_iter, typename _Allocator > void std::swap
(match_results< _Bi_iter, _Allocator > & __lhs, match_results<
_Bi_iter, _Allocator > & __rhs) [inline]`

Swaps two match results.

Parameters

lhs A match result.

rhs A match result.

The contents of the two [match_results](#) objects are swapped.

Definition at line 1793 of file regex.h.

References `std::match_results< _Bi_iter, _Allocator >::swap()`.

2.38.3.63 `template<typename _Ch_type > int std::regex_traits< _Ch_type
>::value (_Ch_type __ch, int __radix) const [inherited]`

Converts a digit to an int.

Parameters

ch a character representing a digit.

radix the radix if the numeric conversion (limited to 8, 10, or 16).

Returns

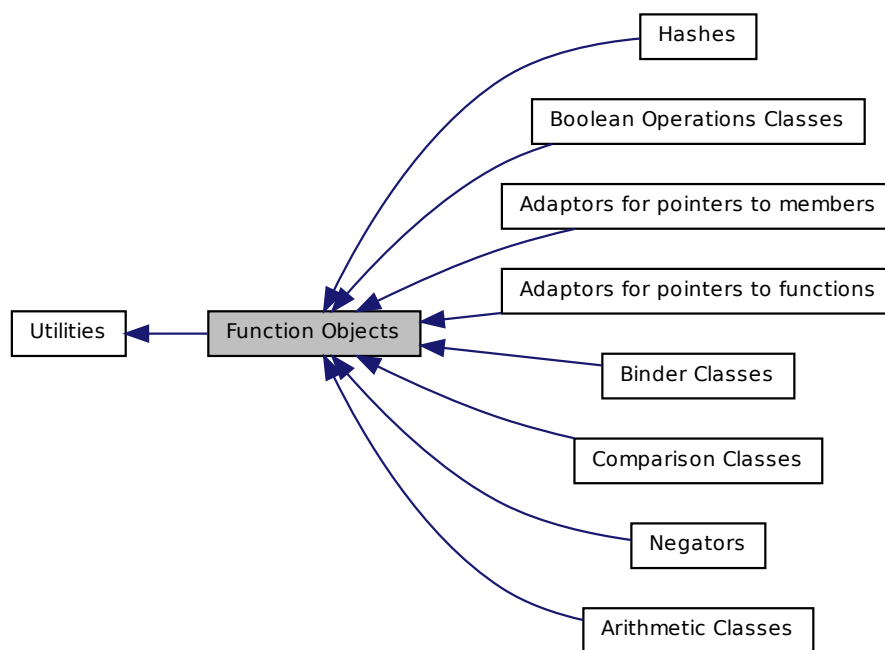
the value represented by the digit *ch* in base *radix* if the character *ch* is a valid digit in base *radix*; otherwise returns -1.

Definition at line 319 of file regex.h.

References `std::basic_ios< _CharT, _Traits >::fail()`.

2.39 Function Objects

Collaboration diagram for Function Objects:



Classes

- struct `std::binary_function< _Arg1, _Arg2, _Result >`
- class `std::function< _Res(_ArgTypes...)>`
*Primary class template for `std::function`.
 Polymorphic function wrapper.*
- class `std::reference_wrapper< _Tp >`

Primary class template for *reference_wrapper*.

- struct `std::unary_function<_Arg, _Result>`

Modules

- [Binder Classes](#)
- [Hashes](#)
- [Arithmetic Classes](#)
- [Comparison Classes](#)
- [Boolean Operations Classes](#)
- [Negators](#)
- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)

Functions

- `template<typename _Tp, typename _Class>`
`_Mem_fn<_Tp _Class::*> std::mem_fn (_Tp _Class::* __pm)`

2.39.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform(a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will be inlined directly.

The standard functors are derived from structs named `unary_function` and `binary_function`. These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.

2.39.2 Function Documentation

2.39.2.1 `template<typename _Tp , typename _Class > _Mem_fn<_Tp
_Class::*> std::mem_fn (_Tp _Class::* __pm) [inline]`

Returns a function object that forwards to the member pointer *pm*.

Definition at line 810 of file functional.

2.40 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



Classes

- struct `std::divides<_Tp>`
One of the [math functors](#).
- struct `std::minus<_Tp>`
One of the [math functors](#).
- struct `std::modulus<_Tp>`
One of the [math functors](#).
- struct `std::multiplies<_Tp>`
One of the [math functors](#).
- struct `std::negate<_Tp>`
One of the [math functors](#).
- struct `std::plus<_Tp>`
One of the [math functors](#).

2.40.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for [the base classes](#) for examples of their use.

2.41 Comparison Classes

Collaboration diagram for Comparison Classes:



Classes

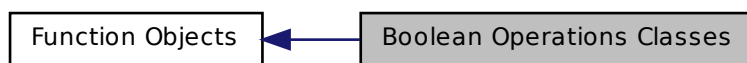
- struct `std::equal_to< _Tp >`
One of the [comparison functors](#).
- struct `std::greater< _Tp >`
One of the [comparison functors](#).
- struct `std::greater_equal< _Tp >`
One of the [comparison functors](#).
- struct `std::less< _Tp >`
One of the [comparison functors](#).
- struct `std::less_equal< _Tp >`
One of the [comparison functors](#).
- struct `std::not_equal_to< _Tp >`
One of the [comparison functors](#).

2.41.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

2.42 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



Classes

- struct `std::logical_and< _Tp >`
*One of the **Boolean operations functors**.*
- struct `std::logical_not< _Tp >`
*One of the **Boolean operations functors**.*
- struct `std::logical_or< _Tp >`
*One of the **Boolean operations functors**.*

2.42.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, `||`, and `!`.

2.43 Negators

Collaboration diagram for Negators:



Classes

- class `std::binary_negate< _Predicate >`
One of the [negation functors](#).
- class `std::unary_negate< _Predicate >`
One of the [negation functors](#).

Functions

- `template<typename _Predicate >`
`unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`
`binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`

2.43.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};
  
```

```
std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
```

The call to `find_if` will locate the first index (*i*) of *v* for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

2.43.2 Function Documentation

2.43.2.1 `template<typename _Predicate > unary_negate<_Predicate>`
`std::not1 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 364 of file `stl_function.h`.

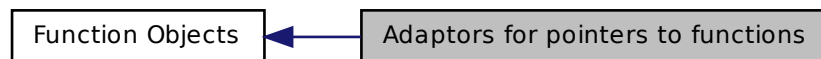
2.43.2.2 `template<typename _Predicate > binary_negate<_Predicate>`
`std::not2 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 389 of file `stl_function.h`.

2.44 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



Classes

- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`
One of the [adaptors for function pointers](#).

- class `std::pointer_to_unary_function< _Arg, _Result >`

One of the [adaptors for function pointers](#).

Functions

- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_`
`Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_`
`Result(*__x)(_Arg1, _Arg2))`

2.44.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes [unary_function](#) and [binary_function](#)). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor [pointer_to_unary_function](#) is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor [pointer_to_binary_function](#) does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

2.44.2 Function Documentation

2.44.2.1 `template<typename _Arg, typename _Result >`
`pointer_to_unary_function<_Arg, _Result> std::ptr_fun (`
`_Result(*)(_Arg) __x) [inline]`

One of the [adaptors for function pointers](#).

Definition at line 437 of file `stl_function.h`.

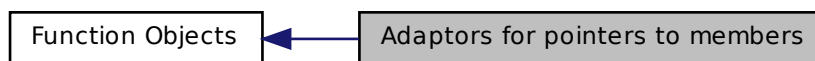
2.44.2.2 `template<typename _Arg1 , typename _Arg2 , typename _Result >
 pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun (
 _Result(*)(_Arg1, _Arg2) __x) [inline]`

One of the [adaptors for function pointers](#).

Definition at line 463 of file `stl_function.h`.

2.45 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



Classes

- class `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun_ref_t< _Ret, _Tp >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun_t< _Ret, _Tp >`
One of the [adaptors for member /// pointers](#).
- class `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).
- class `std::mem_fun1_t< _Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).

- class `std::mem_fun_ref_t<_Ret, _Tp>`
One of the [adaptors](#) for member /// pointers.
- class `std::mem_fun_t<_Ret, _Tp>`
One of the [adaptors](#) for member /// pointers.

Functions

- `template<typename _Ret, typename _Tp>`
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp>`
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`

2.45.1 Detailed Description

There are a total of $8 = 2^3$ function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

2.46 Heap

Collaboration diagram for Heap:



Functions

- `template<typename _RandomAccessIterator >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last)`

2.46.1 Function Documentation

2.46.1.1 `template<typename _RandomAccessIterator > bool std::is_heap (`
 `_RandomAccessIterator __first, _RandomAccessIterator __last)`
 `[inline]`

Determines whether a range is a heap.

Parameters

first Start of range.

last End of range.

Returns

True if range is a heap, false otherwise.

Definition at line 558 of file `stl_heap.h`.

References `std::is_heap_until()`.

2.46.1.2 `template<typename _RandomAccessIterator , typename`
 `_Compare > bool std::is_heap (_RandomAccessIterator __first,`
 `_RandomAccessIterator __last, _Compare __comp) [inline]`

Determines whether a range is a heap using comparison functor.

Parameters

first Start of range.

last End of range.

comp Comparison functor to use.

Returns

True if range is a heap, false otherwise.

Definition at line 571 of file `stl_heap.h`.

References `std::is_heap_until()`.

2.46.1.3 `template<typename _RandomAccessIterator >`
 `_RandomAccessIterator std::is_heap_until (_RandomAccessIterator`
 `__first, _RandomAccessIterator __last) [inline]`

Search the end of a heap.

Parameters

first Start of range.

last End of range.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is a heap.

Definition at line 510 of file `stl_heap.h`.

References `std::distance()`.

2.46.1.4 `template<typename _RandomAccessIterator , typename _Compare >
_RandomAccessIterator std::is_heap_until (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp)
[inline]`

Search the end of a heap using comparison functor.

Parameters

first Start of range.

last End of range.

comp Comparison functor to use.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*first*, *last*) for which the range [*first*, *i*) is a heap. Comparisons are made using *comp*.

Definition at line 536 of file `stl_heap.h`.

References `std::distance()`.

Referenced by `std::is_heap()`.

2.46.1.5 `template<typename _RandomAccessIterator > void std::make_heap (`
`_RandomAccessIterator __first, _RandomAccessIterator __last)`

Construct a heap over a range.

Parameters

first Start of heap.

last End of heap.

This operation makes the elements in [first,last) into a heap.

Definition at line 373 of file stl_heap.h.

```
2.46.1.6  template<typename _RandomAccessIterator , typename _Compare  
            > void std::make_heap ( _RandomAccessIterator __first,  
            _RandomAccessIterator __last, _Compare __comp )
```

Construct a heap over a range using comparison functor.

Parameters

first Start of heap.

last End of heap.

comp Comparison functor to use.

This operation makes the elements in [first,last) into a heap. Comparisons are made using comp.

Definition at line 413 of file stl_heap.h.

Referenced by std::__heap_select(), std::partial_sort_copy(), and std::priority_queue<_Tp, _Sequence, _Compare >::priority_queue().

```
2.46.1.7  template<typename _RandomAccessIterator > void std::pop_heap (   
            _RandomAccessIterator __first, _RandomAccessIterator __last )  
            [inline]
```

Pop an element off a heap.

Parameters

first Start of heap.

last End of heap.

This operation pops the top of the heap. The elements first and last-1 are swapped and [first,last-1) is made into a heap.

Definition at line 276 of file stl_heap.h.

```

2.46.1.8  template<typename _RandomAccessIterator , typename _Compare
            > void std::pop_heap ( _RandomAccessIterator __first,
            _RandomAccessIterator __last, _Compare __comp ) [inline]

```

Pop an element off a heap using comparison functor.

Parameters

first Start of heap.

last End of heap.

comp Comparison functor to use.

This operation pops the top of the heap. The elements *first* and *last*-1 are swapped and [*first*,*last*-1) is made into a heap. Comparisons are made using *comp*.

Definition at line 350 of file `stl_heap.h`.

Referenced by `std::priority_queue< _Tp, _Sequence, _Compare >::pop()`.

```

2.46.1.9  template<typename _RandomAccessIterator , typename _Compare
            > void std::push_heap ( _RandomAccessIterator __first,
            _RandomAccessIterator __last, _Compare __comp ) [inline]

```

Push an element onto a heap using comparison functor.

Parameters

first Start of heap.

last End of heap + element.

comp Comparison functor.

This operation pushes the element at *last*-1 onto the valid heap over the range [*first*,*last*-1). After completion, [*first*,*last*) is a valid heap. Compare operations are performed using *comp*.

Definition at line 203 of file `stl_heap.h`.

Referenced by `std::priority_queue< _Tp, _Sequence, _Compare >::push()`.

```

2.46.1.10 template<typename _RandomAccessIterator > void std::push_heap (
            _RandomAccessIterator __first, _RandomAccessIterator __last )
            [inline]

```

Push an element onto a heap.

Parameters

first Start of heap.

last End of heap + element.

This operation pushes the element at last-1 onto the valid heap over the range [first,last-1). After completion, [first,last) is a valid heap.

Definition at line 154 of file stl_heap.h.

2.46.1.11 `template<typename _RandomAccessIterator, typename _Compare
> void std::sort_heap (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

Sort a heap using comparison functor.

Parameters

first Start of heap.

last End of heap.

comp Comparison functor to use.

This operation sorts the valid heap in the range [first,last). Comparisons are made using comp.

Definition at line 481 of file stl_heap.h.

Referenced by std::partial_sort(), and std::partial_sort_copy().

2.46.1.12 `template<typename _RandomAccessIterator > void std::sort_heap (`
`_RandomAccessIterator __first, _RandomAccessIterator __last)`

Sort a heap.

Parameters

first Start of heap.

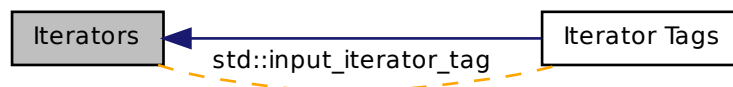
last End of heap.

This operation sorts the valid heap in the range [first,last).

Definition at line 452 of file stl_heap.h.

2.47 Iterators

Collaboration diagram for Iterators:



Classes

- struct `std::__iterator_traits<_Iterator, bool>`
Traits class for iterators.
- class `std::back_insert_iterator<_Container>`
Turns assignment into insertion.
- class `std::front_insert_iterator<_Container>`
Turns assignment into insertion.
- struct `std::input_iterator_tag`
Marking input iterators.
- class `std::insert_iterator<_Container>`
Turns assignment into insertion.
- class `std::istream_iterator<_Tp, _CharT, _Traits, _Dist>`
Provides input iterator semantics for streams.
- class `std::istreambuf_iterator<_CharT, _Traits>`
Provides input iterator semantics for streambufs.
- struct `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>`
Common iterator class.
- struct `std::iterator_traits<_Tp*>`

Partial specialization for pointer types.

- struct `std::iterator_traits< const _Tp * >`
Partial specialization for const pointer types.
- class `std::move_iterator< _Iterator >`
- class `std::ostream_iterator< _Tp, _CharT, _Traits >`
Provides output iterator semantics for streams.
- class `std::ostreambuf_iterator< _CharT, _Traits >`
Provides output iterator semantics for streambufs.
- class `std::reverse_iterator< _Iterator >`

Modules

- [Iterator Tags](#)

Functions

- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`
`iterator< _CharT > >::__type std::__copy_move_a2 (_CharT * __first, _CharT`
`* __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`
`iterator< _CharT > >::__type std::__copy_move_a2 (const _CharT * __first,`
`const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type`
`std::__copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_`
`iterator< _CharT > __last, _CharT * __result)`
- `template<typename _Iter >`
`iterator_traits< _Iter >::iterator_category std::__iterator_category (const _Iter`
`&)`
- `template<typename _Container >`
`back_insert_iterator< _Container > std::back_inserter (_Container & __x)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_`
`iterator< _CharT > >::__type std::copy (istreambuf_iterator< _CharT > _`
`__first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`
`__result)`

- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_`
`iterator< _CharT > >::__type std::find (istreambuf_iterator< _CharT >`
`__first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Container >`
`front_insert_iterator< _Container > std::front_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator >`
`insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::make_move_iterator (const _Iterator &__i)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a,`
`const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Iterator >`
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_`
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__`
`__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _`
`Iterator >::__difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator`
`>::__difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >::__difference_type std::operator- (const reverse_`
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_`
`iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const`
`reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a,`
`const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_`
`iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &_`
`__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_`
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const`
`reverse_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`

2.47.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

2.47.2 Function Documentation

2.47.2.1 `template<typename _Iter > iterator_traits<_Iter>::iterator_category` `std::__iterator_category (const _Iter &) [inline]`

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 200 of file `stl_iterator_base_types.h`.

Referenced by `std::advance()`, `std::copy_n()`, `__gnu_cxx::copy_n()`, `std::distance()`, `__gnu_cxx::distance()`, `std::find()`, `std::find_end()`, `std::find_if()`, `std::find_if_not()`, `std::partition()`, `std::reverse()`, `std::search_n()`, `std::uninitialized_copy_n()`, `__gnu_cxx::uninitialized_copy_n()`, and `std::unique_copy()`.

2.47.2.2 `template<typename _Container > back_insert_iterator<_Container>` `std::back_inserter (_Container & __x) [inline]`

Parameters

x A container of arbitrary type.

Returns

An instance of [back_insert_iterator](#) working on **x**.

This wrapper function helps in creating [back_insert_iterator](#) instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 471 of file `stl_iterator.h`.

Referenced by `std::regex_replace()`.

2.47.2.3 `template<typename _Container > front_insert_iterator<_Container>
std::front_inserter (_Container & __x) [inline]`

Parameters

x A container of arbitrary type.

Returns

An instance of `front_insert_iterator` working on **x**.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 561 of file `stl_iterator.h`.

2.47.2.4 `template<typename _Container , typename _Iterator >
insert_iterator<_Container> std::inserter (_Container & __x,
_Iterator __i) [inline]`

Parameters

x A container of arbitrary type.

Returns

An instance of `insert_iterator` working on **x**.

This wrapper function helps in creating `insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 675 of file `stl_iterator.h`.

2.47.2.5 `template<class _Tp , class _CharT , class _Traits , class _Dist > bool
std::operator!=(const istream_iterator< _Tp, _CharT, _Traits, _Dist
> & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &
__y) [inline]`

Return false if **x** and **y** are both end or not end, or **x** and **y** are the same.

Definition at line 135 of file `stream_iterator.h`.

2.47.2.6 `template<typename _Iterator > bool std::operator==(const reverse_iterator< _Iterator > & __x, const reverse_iterator< _Iterator > & __y) [inline]`

Parameters

x A reverse_iterator.

y A reverse_iterator.

Returns

A simple bool.

Reverse iterators forward many operations to their underlying base() iterators. Others are implemented in terms of one another.

Definition at line 283 of file stl_iterator.h.

References `std::reverse_iterator< _Iterator >::base()`.

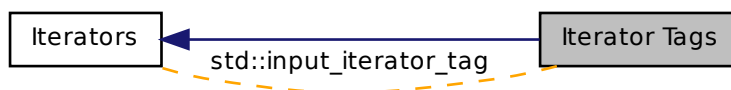
2.47.2.7 `template<typename _Tp , typename _CharT , typename _Traits , typename _Dist > bool std::operator==(const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y) [inline]`

Return true if *x* and *y* are both end or not end, or *x* and *y* are the same.

Definition at line 128 of file stream_iterator.h.

2.48 Iterator Tags

Collaboration diagram for Iterator Tags:



Classes

- struct `std::bidirectional_iterator_tag`

Bidirectional iterators support a superset of forward iterator /// operations.

- struct `std::forward_iterator_tag`

Forward iterators support a superset of input iterator operations.

- struct `std::input_iterator_tag`

Marking input iterators.

- struct `std::output_iterator_tag`

Marking output iterators.

- struct `std::random_access_iterator_tag`

Random-access iterators support a superset of bidirectional /// iterator operations.

2.48.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

2.49 Strings

Collaboration diagram for Strings:



Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`

Managing sequences of characters and character-like objects.

Typedefs

- typedef `basic_string< char >` `std::string`
- typedef `basic_string< char16_t >` `std::u16string`
- typedef `basic_string< char32_t >` `std::u32string`
- typedef `basic_string< wchar_t >` `std::wstring`

2.49.1 Typedef Documentation

2.49.1.1 typedef `basic_string<char>` `std::string`

A string of `char`.

Definition at line 63 of file `stringfwd.h`.

2.49.1.2 typedef `basic_string<char16_t>` `std::u16string`

A string of `char16_t`.

Definition at line 77 of file `stringfwd.h`.

2.49.1.3 typedef `basic_string<char32_t>` `std::u32string`

A string of `char32_t`.

Definition at line 78 of file `stringfwd.h`.

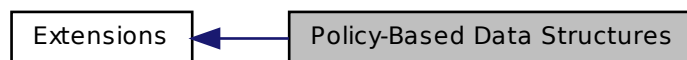
2.49.1.4 typedef `basic_string<wchar_t>` `std::wstring`

A string of `wchar_t`.

Definition at line 68 of file `stringfwd.h`.

2.50 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Classes

- class `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >`
An abstract basic hash-based associative container.
- class `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`
An abstract basic tree-like (tree, trie) associative container.
- class `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`
A concrete collision-chaining hash-based associative container.
- class `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`
An abstract basic associative container.
- class `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >`
A concrete general-probing hash-based associative container.
- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`
A list-update based associative container.
- class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`
A concrete basic tree-based associative container.

- class `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`

A concrete basic trie-based associative container.

Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS_C_DEC`
- `#define PB_DS_TRIE_NODE_AND_ITS_TRAITS`

2.50.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html

2.51 Random Number Generators

Collaboration diagram for Random Number Generators:



Classes

- class `std::discard_block_engine< _RandomNumberEngine, __p, __r >`
- class `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine< _UIntType, __a, __c, __m >`

A model of a linear congruential random number generator.

- class `std::random_device`
- class `std::shuffle_order_engine< _RandomNumberEngine, __k >`

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits __w.

Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` **`std::minstd_rand`**
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` **`std::minstd_rand0`**
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` **`std::mt19937`**
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL >` **`std::mt19937_64`**
- typedef `discard_block_engine< ranlux24_base, 223, 23 >` **`std::ranlux24`**
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >` **`std::ranlux24_base`**
- typedef `discard_block_engine< ranlux48_base, 389, 11 >` **`std::ranlux48`**
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 >` **`std::ranlux48_base`**

Functions

- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool `std::operator!=` (const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &__lhs, const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &__rhs)

- `template<typename _RandomNumberEngine, size_t __k>`
`bool std::operator!= (const std::shuffle_order_engine< _-`
`RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine<`
`_RandomNumberEngine, __k > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-`
`UIntType __f>`
`bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, _-`
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const`
`std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`
`__s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`
`bool std::operator!= (const std::independent_bits_engine< _-`
`RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_-`
`bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`
`bool std::operator!= (const std::discard_block_engine< _-`
`RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine<`
`_RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w,`
`__s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w,`
`__s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT`
`, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _-`
`RandomNumberEngine, __w, _UIntType > &__x)`

2.51.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an `operator()` that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

To be documented.

Table 2.1: Random Number Generator Requirements

2.51.2 Typedef Documentation

2.51.2.1 `typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand`

An alternative LCR (Lehmer Generator function).

Definition at line 1476 of file random.h.

2.51.2.2 `typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0`

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

Definition at line 1470 of file random.h.

2.51.2.3 `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937`

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1492 of file random.h.

2.51.2.4 `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL> std::mt19937_64`

An alternative Mersenne Twister.

Definition at line 1504 of file random.h.

2.51.3 Function Documentation

2.51.3.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool std::operator!=(const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs) [inline]`

Compares two linear congruential random number generator objects of the same type for inequality.

Parameters

`__lhs` A linear congruential random number generator object.

`__rhs` Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 334 of file random.h.

2.51.3.2 `template<typename _RandomNumberEngine, size_t __k> bool std::operator!=(const std::shuffle_order_engine< _RandomNumberEngine, __k > & __lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > & __rhs) [inline]`

Compares two shuffle_order_engine random number generator objects of the same type for inequality.

Parameters

`__lhs` A shuffle_order_engine random number generator object.

`__rhs` Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1459 of file random.h.


```

2.51.3.3  template<typename _UIntType , size_t __w, size_t __n, size_t __m,
size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,
_UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f> bool std::operator!=( const std::mersenne_twister_engine<
_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l,
__f> & __lhs, const std::mersenne_twister_engine< _UIntType, __w,
__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f> & __rhs )
[inline]

```

Compares two % mersenne_twister_engine random number generator objects of the same type for inequality.

Parameters

__lhs A % mersenne_twister_engine random number generator object.

__rhs Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 570 of file random.h.

```

2.51.3.4  template<typename _RandomNumberEngine , size_t
__w, typename _UIntType > bool std::operator!=( const
std::independent_bits_engine< _RandomNumberEngine, __w,
_UIntType > & __lhs, const std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType > & __rhs ) [inline]

```

Compares two independent_bits_engine random number generator objects of the same type for inequality.

Parameters

__lhs A independent_bits_engine random number generator object.

__rhs Another independent_bits_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1203 of file random.h.

2.51.3.5 `template<typename _RandomNumberEngine, size_t __p, size_t __r> bool std::operator!=(const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs) [inline]`

Compares two `discard_block_engine` random number generator objects of the same type for inequality.

Parameters

`__lhs` A `discard_block_engine` random number generator object.

`__rhs` Another `discard_block_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1001 of file `random.h`.

2.51.3.6 `template<typename _UIntType, size_t __w, size_t __s, size_t __r> bool std::operator!=(const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs) [inline]`

Compares two `% subtract_with_carry_engine` random number generator objects of the same type for inequality.

Parameters

`__lhs` A `% subtract_with_carry_engine` random number generator object.

`__rhs` Another `% subtract_with_carry_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 773 of file `random.h`.

```

2.51.3.7 template<typename _RandomNumberEngine, size_t __w,
                typename _UIntType, typename _CharT, typename _Traits
                > std::basic_ostream<_CharT, _Traits>& std::operator<<
                ( std::basic_ostream<_CharT, _Traits> & __os, const
                std::independent_bits_engine<_RandomNumberEngine, __w,
                _UIntType> & __x )

```

Inserts the current state of a `independent_bits_engine` random number generator engine `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `independent_bits_engine` random number generator engine.

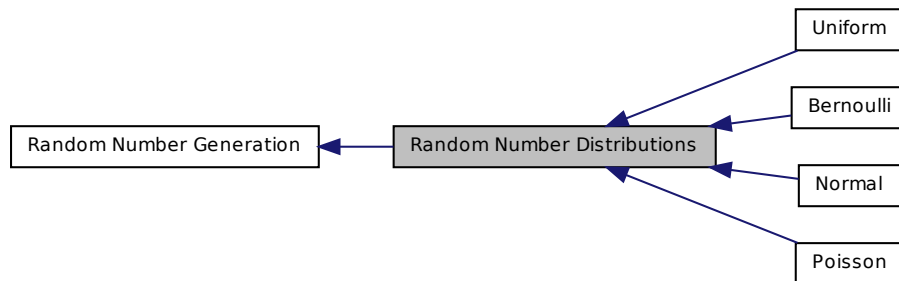
Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1222 of file `random.h`.

2.52 Random Number Distributions

Collaboration diagram for Random Number Distributions:



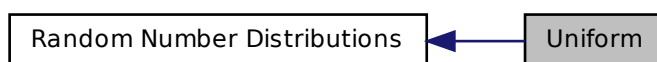
Modules

- [Uniform](#)

- [Normal](#)
- [Bernoulli](#)
- [Poisson](#)

2.53 Uniform

Collaboration diagram for Uniform:



Classes

- class [std::uniform_int_distribution< _IntType >](#)
Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.
- class [std::uniform_real_distribution< _RealType >](#)
Uniform continuous distribution for random numbers.

Functions

- `template<typename _IntType >`
`bool std::operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`

- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _IntType >
bool std::operator==(const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _IntType >
bool std::operator==(const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`

2.53.1 Function Documentation

2.53.1.1 `template<typename _IntType > bool std::operator!=(const
std::uniform_int_distribution< _IntType > & __d1, const
std::uniform_int_distribution< _IntType > & __d2) [inline]`

Return true if two uniform integer distributions have different parameters.

Definition at line 1761 of file random.h.

2.53.1.2 `template<typename _IntType > bool std::operator!=(const
std::uniform_real_distribution< _IntType > & __d1, const
std::uniform_real_distribution< _IntType > & __d2) [inline]`

Return true if two uniform real distributions have different parameters.

Definition at line 1942 of file random.h.

2.53.1.3 `template<typename _RealType, typename _CharT, typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::uniform_real_distribution< _RealType > & __x)`

Inserts a uniform_real_distribution random number distribution __x into the output stream __os.

Parameters

`__os` An output stream.

`__x` A `uniform_real_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 937 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.53.1.4 `template<typename _IntType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::uniform_int_distribution< _IntType > & __x)`

Inserts a `uniform_int_distribution` random number distribution `__x` into the output stream `os`.

Parameters

`__os` An output stream.

`__x` A `uniform_int_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 895 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

2.53.1.5 `template<typename _IntType > bool std::operator==(const
std::uniform_real_distribution< _IntType > & __d1, const
std::uniform_real_distribution< _IntType > & __d2) [inline]`

Return true if two uniform real distributions have the same parameters.

Definition at line 1932 of file `random.h`.

References `std::uniform_real_distribution< _RealType >::param()`.

2.53.1.6 `template<typename _IntType > bool std::operator==(const
std::uniform_int_distribution< _IntType > & __d1, const
std::uniform_int_distribution< _IntType > & __d2) [inline]`

Return true if two uniform integer distributions have the same parameters.

Definition at line 1751 of file random.h.

References `std::uniform_int_distribution< _IntType >::param()`.

2.53.1.7 `template<typename _RealType , typename _CharT ,
typename _Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::uniform_real_distribution< _RealType > & __x)`

Extracts a `uniform_real_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `uniform_real_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 961 of file random.tcc.

References `std::ios_base::flags()`, `std::uniform_real_distribution< _RealType >::param()`, and `std::skipws()`.

2.53.1.8 `template<typename _IntType , typename _CharT , typename
_Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::uniform_int_distribution< _IntType > & __x)`

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `uniform_int_distribution` random number generator engine.

Returns

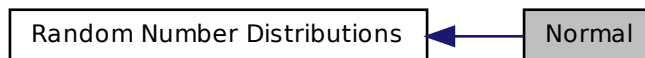
The input stream with `__x` extracted or in an error state.

Definition at line 916 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::uniform_int_distribution< _IntType >::param()`, and `std::skipws()`.

2.54 Normal

Collaboration diagram for Normal:



Classes

- class `std::cauchy_distribution<_RealType>`
A *cauchy_distribution* random number distribution.
- class `std::chi_squared_distribution<_RealType>`
A *chi_squared_distribution* random number distribution.
- class `std::fisher_f_distribution<_RealType>`
A *fisher_f_distribution* random number distribution.
- class `std::gamma_distribution<_RealType>`
A *gamma* continuous distribution for random numbers.
- class `std::lognormal_distribution<_RealType>`
A *lognormal_distribution* random number distribution.
- class `std::normal_distribution<_RealType>`
A *normal* continuous distribution for random numbers.
- class `std::student_t_distribution<_RealType>`
A *student_t_distribution* random number distribution.

Functions

- template<typename `_RealType`>
bool `std::operator!=` (const `std::normal_distribution<_RealType>` &__d1,
const `std::normal_distribution<_RealType>` &__d2)

- `template<typename _RealType >`
`bool std::operator!= (const std::lognormal_distribution< _RealType > &__d1,`
`const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1,`
`const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1,`
`const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1,`
`const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1,`
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::gamma_distribution< _RealType > &__d1,`
`const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_-`
`ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType >`
`&)`
- `template<typename _RealType >`
`bool std::operator== (const std::cauchy_distribution< _RealType > &__d1,`
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

2.54.1 Function Documentation

2.54.1.1 `template<typename _RealType > bool std::operator!= (const`
`std::normal_distribution< _RealType > & __d1, const`
`std::normal_distribution< _RealType > & __d2) [inline]`

Return true if two normal distributions are different.

Definition at line 2162 of file random.h.

2.54.1.2 `template<typename _RealType > bool std::operator!= (const`
`std::lognormal_distribution< _RealType > & __d1, const`
`std::lognormal_distribution< _RealType > & __d2) [inline]`

Return true if two lognormal distributions are different.

Definition at line 2338 of file random.h.

2.54.1.3 `template<typename _RealType > bool std::operator!= (const
std::chi_squared_distribution< _RealType > & __d1, const
std::chi_squared_distribution< _RealType > & __d2) [inline]`

Return true if two Chi-squared distributions are different.

Definition at line 2695 of file random.h.

2.54.1.4 `template<typename _RealType > bool std::operator!= (const
std::fisher_f_distribution< _RealType > & __d1, const
std::fisher_f_distribution< _RealType > & __d2) [inline]`

Return true if two Fisher f distributions are diferent.

Definition at line 3050 of file random.h.

2.54.1.5 `template<typename _RealType > bool std::operator!= (const
std::student_t_distribution< _RealType > & __d1, const
std::student_t_distribution< _RealType > & __d2) [inline]`

Return true if two Student t distributions are different.

Definition at line 3224 of file random.h.

2.54.1.6 `template<typename _RealType > bool std::operator!= (const
std::cauchy_distribution< _RealType > & __d1, const
std::cauchy_distribution< _RealType > & __d2) [inline]`

Return true if two Cauchy distributions have different parameters.

Definition at line 2834 of file random.h.

2.54.1.7 `template<typename _RealType > bool std::operator!= (const
std::gamma_distribution< _RealType > & __d1, const
std::gamma_distribution< _RealType > & __d2) [inline]`

Return true if two gamma distributions are different.

Definition at line 2530 of file random.h.

2.54.1.8 `template<typename _RealType, typename _CharT, typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::cauchy_distribution< _RealType > & __x)`

Inserts a `cauchy_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `cauchy_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1859 of file `random.tcc`.

References `std::left()`, and `std::scientific()`.

2.54.1.9 `template<typename _RealType > bool std::operator==(
const std::cauchy_distribution< _RealType > & __d1, const
std::cauchy_distribution< _RealType > & __d2) [inline]`

Return true if two Cauchy distributions have the same parameters.

Definition at line 2824 of file `random.h`.

References `std::cauchy_distribution< _RealType >::param()`.

2.54.1.10 `template<typename _RealType, typename _CharT,
typename _Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::cauchy_distribution< _RealType > & __x)`

Extracts a `cauchy_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `cauchy_distribution` random number generator engine.

Returns

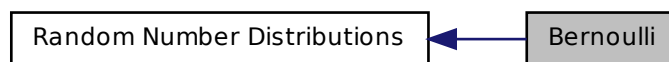
The input stream with `__x` extracted or in an error state.

Definition at line 1883 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::cauchy_distribution< _RealType >::param()`, and `std::skipws()`.

2.55 Bernoulli

Collaboration diagram for Bernoulli:



Classes

- class [std::bernoulli_distribution](#)
A Bernoulli random number distribution.
- class [std::binomial_distribution< _IntType >](#)
A discrete binomial random number distribution.
- class [std::geometric_distribution< _IntType >](#)
A discrete geometric random number distribution.
- class [std::negative_binomial_distribution< _IntType >](#)
A [negative_binomial_distribution](#) random number distribution.

Functions

- `bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`

- `template<typename _IntType >`
`bool std::operator!= (const std::negative_binomial_distribution< _IntType >`
`&__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::geometric_distribution< _IntType > &__d1,`
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType >`
`&)`
- `template<typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `bool std::operator== (const std::bernoulli_distribution &__d1, const`
`std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`
`bool std::operator== (const std::geometric_distribution< _IntType > &__d1,`
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`
`_CharT, _Traits > &__is, std::bernoulli_distribution &__x)`

2.55.1 Function Documentation

2.55.1.1 `bool std::operator!= (const std::bernoulli_distribution & __d1, const std::bernoulli_distribution & __d2) [inline]`

Return true if two Bernoulli distributions have different parameters.

Definition at line 3372 of file random.h.

2.55.1.2 `template<typename _IntType > bool std::operator!= (const std::binomial_distribution< _IntType > & __d1, const std::binomial_distribution< _IntType > & __d2) [inline]`

Return true if two binomial distributions are different.

Definition at line 3609 of file random.h.

2.55.1.3 `template<typename _IntType > bool std::operator!= (const
std::negative_binomial_distribution< _IntType > & __d1, const
std::negative_binomial_distribution< _IntType > & __d2)
[inline]`

Return true if two negative binomial distributions are different.

Definition at line 3955 of file random.h.

2.55.1.4 `template<typename _IntType > bool std::operator!= (const
std::geometric_distribution< _IntType > & __d1, const
std::geometric_distribution< _IntType > & __d2) [inline]`

Return true if two geometric distributions have different parameters.

Definition at line 3751 of file random.h.

2.55.1.5 `template<typename _IntType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::geometric_distribution< _IntType > & __x)`

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `geometric_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1032 of file random.tcc.

References `std::left()`, and `std::scientific()`.

2.55.1.6 `template<typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::bernoulli_distribution & __x)`

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.
`__x` A `bernoulli_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 982 of file `random.tcc`.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.55.1.7 `bool std::operator==(const std::bernoulli_distribution & __d1,
const std::bernoulli_distribution & __d2) [inline]`

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3363 of file `random.h`.

References `std::bernoulli_distribution::param()`.

2.55.1.8 `template<typename _IntType > bool std::operator==(const
std::geometric_distribution< _IntType > & __d1, const
std::geometric_distribution< _IntType > & __d2) [inline]`

Return true if two geometric distributions have the same parameters.

Definition at line 3741 of file `random.h`.

References `std::geometric_distribution< _IntType >::param()`.

2.55.1.9 `template<typename _IntType , typename _CharT , typename
_Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::geometric_distribution< _IntType > & __x)`

Extracts a `geometric_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.
`__x` A `geometric_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1056 of file random.tcc.

References `std::ios_base::flags()`, `std::geometric_distribution< _IntType >::param()`, and `std::skipws()`.

2.55.1.10 `template<typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& std::operator>>
(std::basic_istream< _CharT, _Traits > & __is,
std::bernoulli_distribution & __x)`

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `bernoulli_distribution` random number generator engine.

Returns

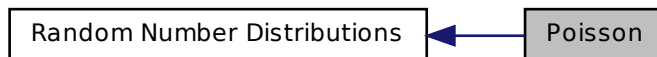
The input stream with `__x` extracted or in an error state.

Definition at line 3402 of file random.h.

References `std::bernoulli_distribution::param()`.

2.56 Poisson

Collaboration diagram for Poisson:



Classes

- class `std::discrete_distribution< _IntType >`

A *discrete_distribution* random number distribution.

- class `std::exponential_distribution<_RealType>`
A *exponential continuous distribution* for random numbers.
- class `std::extreme_value_distribution<_RealType>`
A *extreme_value_distribution* random number distribution.
- class `std::piecewise_constant_distribution<_RealType>`
A *piecewise_constant_distribution* random number distribution.
- class `std::piecewise_linear_distribution<_RealType>`
A *piecewise_linear_distribution* random number distribution.
- class `std::poisson_distribution<_IntType>`
A *discrete Poisson* random number distribution.
- class `std::weibull_distribution<_RealType>`
A *weibull_distribution* random number distribution.

Functions

- template<typename _IntType>
bool `std::operator!=` (const `std::poisson_distribution<_IntType>` &__d1, const `std::poisson_distribution<_IntType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::extreme_value_distribution<_RealType>` &__d1, const `std::extreme_value_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::piecewise_constant_distribution<_RealType>` &__d1, const `std::piecewise_constant_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::piecewise_linear_distribution<_RealType>` &__d1, const `std::piecewise_linear_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::exponential_distribution<_RealType>` &__d1, const `std::exponential_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::weibull_distribution<_RealType>` &__d1, const `std::weibull_distribution<_RealType>` &__d2)
- template<typename _IntType>
bool `std::operator!=` (const `std::discrete_distribution<_IntType>` &__d1, const `std::discrete_distribution<_IntType>` &__d2)

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _RealType >`
`bool std::operator== (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator== (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator== (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`

2.56.1 Function Documentation

2.56.1.1 `template<typename _IntType > bool std::operator!=(const
std::poisson_distribution< _IntType > & __d1, const
std::poisson_distribution< _IntType > & __d2) [inline]`

Return true if two Poisson distributions are different.

Definition at line 4143 of file random.h.

2.56.1.2 `template<typename _RealType > bool std::operator!=(const
std::extreme_value_distribution< _RealType > & __d1, const
std::extreme_value_distribution< _RealType > & __d2) [inline]`

Return true if two extreme value distributions have different parameters.

Definition at line 4643 of file random.h.

2.56.1.3 `template<typename _RealType > bool std::operator!=(const
std::piecewise_constant_distribution< _RealType > & __d1, const
std::piecewise_constant_distribution< _RealType > & __d2)
[inline]`

Return true if two piecewise constant distributions have different parameters.

Definition at line 5073 of file random.h.

2.56.1.4 `template<typename _RealType > bool std::operator!=(const
std::piecewise_linear_distribution< _RealType > & __d1, const
std::piecewise_linear_distribution< _RealType > & __d2)
[inline]`

Return true if two piecewise linear distributions have different parameters.

Definition at line 5284 of file random.h.

2.56.1.5 `template<typename _RealType > bool std::operator!=(const
std::exponential_distribution< _RealType > & __d1, const
std::exponential_distribution< _RealType > & __d2) [inline]`

Return true if two exponential distributions have different parameters.

Definition at line 4293 of file random.h.

2.56.1.6 `template<typename _RealType > bool std::operator!=(
const std::weibull_distribution< _RealType > & __d1, const
std::weibull_distribution< _RealType > & __d2) [inline]`

Return true if two Weibull distributions have different parameters.

Definition at line 4468 of file random.h.

2.56.1.7 `template<typename _IntType > bool std::operator!=(const
std::discrete_distribution< _IntType > & __d1, const
std::discrete_distribution< _IntType > & __d2) [inline]`

Return true if two discrete distributions have different parameters.

Definition at line 4865 of file random.h.

2.56.1.8 `template<typename _RealType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT , _Traits > & std::operator<<
(std::basic_ostream< _CharT , _Traits > & __os, const
std::weibull_distribution< _RealType > & __x)`

Inserts a weibull_distribution random number distribution __x into the output stream __os.

Parameters

`__os` An output stream.

`__x` A weibull_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

Definition at line 2112 of file random.tcc.

References `std::left()`, and `std::scientific()`.

2.56.1.9 `template<typename _RealType , typename _CharT , typename _Traits
> std::basic_ostream< _CharT , _Traits > & std::operator<<
(std::basic_ostream< _CharT , _Traits > & __os, const
std::exponential_distribution< _RealType > & __x)`

Inserts a exponential_distribution random number distribution __x into the output stream __os.

Parameters

`__os` An output stream.

`__x` A exponential_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1592 of file random.tcc.

References `std::ios_base::flags()`, `std::left()`, and `std::scientific()`.

2.56.1.10 `template<typename _RealType , typename _CharT , typename
_Traits > std::basic_ostream< _CharT, _Traits > & std::operator<<
(std::basic_ostream< _CharT, _Traits > & __os, const
std::extreme_value_distribution< _RealType > & __x)`

Inserts a extreme_value_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A extreme_value_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2169 of file random.tcc.

References `std::left()`, and `std::scientific()`.

2.56.1.11 `template<typename _RealType > bool std::operator==(const
std::exponential_distribution< _RealType > & __d1, const
std::exponential_distribution< _RealType > & __d2) [inline]`

Return true if two exponential distributions have the same parameters.

Definition at line 4283 of file random.h.

References `std::exponential_distribution< _RealType >::param()`.

2.56.1.12 `template<typename _RealType > bool std::operator==(const
std::piecewise_constant_distribution< _RealType > & __d1, const
std::piecewise_constant_distribution< _RealType > & __d2)
[inline]`

Return true if two piecewise constant distributions have the same parameters.

Definition at line 5063 of file random.h.

References `std::piecewise_constant_distribution< _RealType >::param()`.

2.56.1.13 `template<typename _RealType > bool std::operator==(const
std::piecewise_linear_distribution< _RealType > & __d1, const
std::piecewise_linear_distribution< _RealType > & __d2)
[inline]`

Return true if two piecewise linear distributions have the same parameters.

Definition at line 5274 of file random.h.

References `std::piecewise_linear_distribution< _RealType >::param()`.

2.56.1.14 `template<typename _IntType > bool std::operator==(const
std::discrete_distribution< _IntType > & __d1, const
std::discrete_distribution< _IntType > & __d2) [inline]`

Return true if two discrete distributions have the same parameters.

Definition at line 4855 of file random.h.

References `std::discrete_distribution< _IntType >::param()`.

2.56.1.15 `template<typename _RealType > bool std::operator==(const
std::extreme_value_distribution< _RealType > & __d1, const
std::extreme_value_distribution< _RealType > & __d2)
[inline]`

Return true if two extreme value distributions have the same parameters.

Definition at line 4633 of file random.h.

References `std::extreme_value_distribution< _RealType >::param()`.

2.56.1.16 `template<typename _RealType > bool std::operator==(const std::weibull_distribution< _RealType > & __d1, const std::weibull_distribution< _RealType > & __d2) [inline]`

Return true if two Weibull distributions have the same parameters.

Definition at line 4458 of file random.h.

References `std::weibull_distribution< _RealType >::param()`.

2.56.1.17 `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::exponential_distribution< _RealType > & __x)`

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `exponential_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1615 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::exponential_distribution< _RealType >::param()`, and `std::skipws()`.

2.56.1.18 `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::weibull_distribution< _RealType > & __x)`

Extracts a `weibull_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `weibull_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2136 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::weibull_distribution< _RealType >::param()`, and `std::skipws()`.

2.56.1.19 `template<typename _RealType , typename _CharT ,
typename _Traits > std::basic_istream< _CharT, _Traits > &
std::operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::extreme_value_distribution< _RealType > & __x)`

Extracts a `extreme_value_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `extreme_value_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2193 of file random.tcc.

References `std::dec()`, `std::ios_base::flags()`, `std::extreme_value_distribution< _RealType >::param()`, and `std::skipws()`.

2.57 Random Number Utilities

Collaboration diagram for Random Number Utilities:



Classes

- class [std::seed_seq](#)

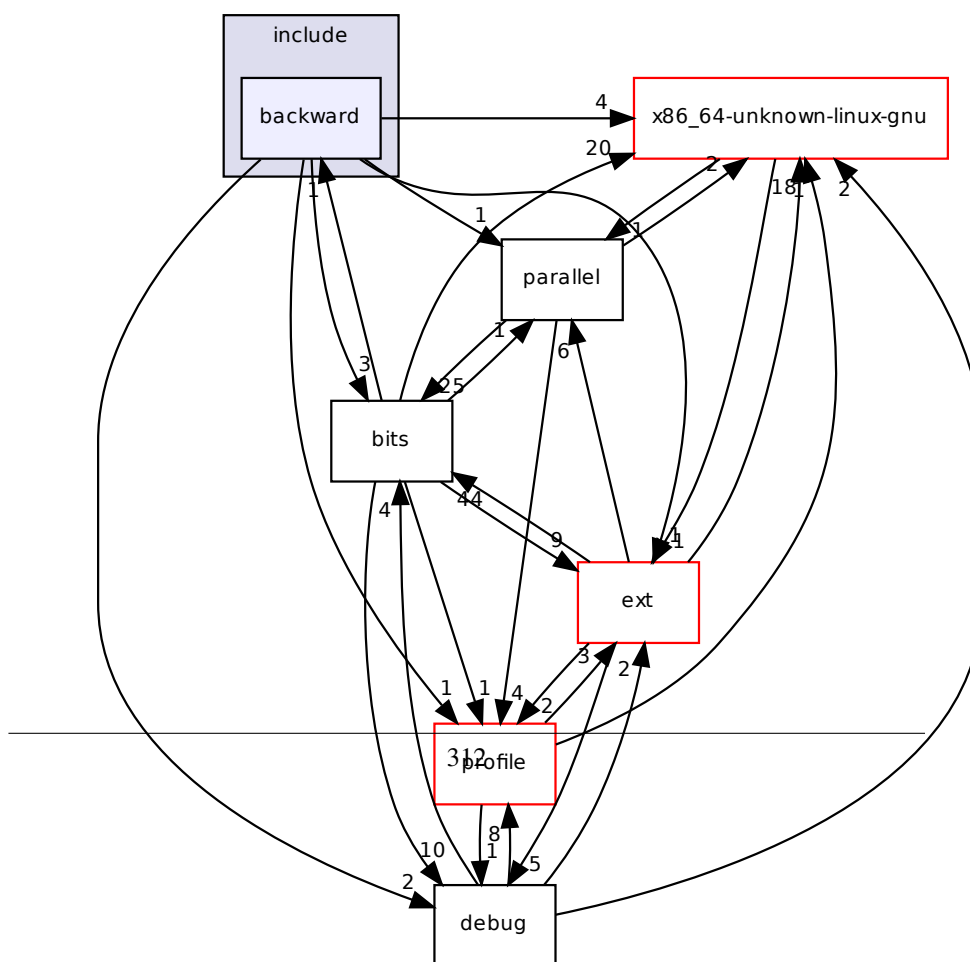
The [seed_seq](#) class generates sequences of seeds for random number generators.

Chapter 3

Directory Documentation

3.1 include/backward/ Directory Reference

Directory dependency graph for include/backward/:

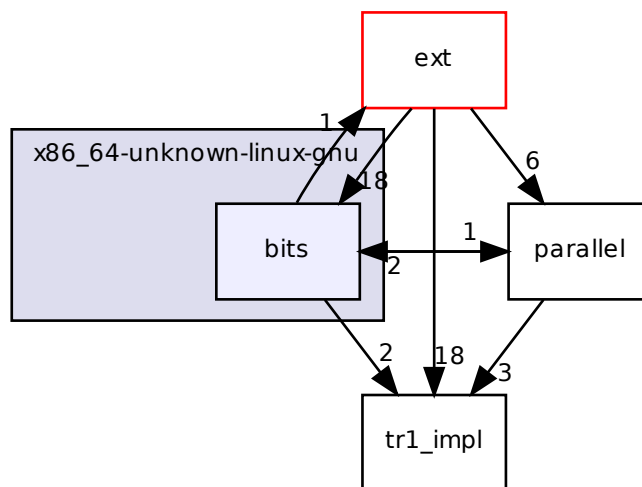


Files

- file [auto_ptr.h](#)
- file **backward_warning.h**
- file [binders.h](#)
- file [hash_fun.h](#)
- file [hash_map](#)
- file [hash_set](#)
- file [backward/hashtable.h](#)
- file **strstream**

3.2 include/x86_64-unknown-linux-gnu/bits/ Directory Reference

Directory dependency graph for include/x86_64-unknown-linux-gnu/bits/:



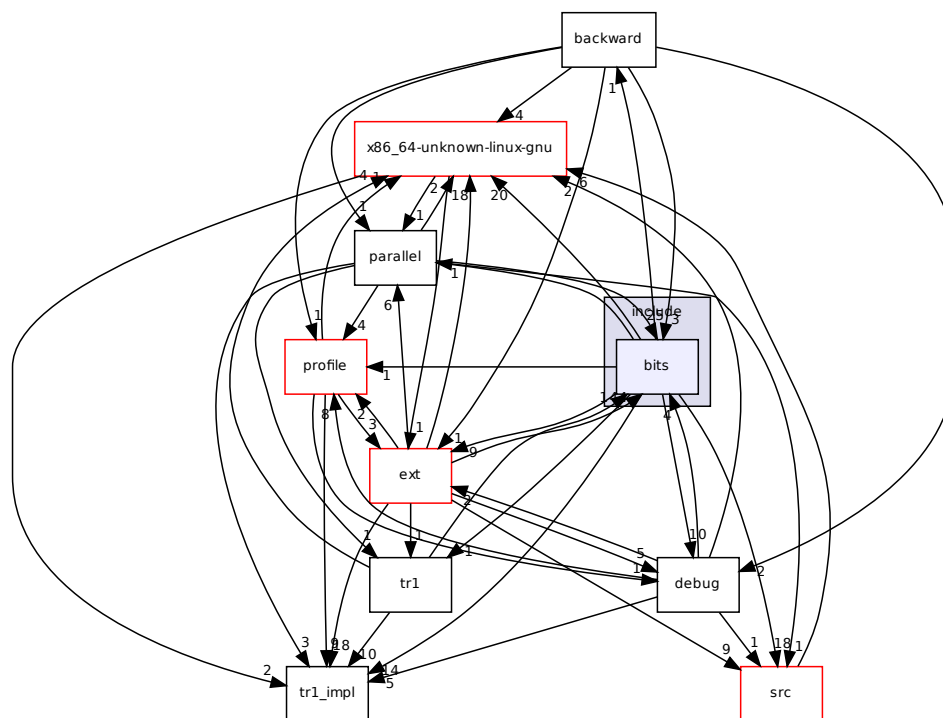
Files

- file [atomic_word.h](#)

- file [basic_file.h](#)
- file [c++allocator.h](#)
- file [c++config.h](#)
- file [c++io.h](#)
- file [c++locale.h](#)
- file [c++locale_internal.h](#)
- file [x86_64-unknown-linux-gnu/bits/compatibility.h](#)
- file [cpu_defines.h](#)
- file [ctype_base.h](#)
- file [ctype_inline.h](#)
- file [ctype_noninline.h](#)
- file [cxxabi_tweaks.h](#)
- file [error_constants.h](#)
- file **gthr-default.h**
- file **gthr-posix.h**
- file **gthr-single.h**
- file **gthr-tpf.h**
- file **gthr.h**
- file [messages_members.h](#)
- file [os_defines.h](#)
- file [time_members.h](#)

3.3 include/bits/ Directory Reference

Directory dependency graph for include/bits/:



Files

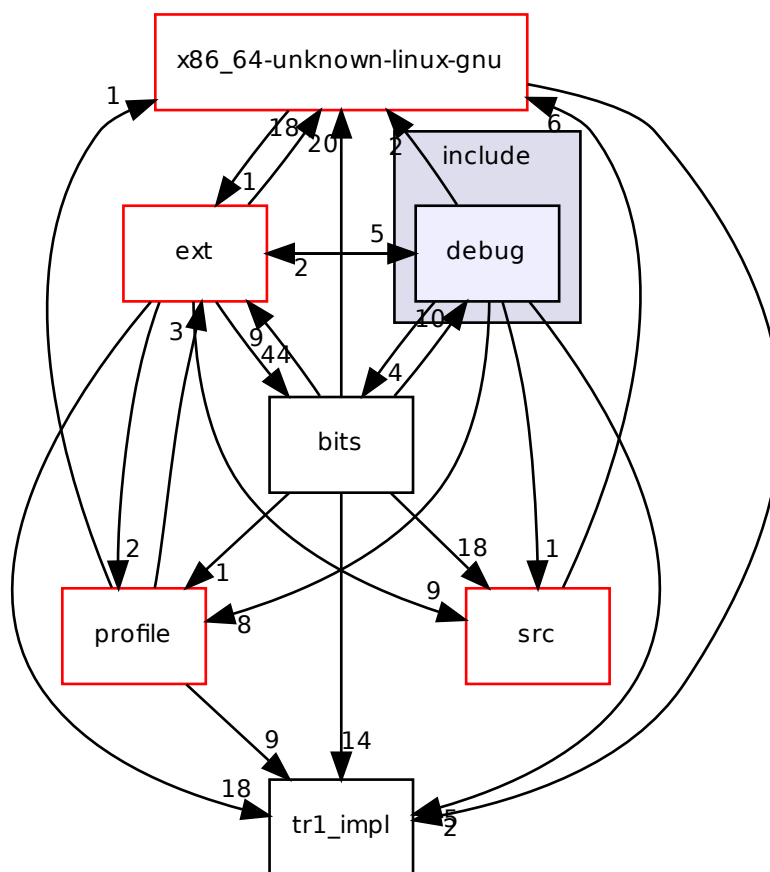
- file [bits/algorithmfwd.h](#)
- file [allocator.h](#)
- file [atomic_0.h](#)
- file [atomic_2.h](#)
- file [atomic_base.h](#)
- file [atomicfwd_c.h](#)
- file [atomicfwd_cxx.h](#)
- file [basic_ios.h](#)
- file [basic_ios.tcc](#)
- file [basic_string.h](#)

- file [basic_string.tcc](#)
- file [boost_concept_check.h](#)
- file [c++0x_warning.h](#)
- file [char_traits.h](#)
- file [codecvt.h](#)
- file [concept_check.h](#)
- file [cpp_type_traits.h](#)
- file [deque.tcc](#)
- file [forward_list.h](#)
- file [forward_list.tcc](#)
- file [fstream.tcc](#)
- file [functexcept.h](#)
- file [functional_hash.h](#)
- file [gslice.h](#)
- file [gslice_array.h](#)
- file [bits/hashtable.h](#)
- file [hashtable_policy.h](#)
- file [indirect_array.h](#)
- file [ios_base.h](#)
- file [istream.tcc](#)
- file [list.tcc](#)
- file [locale_classes.h](#)
- file [locale_classes.tcc](#)
- file [locale_facets.h](#)
- file [locale_facets.tcc](#)
- file [locale_facets_nonio.h](#)
- file [locale_facets_nonio.tcc](#)
- file [localefwd.h](#)
- file [mask_array.h](#)
- file [move.h](#)
- file [ostream.tcc](#)
- file [ostream_insert.h](#)
- file [postypes.h](#)
- file [random.h](#)
- file [random.tcc](#)
- file [range_access.h](#)
- file [regex.h](#)
- file [regex_compiler.h](#)
- file [regex_constants.h](#)
- file [regex_cursor.h](#)
- file [regex_error.h](#)
- file [regex_grep_matcher.h](#)
- file [regex_grep_matcher.tcc](#)

- file [regex_nfa.h](#)
- file [regex_nfa.tcc](#)
- file [shared_ptr.h](#)
- file [shared_ptr_base.h](#)
- file [slice_array.h](#)
- file [sstream.tcc](#)
- file [stl_algo.h](#)
- file [stl_algobase.h](#)
- file [stl_bvector.h](#)
- file [stl_construct.h](#)
- file [stl_deque.h](#)
- file [stl_function.h](#)
- file [stl_heap.h](#)
- file [stl_iterator.h](#)
- file [stl_iterator_base_funcs.h](#)
- file [stl_iterator_base_types.h](#)
- file [stl_list.h](#)
- file [stl_map.h](#)
- file [stl_multimap.h](#)
- file [stl_multiset.h](#)
- file [stl_numeric.h](#)
- file [stl_pair.h](#)
- file [stl_queue.h](#)
- file [stl_raw_storage_iter.h](#)
- file [stl_relops.h](#)
- file [stl_set.h](#)
- file [stl_stack.h](#)
- file [stl_tempbuf.h](#)
- file [stl_tree.h](#)
- file [stl_uninitialized.h](#)
- file [stl_vector.h](#)
- file [stream_iterator.h](#)
- file [streambuf.tcc](#)
- file [streambuf_iterator.h](#)
- file [stringfwd.h](#)
- file [unique_ptr.h](#)
- file [unordered_map.h](#)
- file [unordered_set.h](#)
- file [valarray_after.h](#)
- file [valarray_array.h](#)
- file [valarray_array.tcc](#)
- file [valarray_before.h](#)
- file [vector.tcc](#)

3.4 include/debug/ Directory Reference

Directory dependency graph for include/debug/:



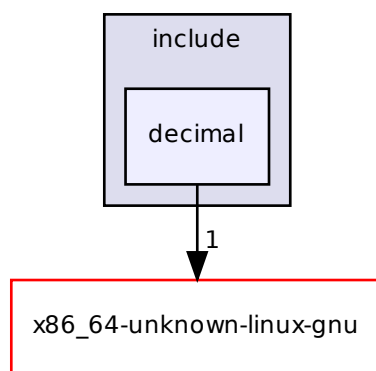
Files

- file [debug/bitset](#)
- file [debug.h](#)
- file [debug/deque](#)

- file [formatter.h](#)
- file [functions.h](#)
- file [debug/list](#)
- file [macros.h](#)
- file [debug/map](#)
- file [debug/map.h](#)
- file [debug/multimap.h](#)
- file [debug/multiset.h](#)
- file [safe_base.h](#)
- file [safe_iterator.h](#)
- file [safe_iterator.tcc](#)
- file [safe_sequence.h](#)
- file [debug/set](#)
- file [debug/set.h](#)
- file [debug/string](#)
- file [debug/unordered_map](#)
- file [debug/unordered_set](#)
- file [debug/vector](#)

3.5 include/decimal/ Directory Reference

Directory dependency graph for include/decimal/:

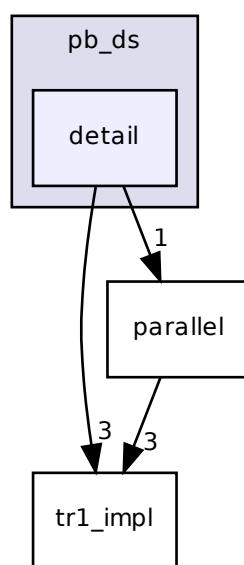


Files

- file [decimal](#)

3.6 include/ext/pb_ds/detail/ Directory Reference

Directory dependency graph for include/ext/pb_ds/detail/:



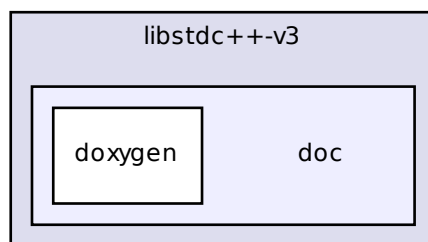
Files

- file [basic_types.hpp](#)
- file [cond_dealtor.hpp](#)
- file [constructors_destructor_fn_imps.hpp](#)
- file [container_base_dispatch.hpp](#)
- file [debug_map_base.hpp](#)
- file [priority_queue_base_dispatch.hpp](#)

- file [standard_policies.hpp](#)
- file [tree_trace_base.hpp](#)
- file [type_utils.hpp](#)
- file [types_traits.hpp](#)

3.7 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/:

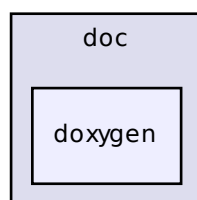


Directories

- directory [doxygen](#)

3.8 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/doxygen/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/doxygen/:

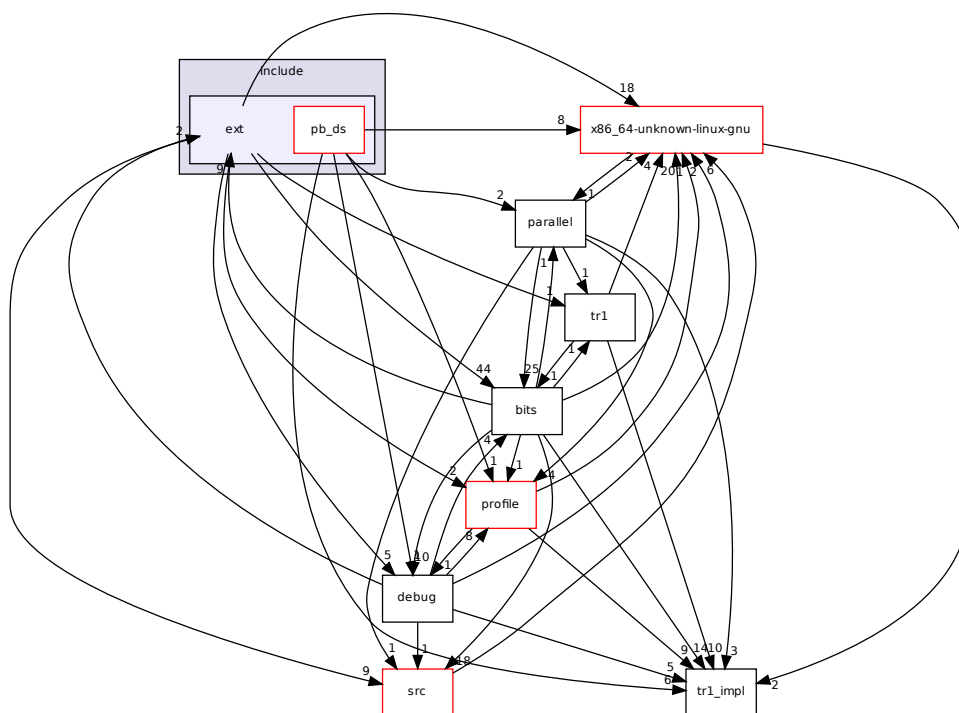


Files

- file **doxygroups.cc**

3.9 include/ext/ Directory Reference

Directory dependency graph for include/ext/:



Directories

- directory [pb_ds](#)

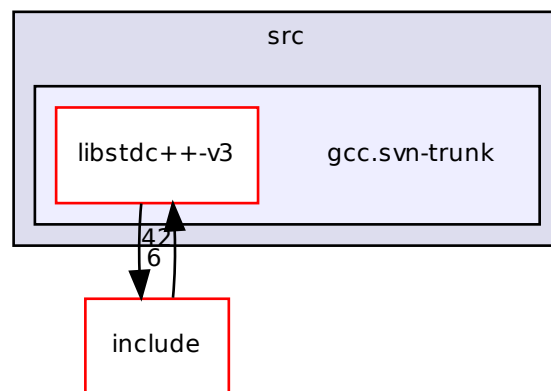
Files

- file [ext/algorithm](#)
- file [array_allocator.h](#)
- file [atomicity.h](#)
- file [bitmap_allocator.h](#)
- file [cast.h](#)
- file [codecvt_specializations.h](#)

- file [concurrency.h](#)
- file [debug_allocator.h](#)
- file [enc_filebuf.h](#)
- file [extptr_allocator.h](#)
- file [ext/functional](#)
- file [ext/iterator](#)
- file [malloc_allocator.h](#)
- file [ext/memory](#)
- file [mt_allocator.h](#)
- file [new_allocator.h](#)
- file [ext/numeric](#)
- file [numeric_traits.h](#)
- file [pod_char_traits.h](#)
- file [pointer.h](#)
- file [pool_allocator.h](#)
- file [rb_tree](#)
- file [rc_string_base.h](#)
- file [rope](#)
- file [ropeimpl.h](#)
- file [slist](#)
- file [sso_string_base.h](#)
- file [stdio_filebuf.h](#)
- file [stdio_sync_filebuf.h](#)
- file **[string_conversions.h](#)**
- file [throw_allocator.h](#)
- file [type_traits.h](#)
- file [typelist.h](#)
- file [vstring.h](#)
- file [vstring.tcc](#)
- file [vstring_fwd.h](#)
- file [vstring_util.h](#)

3.10 /mnt/share/src/gcc.svn-trunk/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/:

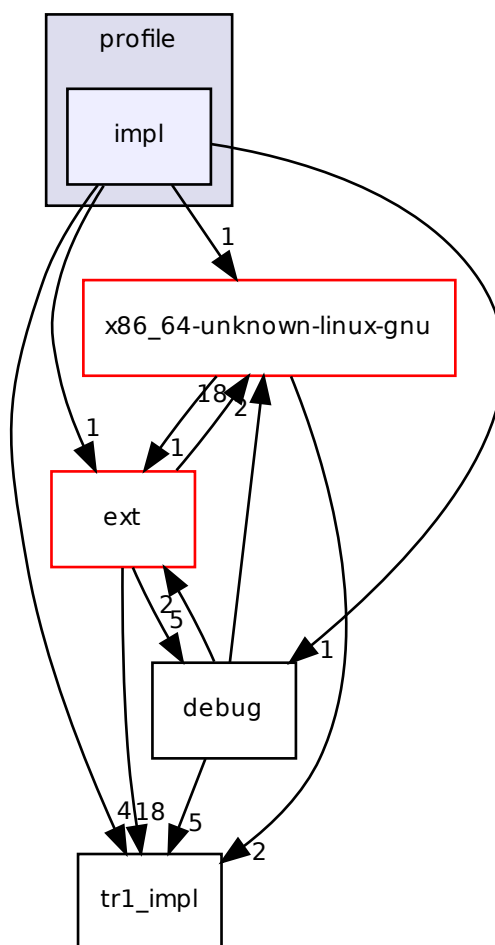


Directories

- directory [libstdc++-v3](#)

3.11 include/profile/impl/ Directory Reference

Directory dependency graph for include/profile/impl/:

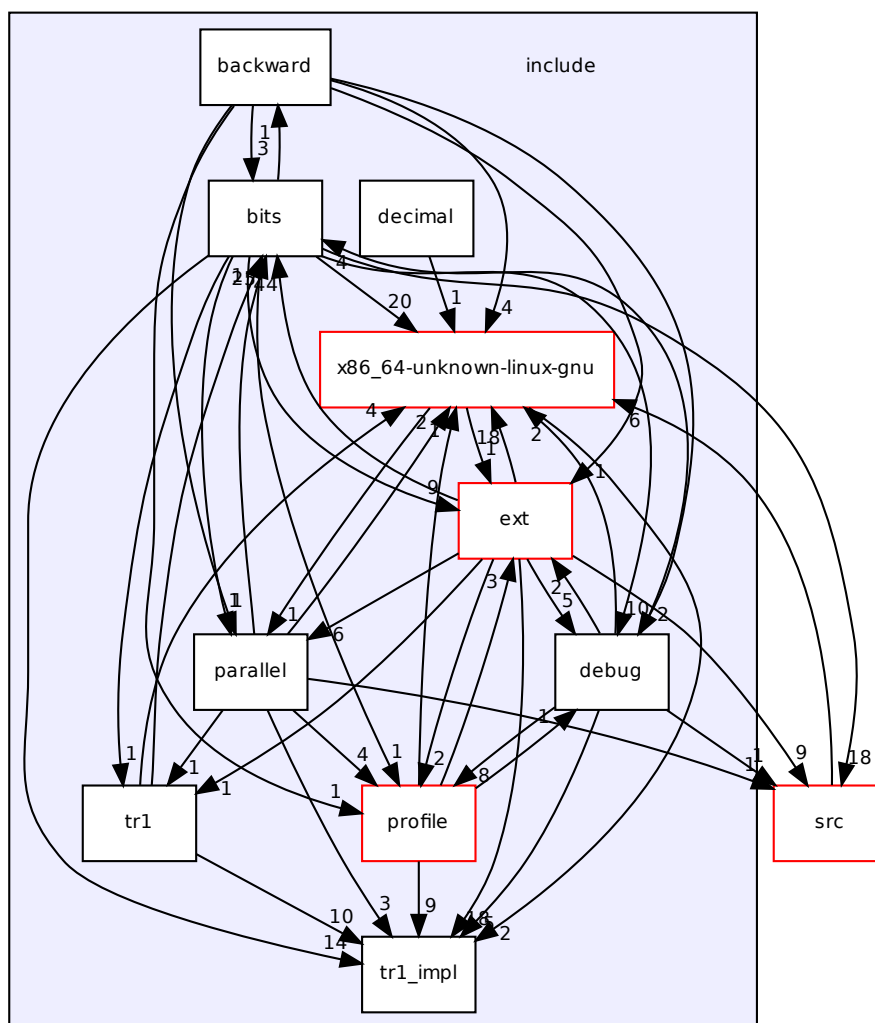


Files

- file [profiler.h](#)
- file [profiler_algos.h](#)
- file **profiler_container_size.h**
- file **profiler_hash_func.h**
- file [profiler_hashtable_size.h](#)
- file [profiler_list_to_slist.h](#)
- file [profiler_list_to_vector.h](#)
- file [profiler_map_to_unordered_map.h](#)
- file [profiler_node.h](#)
- file [profiler_state.h](#)
- file [profiler_trace.h](#)
- file [profiler_vector_size.h](#)
- file [profiler_vector_to_list.h](#)

3.12 include/ Directory Reference

Directory dependency graph for include/:



Directories

- directory [backward](#)
- directory [bits](#)
- directory [debug](#)
- directory [decimal](#)
- directory [ext](#)
- directory [parallel](#)
- directory [profile](#)
- directory [tr1](#)
- directory [tr1_impl](#)
- directory [x86_64-unknown-linux-gnu](#)

Files

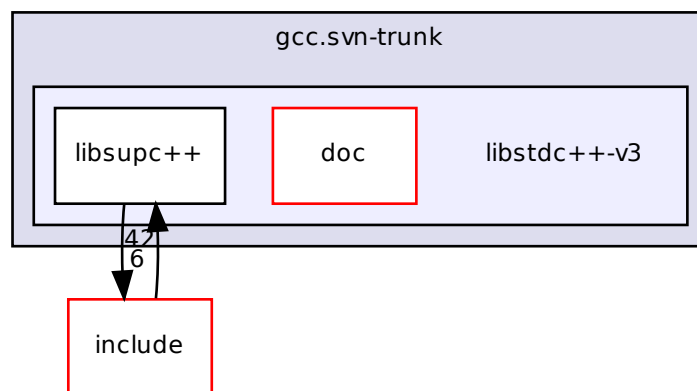
- file [algorithm](#)
- file [array](#)
- file [atomic](#)
- file [bitset](#)
- file [cassert](#)
- file [ccomplex](#)
- file [cctype](#)
- file [cerrno](#)
- file [cfenv](#)
- file [cfloat](#)
- file [chrono](#)
- file [cinttypes](#)
- file [ciso646](#)
- file [climits](#)
- file [clocale](#)
- file [cmath](#)
- file [complex](#)
- file [complex.h](#)
- file [condition_variable](#)
- file [csetjmp](#)
- file [csignal](#)
- file [cstdarg](#)
- file [cstdbool](#)
- file [cstddef](#)
- file [cstdint](#)
- file [cstdio](#)
- file [cstdlib](#)

- file [cstring](#)
- file [ctgmath](#)
- file [ctime](#)
- file [cwchar](#)
- file [cwctype](#)
- file [deque](#)
- file [fenv.h](#)
- file [fstream](#)
- file [functional](#)
- file [future](#)
- file **gstdint.h**
- file [iomanip](#)
- file [ios](#)
- file [iosfwd](#)
- file [iostream](#)
- file [istream](#)
- file [iterator](#)
- file [limits](#)
- file [list](#)
- file [locale](#)
- file [map](#)
- file [memory](#)
- file [mutex](#)
- file [numeric](#)
- file [ostream](#)
- file [queue](#)
- file [random](#)
- file [ratio](#)
- file [regex](#)
- file [set](#)
- file [sstream](#)
- file [stack](#)
- file [stdatomic.h](#)
- file [stdexcept](#)
- file [streambuf](#)
- file [string](#)
- file [system_error](#)
- file [tgmath.h](#)
- file [thread](#)
- file [tuple](#)
- file [type_traits](#)
- file [unordered_map](#)
- file [unordered_set](#)

- file [utility](#)
- file [valarray](#)
- file [vector](#)

3.13 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/libstdc++-v3/:

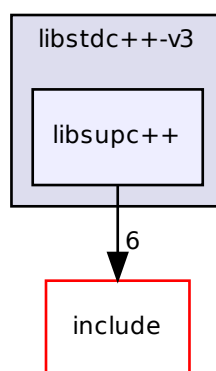


Directories

- directory [doc](#)
- directory [libsupc++](#)

3.14 /mnt/share/src/gcc.svn-trunk/libstdc++-v3/libsupc++/ Directory Reference

Directory dependency graph for /mnt/share/src/gcc.svn-trunk/libstdc++-v3/libsupc++/:

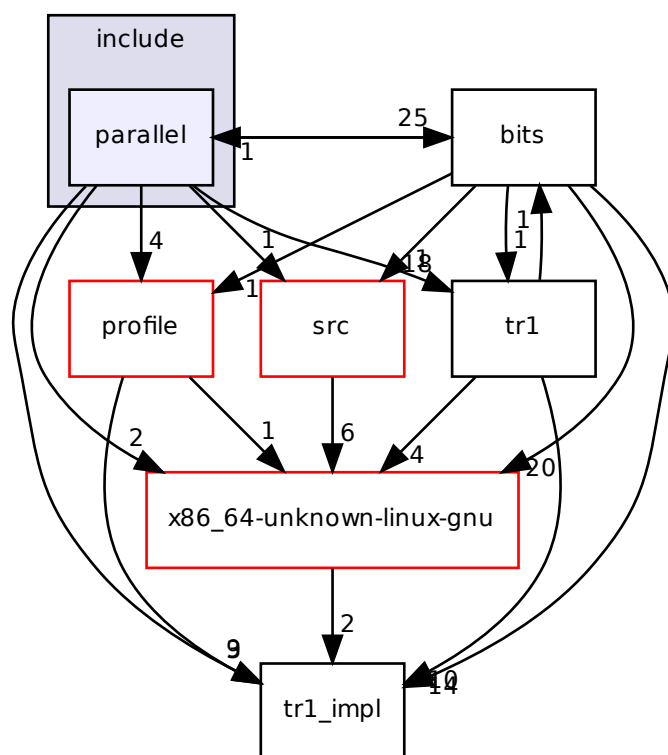


Files

- file [cxxabi-forced.h](#)
- file [cxxabi.h](#)
- file [exception](#)
- file [exception_ptr.h](#)
- file [initializer_list](#)
- file [nested_exception.h](#)
- file [new](#)
- file [typeinfo](#)

3.15 include/parallel/ Directory Reference

Directory dependency graph for include/parallel/:



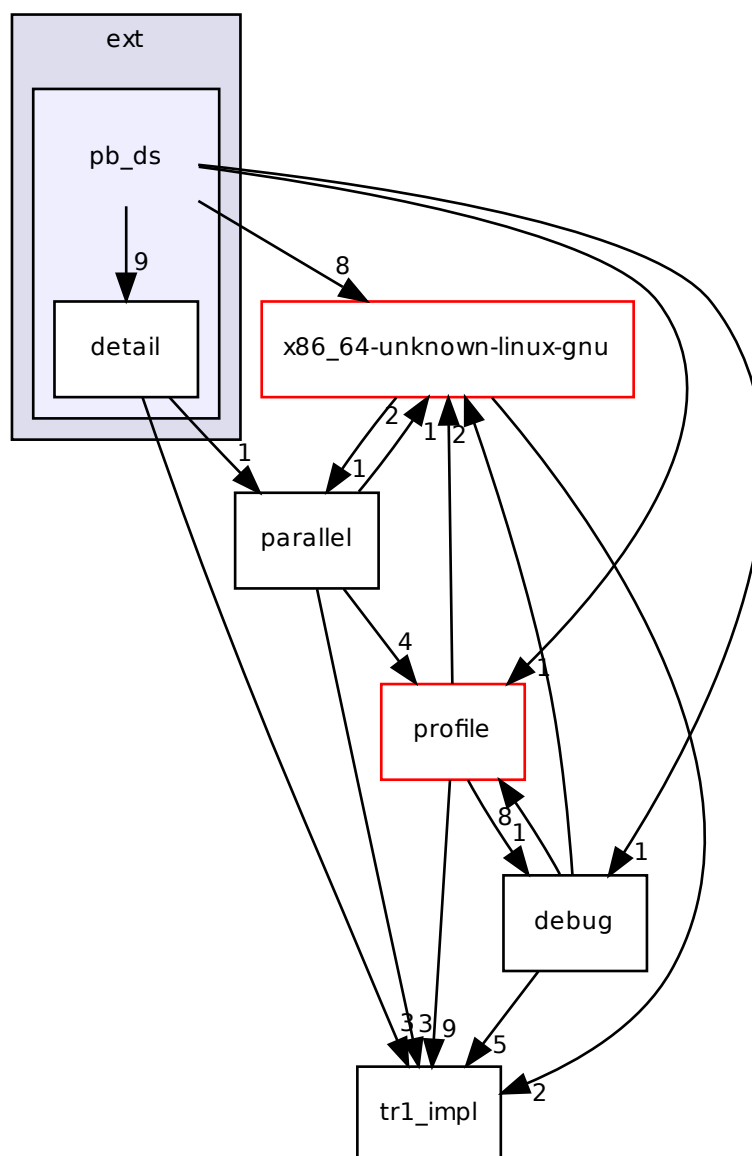
Files

- file `algo.h`
- file `algotbase.h`
- file `parallel/algorithm`
- file `parallel/algorithmfwd.h`
- file `balanced_quicksort.h`
- file `parallel/base.h`

- file [basic_iterator.h](#)
- file [checkers.h](#)
- file [parallel/compatibility.h](#)
- file [compiletime_settings.h](#)
- file [equally_split.h](#)
- file [features.h](#)
- file [find.h](#)
- file [find_selectors.h](#)
- file [for_each.h](#)
- file [for_each_selectors.h](#)
- file [iterator.h](#)
- file [list_partition.h](#)
- file [losertree.h](#)
- file [merge.h](#)
- file [multiseq_selection.h](#)
- file [multiway_merge.h](#)
- file [multiway_mergesort.h](#)
- file [parallel/numeric](#)
- file [numericfwd.h](#)
- file [omp_loop.h](#)
- file [omp_loop_static.h](#)
- file [par_loop.h](#)
- file [parallel.h](#)
- file [partial_sum.h](#)
- file [partition.h](#)
- file [queue.h](#)
- file [quicksort.h](#)
- file [random_number.h](#)
- file [random_shuffle.h](#)
- file [search.h](#)
- file [set_operations.h](#)
- file [settings.h](#)
- file [sort.h](#)
- file [tags.h](#)
- file [types.h](#)
- file [unique_copy.h](#)
- file [workstealing.h](#)

3.16 include/ext/pb_ds/ Directory Reference

Directory dependency graph for include/ext/pb_ds/:



Directories

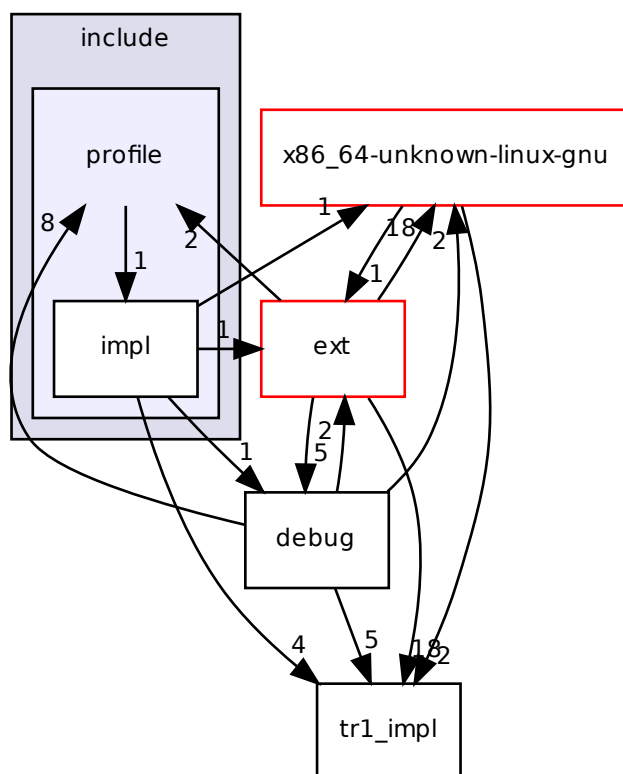
- directory [detail](#)

Files

- file [assoc_container.hpp](#)
- file [exception.hpp](#)
- file [hash_policy.hpp](#)
- file [list_update_policy.hpp](#)
- file [priority_queue.hpp](#)
- file [tag_and_trait.hpp](#)
- file [tree_policy.hpp](#)
- file [trie_policy.hpp](#)

3.17 include/profile/ Directory Reference

Directory dependency graph for include/profile/:



Directories

- directory [impl](#)

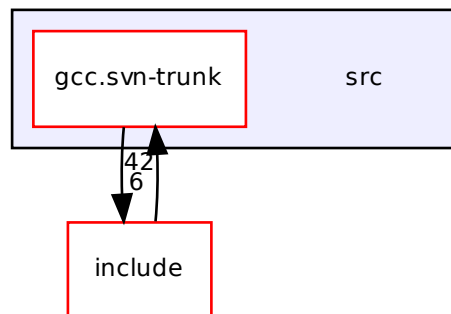
Files

- file [profile/base.h](#)

- file [profile/bitset](#)
- file [profile/deque](#)
- file [iterator_tracker.h](#)
- file [profile/list](#)
- file [profile/map](#)
- file [profile/map.h](#)
- file [profile/multimap.h](#)
- file [profile/multiset.h](#)
- file [profile/set](#)
- file [profile/set.h](#)
- file [profile/unordered_map](#)
- file [profile/unordered_set](#)
- file [profile/vector](#)

3.18 /mnt/share/src/ Directory Reference

Directory dependency graph for /mnt/share/src/:

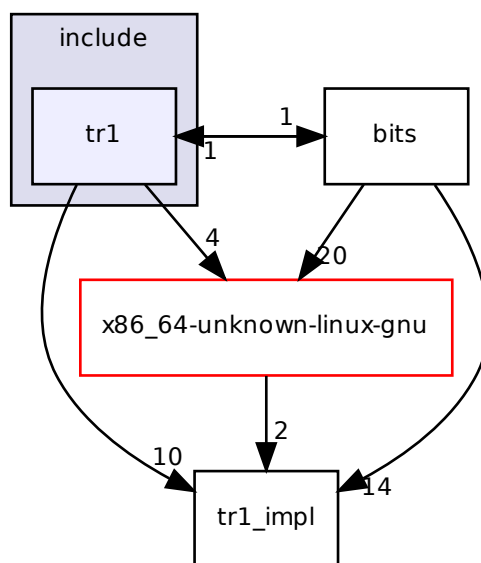


Directories

- directory [gcc.svn-trunk](#)

3.19 include/tr1/ Directory Reference

Directory dependency graph for include/tr1/:



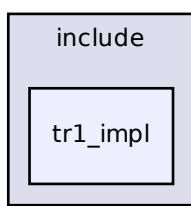
Files

- file [tr1/ccomplex](#)
- file [tr1/cctype](#)
- file [tr1/cfenv](#)
- file [tr1/cfloat](#)
- file [tr1/cinttypes](#)
- file [tr1/climits](#)
- file [tr1/cmath](#)
- file [tr1/complex](#)
- file [tr1/cstdarg](#)
- file [tr1/cstdbool](#)
- file [tr1/cstdint](#)

- file [tr1/cstdio](#)
- file [tr1/cstdlib](#)
- file [tr1/ctgmath](#)
- file [tr1/ctime](#)
- file [tr1/cwchar](#)
- file [tr1/cwctype](#)

3.20 include/tr1_impl/ Directory Reference

Directory dependency graph for include/tr1_impl/:

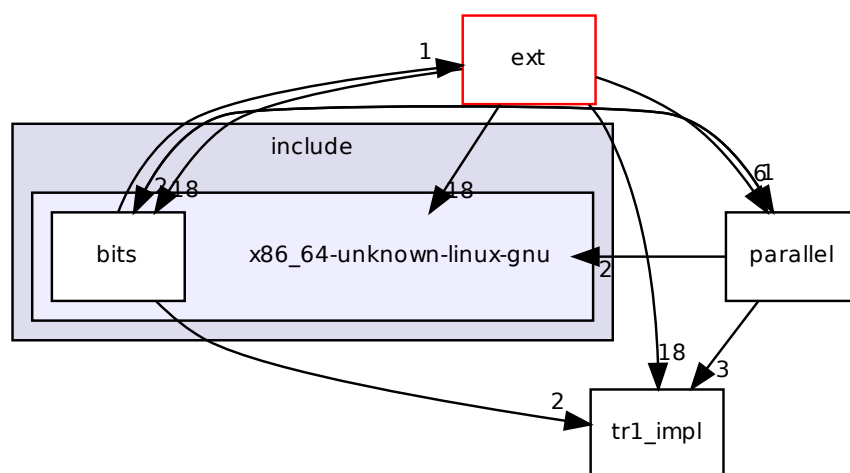


Files

- file [tr1_impl/array](#)
- file [boost_sp_counted_base.h](#)
- file [tr1_impl/cctype](#)
- file [tr1_impl/cfenv](#)
- file [tr1_impl/cinttypes](#)
- file [tr1_impl/cmath](#)
- file [tr1_impl/complex](#)
- file [tr1_impl/cstdint](#)
- file [tr1_impl/cstdio](#)
- file [tr1_impl/cstdlib](#)
- file [tr1_impl/cwchar](#)
- file [tr1_impl/cwctype](#)
- file [tr1_impl/type_traits](#)
- file [tr1_impl/utility](#)

3.21 include/x86_64-unknown-linux-gnu/ Directory Reference

Directory dependency graph for include/x86_64-unknown-linux-gnu/:



Directories

- directory [bits](#)

Chapter 4

Namespace Documentation

4.1 `__gnu_cxx` Namespace Reference

GNU extensions for public use.

Namespaces

- namespace [__detail](#)
- namespace [typelist](#)

Classes

- struct [__common_pool_policy](#)
Policy for shared `__pool` objects.
- class [__mt_alloc](#)
*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list).
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*
- class [__mt_alloc_base](#)
Base class for `_Tp` dependent member functions.
- struct [__per_type_pool_policy](#)
Policy for individual `__pool` objects.

- class [__pool< false >](#)
Specialization for single thread.
- class [__pool< true >](#)
Specialization for thread enabled, via `gthreads.h`.
- class [__pool_alloc](#)
Allocator using a memory pool with a single lock.
- class [__pool_alloc_base](#)
Base class for [__pool_alloc](#).
- struct [__pool_base](#)
Base class for pool object.
- class [__rc_string_base](#)
- class [__scoped_lock](#)
Scoped lock idiom.
- class [__versa_string](#)
*Template class [__versa_string](#).
Data structure managing sequences of characters and character-like objects.*
- struct [_Caster](#)
- struct [_Char_types](#)
Mapping from character type to associated types.
- class [_ExtPtr_allocator](#)
*An example allocator which uses a non-standard pointer type.
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).*
- struct [_Invalid_type](#)
- class [_Pointer_adapter](#)
- class [_Relative_pointer_impl](#)
A storage policy for use with [_Pointer_adapter<>](#) which stores the pointer's address as an offset value which is relative to its own address.
- class [_Relative_pointer_impl< const _Tp >](#)
- class [_Std_pointer_impl](#)
A storage policy for use with [_Pointer_adapter<>](#) which yields a standard pointer.

- struct `_Unqualified_type`
- struct `annotate_base`

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (`void`) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.*
- class `array_allocator`

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.
- class `array_allocator_base`

Base class.
- class `binary_compose`

An SGI extension .
- class `bitmap_allocator`

Bitmap Allocator, primary template.
- struct `char_traits`

Base class used to implement `std::char_traits`.
- struct `character`

A POD class that serves as a character abstraction class.
- struct `condition_base`

Base struct for condition policy.
- struct `constant_binary_fun`

An SGI extension .
- struct `constant_unary_fun`

An SGI extension .
- struct `constant_void_fun`

An SGI extension .
- class `debug_allocator`

*A meta-allocator with debugging bits, as per [20.4].
This is precisely the allocator defined in the C++ Standard.*

 - all allocation calls operator `new`
 - all deallocation calls operator `delete`.

- class [enc_filebuf](#)
class [enc_filebuf](#).
- struct [encoding_char_traits](#)
[encoding_char_traits](#)
- class [encoding_state](#)
Extension to use [iconv](#) for dealing with character encodings.
- struct [forced_error](#)
Thrown by exception safety machinery.
- class [free_list](#)
The free list class for managing chunks of memory to be given to and returned by the [bitmap_allocator](#).
- class [hash_map](#)
- class [hash_multimap](#)
- class [hash_multiset](#)
- class [hash_set](#)
- struct [limit_condition](#)
Base class for incremental control and throw.
- class [malloc_allocator](#)
*An allocator that uses [malloc](#).
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls [malloc](#)
 - all deallocation calls [free](#).
- class [new_allocator](#)
*An allocator that uses global [new](#), as per [20.4].
This is precisely the allocator defined in the C++ Standard.*
 - all allocation calls [operator new](#)
 - all deallocation calls [operator delete](#).
- struct [project1st](#)
An [SGI extension](#) .
- struct [project2nd](#)
An [SGI extension](#) .
- struct [random_condition](#)

Base class for random probability control and throw.

- struct `rb_tree`
- class `recursive_init_error`

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

- class `rope`
- struct `select1st`

An SGI extension .

- struct `select2nd`

An SGI extension .

- class `slist`
- class `stdio_filebuf`

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C `FILE`'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

- class `stdio_sync_filebuf`

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C `FILE`'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

- class `subtractive_rng`
- struct `temporary_buffer`
- class `throw_allocator_base`

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

- struct `throw_allocator_limit`

Allocator throwing via limit condition.

- struct `throw_allocator_random`

Allocator throwing via random condition.

- struct `throw_value_base`

Class with exception generation control. Intended to be used as a `value_type` in templated code.

- struct [throw_value_limit](#)
Type throwing via limit condition.
- struct [throw_value_random](#)
Type throwing via random condition.
- class [unary_compose](#)
An SGI extension .

Typedefs

- typedef void(* [__destroy_handler](#))(void *)
- typedef [__versa_string](#)< char, [std::char_traits](#)< char >, [std::allocator](#)< char >, [__rc_string_base](#) > [__rc_string](#)
- typedef [__vstring](#) [__sso_string](#)
- typedef [__versa_string](#)< char16_t, [std::char_traits](#)< char16_t >, [std::allocator](#)< char16_t >, [__rc_string_base](#) > [__u16rc_string](#)
- typedef [__u16vstring](#) [__u16sso_string](#)
- typedef [__versa_string](#)< char16_t > [__u16vstring](#)
- typedef [__versa_string](#)< char32_t, [std::char_traits](#)< char32_t >, [std::allocator](#)< char32_t >, [__rc_string_base](#) > [__u32rc_string](#)
- typedef [__u32vstring](#) [__u32sso_string](#)
- typedef [__versa_string](#)< char32_t > [__u32vstring](#)
- typedef [__versa_string](#)< char > [__vstring](#)
- typedef [__versa_string](#)< wchar_t, [std::char_traits](#)< wchar_t >, [std::allocator](#)< wchar_t >, [__rc_string_base](#) > [__wrc_string](#)
- typedef [__wvstring](#) [__wsso_string](#)
- typedef [__versa_string](#)< wchar_t > [__wvstring](#)
- typedef [rope](#)< char > [crope](#)
- typedef [rope](#)< wchar_t > [wrope](#)

Enumerations

- enum { [_S_num_primes](#) }
- enum [_Lock_policy](#) { [_S_single](#), [_S_mutex](#), [_S_atomic](#) }

Functions

- static void **__atomic_add** (volatile `_Atomic_word` * __mem, int __val)
- static void **__atomic_add_single** (`_Atomic_word` * __mem, int __val)
- static `_Atomic_word` **__attribute__** ((__unused__)) **__exchange_and_add_dispatch**(`_Atomic_word` * __mem
- template<class `_Tp` >
void **__aux_require_boolean_expr** (const `_Tp` & __t)
- template<typename `_ToType` , typename `_FromType` >
`_ToType` **__const_pointer_cast** (const `_FromType` & __arg)
- template<typename `_ToType` , typename `_FromType` >
`_ToType` **__const_pointer_cast** (`_FromType` * __arg)
- template<typename `_InputIterator` , typename `_Size` , typename `_OutputIterator` >
[pair](#)< `_InputIterator` , `_OutputIterator` > **__copy_n** (`_InputIterator` __first, `_Size` __count, `_OutputIterator` __result, [input_iterator_tag](#))
- template<typename `_RAIterator` , typename `_Size` , typename `_OutputIterator` >
[pair](#)< `_RAIterator` , `_OutputIterator` > **__copy_n** (`_RAIterator` __first, `_Size` __count, `_OutputIterator` __result, [random_access_iterator_tag](#))
- template<typename `_InputIterator` , typename `_Distance` >
void **__distance** (`_InputIterator` __first, `_InputIterator` __last, `_Distance` & __n, [std::input_iterator_tag](#))
- template<typename `_RandomAccessIterator` , typename `_Distance` >
void **__distance** (`_RandomAccessIterator` __first, `_RandomAccessIterator` __last, `_Distance` & __n, [std::random_access_iterator_tag](#))
- template<typename `_ToType` , typename `_FromType` >
`_ToType` **__dynamic_pointer_cast** (const `_FromType` & __arg)
- template<typename `_ToType` , typename `_FromType` >
`_ToType` **__dynamic_pointer_cast** (`_FromType` * __arg)
- void **__error_type_must_be_a_signed_integer_type** ()
- void **__error_type_must_be_an_integer_type** ()
- void **__error_type_must_be_an_unsigned_integer_type** ()
- static `_Atomic_word` **__exchange_and_add** (volatile `_Atomic_word` * __mem, int __val)
- static `_Atomic_word` **__exchange_and_add_single** (`_Atomic_word` * __mem, int __val)
- else return **__exchange_and_add_single** (__mem, __val)
- template<class `_Concept` >
void **__function_requires** ()
- template<typename `_Type` >
bool **__is_null_pointer** (`_Type` * __ptr)
- template<typename `_Type` >
bool **__is_null_pointer** (`_Type`)
- int **__lexicographical_compare_3way** (const unsigned char * __first1, const unsigned char * __last1, const unsigned char * __first2, const unsigned char * __last2)

- `int __lexicographical_compare_3way (const char *__first1, const char *__last1, const char *__first2, const char *__last2)`
- `template<typename _InputIterator1, typename _InputIterator2 >
int __lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp >
const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >
const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `crope::reference __mutable_reference_at (crope &__c, size_t __i)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >
_Tp __power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >
_Tp __power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >
_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >
_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _ToType, typename _FromType >
_ToType __reinterpret_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >
_ToType __reinterpret_pointer_cast (_FromType *__arg)`
- `_Slist_node_base * __slist_make_link (_Slist_node_base *__prev_node, _Slist_node_base *__new_node)`
- `_Slist_node_base * __slist_previous (_Slist_node_base *__head, const _Slist_node_base *__node)`
- `const _Slist_node_base * __slist_previous (const _Slist_node_base *__head, const _Slist_node_base *__node)`
- `_Slist_node_base * __slist_reverse (_Slist_node_base *__node)`
- `size_t __slist_size (_Slist_node_base *__node)`
- `void __slist_splice_after (_Slist_node_base *__pos, _Slist_node_base *__before_first, _Slist_node_base *__before_last)`
- `void __slist_splice_after (_Slist_node_base *__pos, _Slist_node_base *__head)`
- `template<typename _ToType, typename _FromType >
_ToType __static_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >
_ToType __static_pointer_cast (_FromType *__arg)`

- `size_t __stl_hash_string` (const char * __s)
- `unsigned long __stl_next_prime` (unsigned long __n)
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`
`_Ret __stoa` (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char
* __name, const _CharT * __str, std::size_t * __idx, _Base... __base)
- `void __throw_concurrency_lock_error` ()
- `void __throw_concurrency_unlock_error` ()
- `void __throw_forced_error` ()
- `template<typename _String, typename _CharT = typename _String::value_type>`
`_String __to_xstring` (int(*__convf)(_CharT *, std::size_t, const _CharT *, __-
builtin_va_list), std::size_t __n, const _CharT * __fmt,...)
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
[pair](#)< _InputIter, _ForwardIter > `__uninitialized_copy_n` (_InputIter __first, _-
Size __count, _ForwardIter __result, [std::input_iterator_tag](#))
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >`
[pair](#)< _RandomAccessIter, _ForwardIter > `__uninitialized_copy_n` (_-
RandomAccessIter __first, _Size __count, _ForwardIter __result, [std::random_-](#)
[access_iterator_tag](#))
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
[pair](#)< _InputIter, _ForwardIter > `__uninitialized_copy_n` (_InputIter __first, _-
Size __count, _ForwardIter __result)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`
[pair](#)< _InputIter, _ForwardIter > `__uninitialized_copy_n_a` (_InputIter __first,
_Size __count, _ForwardIter __result, _Allocator __alloc)
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`
[pair](#)< _InputIter, _ForwardIter > `__uninitialized_copy_n_a` (_InputIter __first,
_Size __count, _ForwardIter __result, [std::allocator](#)< _Tp >)
- `void __verbose_terminate_handler` ()
- `size_t __Bit_scan_forward` (size_t __num)
- `template<typename _ForwardIterator, typename _Allocator >`
`void __Destroy_const` (_ForwardIterator __first, _ForwardIterator __last, _-
Allocator __alloc)
- `template<typename _ForwardIterator, typename _Tp >`
`void __Destroy_const` (_ForwardIterator __first, _ForwardIterator __last, [alloca-](#)
[tor](#)< _Tp >)
- `template<class _CharT, class _Traits >`
`void __Rope_fill` ([basic_ostream](#)< _CharT, _Traits > & __o, size_t __n)
- `template<class _CharT >`
`bool __Rope_is_simple` (_CharT *)
- `bool __Rope_is_simple` (char *)
- `bool __Rope_is_simple` (wchar_t *)
- `template<class _Rope_iterator >`
`void __Rope_rotate` (_Rope_iterator __first, _Rope_iterator __middle, _Rope _-
iterator __last)

- `template<class _CharT >`
`void _S_cond_store_eos (_CharT &)`
- `void _S_cond_store_eos (char &__c)`
- `void _S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`
`_CharT _S_eos (_CharT *)`
- `bool _S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`
`bool _S_is_basic_char_type (_CharT *)`
- `bool _S_is_basic_char_type (char *)`
- `bool _S_is_one_byte_char_type (char *)`
- `template<class _CharT >`
`bool _S_is_one_byte_char_type (_CharT *)`
- `template<class _Operation1 , class _Operation2 >`
`unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1 , class _Operation2 , class _Operation3 >`
`binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<class _Result >`
`constant_void_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result >`
`constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result >`
`constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &__val)`
- `template<typename _InputIterator , typename _Size , typename _OutputIterator >`
`pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator , typename _Tp , typename _Size >`
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator , typename _Predicate , typename _Size >`
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator , typename _Distance >`
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<class _Tp >`
`_Tp identity_element (std::plus< _Tp >)`
- `template<class _Tp >`
`_Tp identity_element (std::multiplies< _Tp >)`
- `static _Atomic_word int __val if (__gthread_active_p()) return __exchange_and_add(__mem`

- `template<typename _ForwardIter, typename _Tp >`
`void iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`
`StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _-`
`StrictWeakOrdering __comp)`
- `template<typename _ForwardIterator >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::* __f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::* __f)(_Arg))`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &_
__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs,`
`const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator!= (const __normal_iterator< _Iterator, _Container > &__lhs,`
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp, typename _Array >`
`bool operator!= (const array_allocator< _Tp, _Array > &, const array_-`
`allocator< _Tp, _Array > &)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const bitmap_allocator< _Tp1 > &, const bitmap_allocator<`
`_Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits,`
`_Alloc, _Base > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const malloc_allocator< _Tp > &, const malloc_allocator<`
`_Tp > &)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const`
`_Rope_const_iterator< _CharT, _Alloc > &__y)`

- `template<typename _Tp, typename _Poolp >`
`bool operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp >`
`bool operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp >`
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`
`bool operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp >`
`bool operator!= (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<typename _Tp, typename _Cond >`
`bool operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<class _Tp, class _Alloc >`
`bool operator!= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator, _Container > operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator+ (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- [__versa_string](#)< _CharT, _Traits, _Alloc, _Base > **operator+** (const [__versa_string](#)< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _Cond >
[throw_value_base](#)< _Cond > **operator+** (const [throw_value_base](#)< _Cond > &__a, const [throw_value_base](#)< _Cond > &__b)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
[__versa_string](#)< _CharT, _Traits, _Alloc, _Base > **operator+** (const [__versa_string](#)< _CharT, _Traits, _Alloc, _Base > &__lhs, const [__versa_string](#)< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
[__versa_string](#)< _CharT, _Traits, _Alloc, _Base > **operator+** (const _CharT * __lhs, const [__versa_string](#)< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
[__versa_string](#)< _CharT, _Traits, _Alloc, _Base > **operator+** (_CharT __lhs, const [__versa_string](#)< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
[__versa_string](#)< _CharT, _Traits, _Alloc, _Base > **operator+** (const [__versa_string](#)< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)
- template<class _CharT, class _Alloc >
[rope](#)< _CharT, _Alloc > & **operator+=** ([rope](#)< _CharT, _Alloc > &__left, const [rope](#)< _CharT, _Alloc > &__right)
- template<class _CharT, class _Alloc >
[rope](#)< _CharT, _Alloc > & **operator+=** ([rope](#)< _CharT, _Alloc > &__left, const _CharT * __right)
- template<class _CharT, class _Alloc >
[rope](#)< _CharT, _Alloc > & **operator+=** ([rope](#)< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc >
_Rope_const_iterator< _CharT, _Alloc > **operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<typename _IteratorL, typename _IteratorR, typename _Container >
auto **operator-** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)-> decltype(__lhs.base()-__rhs.base())
- template<typename _Iterator, typename _Container >
__normal_iterator< _Iterator, _Container >::difference_type **operator-** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<class _CharT, class _Alloc >
ptrdiff_t **operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)

- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator- (const _Rope_iterator< _CharT,`
`_Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`ptrdiff_t operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _`
`Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator- (const throw_value_base< _Cond >`
`&__a, const throw_value_base< _Cond > &__b)`
- `template<class _Tp, class _Alloc >`
`bool operator< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc`
`> &_SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator< (const __normal_iterator< _IteratorL, _Container > &__lhs,`
`const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator< (const __normal_iterator< _Iterator, _Container > &__lhs,`
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const`
`_Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _`
`Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename V, typename I, typename S >`
`bool operator< (const character< V, I, S > &lhs, const character< V, I, S >`
`&rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_`
`adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT,`
`_Alloc > &__right)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits,`
`_Alloc, _Base > &__rhs)`
- `template<typename _Cond >`
`bool operator< (const throw_value_base< _Cond > &__a, const throw_`
`value_base< _Cond > &__b)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp >`
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _Tp, class _Alloc >`
`bool operator<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator== (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Tp >`
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator== (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp >`
`bool operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<typename _Tp, typename _Cond >`
`bool operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp >`
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`

- `template<class _Tp, class _Alloc >`
`bool operator== (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`
`__enable_if< std::__is_char< _CharT >::__value, bool >::__type operator==`
`(const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _`
`CharT >, _Base > &__lhs, const __versa_string< _CharT, std::char_traits< _`
`CharT >, std::allocator< _CharT >, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool operator== (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc >`
`&__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp >`
`bool operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp, typename _Poolp >`
`bool operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp >`
`bool operator== (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename V, typename I, typename S >`
`bool operator== (const character< V, I, S > &lhs, const character< V, I, S > &rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc >`
`&__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp >`
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- bool **operator==** (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &_
_lhs, const `_CharT` *__rhs)
- template<typename `_Cond` >
bool **operator==** (const `throw_value_base`< `_Cond` > &__a, const `throw_`-
`value_base`< `_Cond` > &__b)
- template<typename `_Tp`, typename `_Array` >
bool **operator==** (const `array_allocator`< `_Tp`, `_Array` > &, const `array_`-
`allocator`< `_Tp`, `_Array` > &)
- template<class `_CharT`, class `_Alloc` >
bool **operator==** (const `_Rope_const_iterator`< `_CharT`, `_Alloc` > &__x, const
`_Rope_const_iterator`< `_CharT`, `_Alloc` > &__y)
- template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename,
typename > class `_Base`>
bool **operator==** (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &_
_lhs, const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &__rhs)
- template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename,
typename > class `_Base`>
bool **operator==** (const `_CharT` *__lhs, const `__versa_string`< `_CharT`, `_Traits`,
`_Alloc`, `_Base` > &__rhs)
- template<typename `_IteratorL`, typename `_IteratorR`, typename `_Container` >
bool **operator>** (const `__normal_iterator`< `_IteratorL`, `_Container` > &__lhs,
const `__normal_iterator`< `_IteratorR`, `_Container` > &__rhs)
- template<typename `_Tp` >
bool **operator>** (const `_Pointer_adapter`< `_Tp` > &__lhs, const `_Pointer_`-
`adapter`< `_Tp` > &__rhs)
- template<typename `_Iterator`, typename `_Container` >
bool **operator>** (const `__normal_iterator`< `_Iterator`, `_Container` > &__lhs,
const `__normal_iterator`< `_Iterator`, `_Container` > &__rhs)
- template<typename `_CharT`, typename `_Traits`, typename `_Alloc`, template< typename, typename,
typename > class `_Base`>
bool **operator>** (const `__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `_Base` > &_
lhs, const `_CharT` *__rhs)
- template<class `_CharT`, class `_Alloc` >
bool **operator>** (const `_Rope_const_iterator`< `_CharT`, `_Alloc` > &__x, const
`_Rope_const_iterator`< `_CharT`, `_Alloc` > &__y)
- template<typename `_Tp1`, typename `_Tp2` >
bool **operator>** (const `_Pointer_adapter`< `_Tp1` > &__lhs, `_Tp2` __rhs)
- template<class `_CharT`, class `_Alloc` >
bool **operator>** (const `_Rope_iterator`< `_CharT`, `_Alloc` > &__x, const `_`-
`Rope_iterator`< `_CharT`, `_Alloc` > &__y)
- template<typename `_Tp1`, typename `_Tp2` >
bool **operator>** (`_Tp1` __lhs, const `_Pointer_adapter`< `_Tp2` > &__rhs)
- template<class `_CharT`, class `_Alloc` >
bool **operator>** (const `rope`< `_CharT`, `_Alloc` > &__x, const `rope`< `_CharT`,
`_Alloc` > &__y)

- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<class _Tp, class _Alloc >`
`bool operator> (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Tp >`
`bool operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

- bool **operator**>= (const `__versa_string`< _CharT, _Traits, _Alloc, _Base > &_
_lhs, const _CharT * __rhs)
- template<typename _Tp1, typename _Tp2 >
bool **operator**>= (_Tp1 __lhs, const `__Pointer_adapter`< _Tp2 > & __rhs)
- template<class _Tp, class _Alloc >
bool **operator**>= (const `slist`< _Tp, _Alloc > &_SL1, const `slist`< _Tp, _Alloc
> &_SL2)
- template<typename _Tp1, typename _Tp2 >
bool **operator**>= (const `__Pointer_adapter`< _Tp1 > & __lhs, const `__Pointer_ -`
`adapter`< _Tp2 > & __rhs)
- template<typename _Tp, typename _Integer, typename _MonoidOperation >
_Tp **power** (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _Tp, typename _Integer >
_Tp **power** (_Tp __x, _Integer __n)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _
RandomNumberGenerator >
_RandomAccessIterator **random_sample** (_InputIterator __first, _InputIterator
__last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last,
_RandomNumberGenerator & __rand)
- template<typename _InputIterator, typename _RandomAccessIterator >
_RandomAccessIterator **random_sample** (_InputIterator __first, _InputIterator
__last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename
_RandomNumberGenerator >
_OutputIterator **random_sample_n** (_ForwardIterator __first, _ForwardIterator
__last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator
& __rand)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >
_OutputIterator **random_sample_n** (_ForwardIterator __first, _ForwardIterator
__last, _OutputIterator __out, const _Distance __n)
- void **rotate** (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __
middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __
last)
- double **stod** (const `__vstring` & __str, std::size_t * __idx=0)
- float **stof** (const `__vstring` & __str, std::size_t * __idx=0)
- int **stoi** (const `__vstring` & __str, std::size_t * __idx=0, int __base=10)
- long **stol** (const `__vstring` & __str, std::size_t * __idx=0, int __base=10)
- long double **stold** (const `__vstring` & __str, std::size_t * __idx=0)
- long long **stoll** (const `__vstring` & __str, std::size_t * __idx=0, int __base=10)
- unsigned long **stoul** (const `__vstring` & __str, std::size_t * __idx=0, int __
base=10)
- unsigned long long **stoull** (const `__vstring` & __str, std::size_t * __idx, int __
base=10)

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`void swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _CharT, class __Alloc >`
`void swap (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)`
- `template<typename _Cond >`
`void swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`
- `template<class _Tp, class _Alloc >`
`void swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)`
- `template<typename _Tp >`
`void swap (_ExtPtr_allocator< _Tp > &__larg, ExtPtr_allocator< _Tp > &__rarg)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`void swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`void swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`void swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`
`void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)`
- `template<class _CharT, class _Alloc >`
`void swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`
- `__vstring to_string (long double __val)`
- `__vstring to_string (double __val)`
- `__vstring to_string (float __val)`
- `__vstring to_string (unsigned __val)`
- `__vstring to_string (unsigned long __val)`
- `__vstring to_string (unsigned long long __val)`
- `__vstring to_string (int __val)`
- `__vstring to_string (long long __val)`
- `__vstring to_string (long __val)`
- `__wvstring to_wstring (double __val)`
- `__wvstring to_wstring (long long __val)`
- `__wvstring to_wstring (long double __val)`

- [__wvstring to_wstring](#) (long __val)
- [__wvstring to_wstring](#) (float __val)
- [__wvstring to_wstring](#) (unsigned long __val)
- [__wvstring to_wstring](#) (unsigned long long __val)
- [__wvstring to_wstring](#) (unsigned __val)
- [__wvstring to_wstring](#) (int __val)
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
[pair](#)< _InputIter, _ForwardIter > [uninitialized_copy_n](#) (_InputIter __first, _Size __count, _ForwardIter __result)

Variables

- static const _Lock_policy [__default_lock_policy](#)
- static const unsigned long [__stl_prime_list](#) [_S_num_primes]
- static _Atomic_word int __val [__val](#)
- [rope](#)< _CharT, _Alloc > [identity_element](#) (_Rope_Concat_fn< _CharT, _Alloc >)

4.1.1 Detailed Description

GNU extensions for public use.

4.1.2 Function Documentation

4.1.2.1 `template<typename _ToType, typename _FromType > _ToType
 __gnu_cxx::__static_pointer_cast (const _FromType & __arg)
 [inline]`

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 61 of file `cast.h`.

References `__static_pointer_cast()`.

Referenced by `__static_pointer_cast()`.

4.1.2.2 `template<typename _ToType, typename _FromType > _ToType
 __gnu_cxx::__static_pointer_cast (_FromType * __arg) [inline]`

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

Definition at line 89 of file cast.h.

References `__static_pointer_cast()`.

4.1.2.3 `size_t __gnu_cxx::Bit_scan_forward (size_t __num) [inline]`

Generic Version of the bsf instruction.

Definition at line 512 of file bitmap_allocator.h.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object()`.

4.1.2.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=(const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Test difference of C string and string.

Parameters

`__lhs` C string.

`__rhs` String.

Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2175 of file vstring.h.

4.1.2.5 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=(const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Test difference of two strings.

Parameters

`__lhs` First string.

`__rhs` Second string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2162 of file vstring.h.

4.1.2.6 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator!=(const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT * __rhs) [inline]`

Test difference of string and C string.

Parameters

`__lhs` String.

`__rhs` C string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2188 of file `vstring.h`.

4.1.2.7 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >
__gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __lhs, _CharT __rhs)`

Concatenate string and character.

Parameters

`__lhs` First string.

`__rhs` Last string.

Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 239 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.1.2.8 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >
__gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __rhs)`

Concatenate two strings.

Parameters

`__lhs` First string.

`__rhs` Last string.

Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 179 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.1.2.9 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >
__gnu_cxx::operator+ (const _CharT * __lhs, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __rhs)`

Concatenate C string and string.

Parameters

`__lhs` First string.

`__rhs` Last string.

Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 192 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.1.2.10 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >
__gnu_cxx::operator+ (_CharT __lhs, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __rhs)`

Concatenate character and string.

Parameters

`__lhs` First string.

`__rhs` Last string.

Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 209 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.1.2.11 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base >
__gnu_cxx::operator+ (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT * __rhs)`

Concatenate string and C string.

Parameters

`__lhs` First string.

`__rhs` Last string.

Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 222 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.1.2.12 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator< (const _CharT * __lhs, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Test if C string precedes string.

Parameters

`__lhs` C string.

`__rhs` String.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2228 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.13 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs) [inline]`

Test if string precedes C string.

Parameters

`__lhs` String.

`__rhs` C string.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2215 of file `vstring.h`.

4.1.2.14 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Test if string precedes string.

Parameters

`__lhs` First string.
`__rhs` Second string.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2202 of file `vstring.h`.

```
4.1.2.15 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator<= ( const _CharT * __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )
[inline]
```

Test if C string doesn't follow string.

Parameters

`__lhs` C string.
`__rhs` String.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2308 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.16 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __rhs ) [inline]
```

Test if string doesn't follow string.

Parameters

`__lhs` First string.
`__rhs` Second string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2282 of file `vstring.h`.

```
4.1.2.17  template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT* __rhs ) [inline]
```

Test if string doesn't follow C string.

Parameters

`__lhs` String.

`__rhs` C string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2295 of file `vstring.h`.

```
4.1.2.18  template<typename _Tp > bool __gnu_cxx::operator== ( const
_Pointer_adapter< _Tp > & __lhs, const _Pointer_adapter< _Tp >
& __rhs ) [inline]
```

Comparison operators for [_Pointer_adapter](#) defer to the base class's comparison operators, when possible.

Definition at line 529 of file `pointer.h`.

```
4.1.2.19  template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator== ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT* __rhs ) [inline]
```

Test equivalence of string and C string.

Parameters

`__lhs` String.

`__rhs` C string.

Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2148 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.20  template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator==( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __rhs ) [inline]
```

Test equivalence of two strings.

Parameters

`__lhs` First string.

`__rhs` Second string.

Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2111 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.21  template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator==( const _CharT * __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )
[inline]
```

Test equivalence of C string and string.

Parameters

`__lhs` C string.

`__rhs` String.

Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2135 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.22 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator> ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT * __rhs ) [inline]
```

Test if string follows C string.

Parameters

`__lhs` String.

`__rhs` C string.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2255 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
4.1.2.23 template<typename _CharT , typename _Traits , typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator> ( const _CharT * __lhs, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if C string follows string.

Parameters

`__lhs` C string.

`__rhs` String.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2268 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.24 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator> (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __rhs) [inline]`

Test if string follows string.

Parameters

`__lhs` First string.

`__rhs` Second string.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2242 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.25 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator>= (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __rhs) [inline]`

Test if string doesn't precede string.

Parameters

`__lhs` First string.

`__rhs` Second string.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2322 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.26 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator>= (const _CharT * __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
[inline]`

Test if C string doesn't precede string.

Parameters

`__lhs` C string.

`__rhs` String.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2348 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.27 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base> bool
__gnu_cxx::operator>= (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __lhs, const _CharT * __rhs) [inline]`

Test if string doesn't precede C string.

Parameters

`__lhs` String.

`__rhs` C string.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2335 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

4.1.2.28 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Swap contents of two strings.

Parameters

`__lhs` First string.

`__rhs` Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2362 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap()`.

4.2 `__gnu_cxx::__detail` Namespace Reference

Implementation details not part of the namespace `__gnu_cxx` interface.

Classes

- class `__mini_vector`
`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.
- class `_Bitmap_counter`
The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.
- class `_Ffit_finder`
The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Enumerations

- enum { `_S_max_rope_depth` }
- enum { `bits_per_byte`, `bits_per_block` }
- enum `_Tag` { `_S_leaf`, `_S_concat`, `_S_substringfn`, `_S_function` }

Functions

- void [__bit_allocate](#) (size_t *__pmap, size_t __pos) throw ()
- void [__bit_free](#) (size_t *__pmap, size_t __pos) throw ()
- template<typename _ForwardIterator, typename _Tp, typename _Compare >
_ForwardIterator [__lower_bound](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _AddrPair >
size_t [__num_bitmaps](#) (_AddrPair __ap)
- template<typename _AddrPair >
size_t [__num_blocks](#) (_AddrPair __ap)

4.2.1 Detailed Description

Implementation details not part of the namespace [__gnu_cxx](#) interface.

4.2.2 Function Documentation

4.2.2.1 void [__gnu_cxx::__detail::__bit_allocate](#) (size_t * [__pmap](#), size_t [__pos](#)) throw () [[inline](#)]

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 491 of file [bitmap_allocator.h](#).

Referenced by [__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object\(\)](#).

4.2.2.2 void [__gnu_cxx::__detail::__bit_free](#) (size_t * [__pmap](#), size_t [__pos](#)) throw () [[inline](#)]

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 502 of file [bitmap_allocator.h](#).

Referenced by [__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object\(\)](#).

4.2.2.3 template<typename _AddrPair > size_t [__gnu_cxx::__detail::__num_bitmaps](#) (_AddrPair [__ap](#)) [[inline](#)]

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 279 of file [bitmap_allocator.h](#).

References `__num_blocks()`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::M_allocate_single_object()`,
and `__gnu_cxx::bitmap_allocator< _Tp >::M_deallocate_single_object()`.

4.2.2.4 `template<typename _AddrPair > size_t __gnu_cxx::detail::__num_blocks (_AddrPair __ap)`
`[inline]`

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 271 of file `bitmap_allocator.h`.

Referenced by `__num_bitmaps()`.

4.3 `__gnu_cxx::typelist` Namespace Reference

GNU `typelist` extensions for public compile-time use.

Functions

- `template<typename Fn , typename Typelist >`
`void apply (Fn &, Typelist)`
- `template<typename Fn , typename TypelistT , typename TypelistV >`
`void apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Fn , typename Typelist >`
`void apply_generator (Fn &fn, Typelist)`
- `template<typename Gn , typename TypelistT , typename TypelistV >`
`void apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Gn , typename Typelist >`
`void apply_generator (Gn &, Typelist)`

4.3.1 Detailed Description

GNU `typelist` extensions for public compile-time use.

4.3.2 Function Documentation

4.3.2.1 `template<typename Gn , typename Typelist > void`
`__gnu_cxx::typelist::apply_generator (Gn &, Typelist)`

Apply all `typelist` types to generator functor.

4.4 `__gnu_debug` Namespace Reference

GNU debug classes for public use.

Classes

- struct [__is_same](#)
- class [_After_nth_from](#)
- struct [_BeforeBeginHelper](#)
- class [_Not_equal_to](#)
- class [_Safe_iterator](#)
Safe iterator wrapper.
- class [_Safe_iterator_base](#)
Basic functionality for a safe iterator.
- class [_Safe_sequence](#)
Base class for constructing a safe sequence type that tracks iterators that reference it.
- class [_Safe_sequence_base](#)
Base class that supports tracking of iterators that reference a sequence.
- class [basic_string](#)
Class `std::basic_string` with safety/checking/debug instrumentation.

Typedefs

- typedef [basic_string](#)< char > **string**
- typedef [basic_string](#)< wchar_t > **wstring**

Enumerations

- enum [_Debug_msg_id](#) {
[__msg_valid_range](#), [__msg_insert_singular](#), [__msg_insert_different](#), [__msg_erase_bad](#),
[__msg_erase_different](#), [__msg_subscript_oob](#), [__msg_empty](#), [__msg_unpartitioned](#),
[__msg_unpartitioned_pred](#), [__msg_unsorted](#), [__msg_unsorted_pred](#), [__msg_not_heap](#),


```

__msg_not_heap_pred, __msg_bad_bitset_write, __msg_bad_bitset_read,
__msg_bad_bitset_flip,
__msg_self_splice, __msg_splice_alloc, __msg_splice_bad, __msg_splice_
other,
__msg_splice_overlap, __msg_init_singular, __msg_init_copy_singular, __
msg_init_const_singular,
__msg_copy_singular, __msg_bad_deref, __msg_bad_inc, __msg_bad_dec,
__msg_iter_subscript_oob, __msg_advance_oob, __msg_retreat_oob, __
msg_iter_compare_bad,
__msg_compare_different, __msg_iter_order_bad, __msg_order_different,
__msg_distance_bad,
__msg_distance_different, __msg_deref_istream, __msg_inc_istream, __
msg_output_ostream,
__msg_deref_istreambuf, __msg_inc_istreambuf, __msg_insert_after_end,
__msg_erase_after_bad,
__msg_valid_range2 }

```

Functions

- `template<typename _Iterator >`
`_Siter_base< _Iterator >::iterator_type __base(_Iterator __it)`
- `template<typename _Iterator >`
`bool __check_dereferenceable(_Iterator &)`
- `template<typename _Tp >`
`bool __check_dereferenceable(const _Tp *__ptr)`
- `template<typename _Iterator, typename _Sequence >`
`bool __check_dereferenceable(const _Safe_iterator< _Iterator, _Sequence >`
`&__x)`
- `template<typename _ForwardIterator, typename _Tp >`
`bool __check_partitioned_lower(_ForwardIterator __first, _ForwardIterator _`
`__last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`
`bool __check_partitioned_lower(_ForwardIterator __first, _ForwardIterator _`
`__last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`bool __check_partitioned_upper(_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`
`bool __check_partitioned_upper(_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__value, _Pred __pred)`
- `template<typename _Iterator >`
`bool __check_singular(_Iterator &)`

- `template<typename _Iterator, typename _Sequence >`
`bool __check_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)`
- `template<typename _Tp >`
`bool __check_singular (const _Tp *__ptr)`
- `bool __check_singular_aux (const void *)`
- `bool __check_singular_aux (const _Safe_iterator_base *__x)`
- `template<typename _InputIterator >`
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`
`bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::__true_type)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::__true_type)`
- `template<typename _InputIterator >`
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::__false_type)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::__false_type)`
- `template<typename _CharT >`
`const _CharT * __check_string (const _CharT *__s)`

- `template<typename _CharT, typename _Integer >`
`const _CharT * __check_string (const _CharT *__s, const _Integer &__n __-`
`attribute__((__unused__)))`
- `template<typename _InputIterator >`
`_InputIterator __check_valid_range (const _InputIterator &__first, const _`
`InputIterator &__last __attribute__((__unused__)))`
- `template<typename _InputIterator >`
`bool __valid_range (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _Iterator, typename _Sequence >`
`bool __valid_range (const _Safe_iterator< _Iterator, _Sequence > &__first,`
`const _Safe_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _Integral >`
`bool __valid_range_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator >`
`bool __valid_range_aux (const _InputIterator &__first, const _InputIterator &_`
`__last, std::__false_type)`
- `template<typename _InputIterator >`
`bool __valid_range_aux2 (const _InputIterator &, const _InputIterator &,`
`std::input_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`bool __valid_range_aux2 (const _RandomAccessIterator &__first, const _`
`RandomAccessIterator &__last, std::random_access_iterator_tag)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`std::basic_istream< _CharT, _Traits > & getline (std::basic_istream< _CharT,`
`_Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT`
`__delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`std::basic_istream< _CharT, _Traits > & getline (std::basic_istream< _CharT,`
`_Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator!= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator!= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`
`Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool operator!= (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > operator+ (const _CharT *__lhs,`
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`
`_CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`_Safe_iterator< _Iterator, _Sequence > operator+ (typename _Safe_iterator<`
`_Iterator, _Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _`
`Sequence > &__i)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`
`_CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > operator+ (const basic_string<`
`_CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _`
`Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`basic_string< _CharT, _Traits, _Allocator > operator+ (_CharT __lhs, const`
`basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`_Safe_iterator< _Iterator, _Sequence >::difference_type operator- (const _`
`Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator,`
`_Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`_Safe_iterator< _IteratorL, _Sequence >::difference_type operator- (const _`
`Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _`
`IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator< (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const`
`_Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool operator< (const basic_string< _CharT, _Traits, _Allocator > &__lhs,`
`const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _`
`Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator< (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const`
`_Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`

-
- ```

 _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator >
 &__str)

```
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator<= (const _CharT *__lhs, const basic\_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
  - `template<typename _Iterator, typename _Sequence >`  
`bool operator<= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
  - `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator<= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
  - `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator== (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
  - `template<typename _Iterator, typename _Sequence >`  
`bool operator== (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator== (const _CharT *__lhs, const basic\_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
  - `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator> (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
  - `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _`  
`Allocator > &__rhs)`
  - `template<typename _Iterator, typename _Sequence >`  
`bool operator> (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
-

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator> (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const _CharT *__lhs, const basic\_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool operator>= (const \_Safe\_iterator< _Iterator, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs,`  
`const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool operator>= (const \_Safe\_iterator< _IteratorL, _Sequence > &__lhs, const`  
`\_Safe\_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _`  
`CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void swap (basic\_string< _CharT, _Traits, _Allocator > &__lhs, basic\_string<`  
`_CharT, _Traits, _Allocator > &__rhs)`

### 4.4.1 Detailed Description

GNU debug classes for public use.

### 4.4.2 Function Documentation

#### 4.4.2.1 `template<typename _Iterator > _Siter_base<_Iterator>::iterator_type` `__gnu_debug::__base ( _Iterator __it ) [inline]`

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the `__invalid_range` function thanks to the `<` operator.

Definition at line 697 of file `safe_iterator.h`.

Referenced by `std::internal()`.

**4.4.2.2** `template<typename _Iterator > bool __gnu_debug::__check_dereferenceable ( _Iterator & )`  
`[inline]`

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 70 of file functions.h.

**4.4.2.3** `template<typename _Tp > bool __gnu_debug::__check_dereferenceable ( const _Tp * __ptr )`  
`[inline]`

Non-NULL pointers are dereferenceable.

Definition at line 76 of file functions.h.

**4.4.2.4** `template<typename _Iterator , typename _Sequence > bool __gnu_debug::__check_dereferenceable ( const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 82 of file functions.h.

References `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

**4.4.2.5** `template<typename _Iterator , typename _Sequence > bool __gnu_debug::__check_singular ( const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline]`

Safe iterators know if they are singular.

Definition at line 63 of file functions.h.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**4.4.2.6** `template<typename _Tp > bool __gnu_debug::__check_singular ( const _Tp * __ptr ) [inline]`

Non-NULL pointers are nonsingular.

Definition at line 57 of file functions.h.

**4.4.2.7** `bool __gnu_debug::__check_singular_aux ( const _Safe_iterator_base * __x ) [inline]`

Iterators that derive from [\\_Safe\\_iterator\\_base](#) but that aren't `_Safe_iterators` can be determined singular or non-singular via [\\_Safe\\_iterator\\_base](#).

Definition at line 61 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

**4.4.2.8** `template<typename _CharT > const _CharT* __gnu_debug::__check_string ( const _CharT * __s ) [inline]`

Checks that `__s` is non-NULL and then returns `__s`.

Definition at line 176 of file `functions.h`.

**4.4.2.9** `template<typename _CharT, typename _Integer > const _CharT* __gnu_debug::__check_string ( const _CharT * __s, const _Integer & __n __attribute__((unused)) ) [inline]`

Checks that `__s` is non-NULL or `__n == 0`, and then returns `__s`.

Definition at line 164 of file `functions.h`.

**4.4.2.10** `template<typename _InputIterator > bool __gnu_debug::__valid_range ( const _InputIterator & __first, const _InputIterator & __last ) [inline]`

Don't know what these iterators are, or if they are even iterators (we may get an integral type for `InputIterator`), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 134 of file `functions.h`.

References `__valid_range_aux()`.

**4.4.2.11** `template<typename _Iterator, typename _Sequence > bool __gnu_debug::__valid_range ( const _Safe_iterator< _Iterator, _Sequence > & __first, const _Safe_iterator< _Iterator, _Sequence > & __last ) [inline]`

Safe iterators know how to check if they form a valid range.

Definition at line 143 of file `functions.h`.



**4.4.2.12** `template<typename _Integral > bool __gnu_debug::__valid_range_ -  
aux ( const _Integral &, const _Integral &, std::__true_type )  
[inline]`

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 111 of file functions.h.

Referenced by `__valid_range()`.

**4.4.2.13** `template<typename _InputIterator > bool __gnu_debug::__valid_ -  
range_aux ( const _InputIterator & __first, const _InputIterator &  
__last, std::__false_type ) [inline]`

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 119 of file functions.h.

References `__valid_range_aux2()`.

**4.4.2.14** `template<typename _InputIterator > bool __gnu_debug::__valid_ -  
range_aux2 ( const _InputIterator &, const _InputIterator & ,  
std::input_iterator_tag ) [inline]`

Can't test for a valid range with input iterators, because iteration may be destructive. So we just assume that the range is valid.

Definition at line 101 of file functions.h.

**4.4.2.15** `template<typename _RandomAccessIterator > bool  
__gnu_debug::__valid_range_aux2 ( const _RandomAccessIterator  
& __first, const _RandomAccessIterator & __last,  
std::random_access_iterator_tag ) [inline]`

If the distance between two random access iterators is nonnegative, assume the range is valid.

Definition at line 90 of file functions.h.

Referenced by `__valid_range_aux()`.

## 4.5 `__gnu_internal` Namespace Reference

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

### 4.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

## 4.6 `__gnu_parallel` Namespace Reference

GNU parallel code for public use.

### Classes

- struct [`\_\_accumulate\_binop\_reduct`](#)  
*General reduction, using a binary operator.*
- struct [`\_\_accumulate\_selector`](#)  
*`std::accumulate()` selector.*
- struct [`\_\_adjacent\_difference\_selector`](#)  
*Selector that returns the difference between two adjacent `__elements`.*
- struct [`\_\_adjacent\_find\_selector`](#)  
*Test predicate on two adjacent elements.*
- class [`\_\_binder1st`](#)  
*Similar to `std::binder1st`, but giving the argument types explicitly.*
- class [`\_\_binder2nd`](#)  
*Similar to `std::binder2nd`, but giving the argument types explicitly.*
- struct [`\_\_count\_if\_selector`](#)  
*`std::count_if()` selector.*
- struct [`\_\_count\_selector`](#)  
*`std::count()` selector.*

- struct `__fill_selector`  
*`std::fill()` selector.*
- struct `__find_first_of_selector`  
*Test predicate on several elements.*
- struct `__find_if_selector`  
*Test predicate on a single element, used for `std::find()` and `std::find_if()`.*
- struct `__for_each_selector`  
*`std::for_each()` selector.*
- struct `__generate_selector`  
*`std::generate()` selector.*
- struct `__generic_find_selector`  
*Base class of all `__gnu_parallel::__find_template` selectors.*
- struct `__generic_for_each_selector`  
*Generic `__selector` for embarrassingly parallel functions.*
- struct `__identity_selector`  
*Selector that just returns the passed iterator.*
- struct `__inner_product_selector`  
*`std::inner_product()` selector.*
- struct `__max_element_reduct`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__min_element_reduct`  
*Reduction for finding the maximum element, using a comparator.*
- struct `__mismatch_selector`  
*Test inverted predicate on a single element.*
- struct `__multiway_merge_3_variant_sentinel_switch`  
*Switch for 3-way merging with `__sentinels` turned off.*
- struct `__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 3-way merging with `__sentinels` turned on.*

- struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)  
*Switch for 4-way merging with `__sentinels` turned off.*
- struct [\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch](#)< true, [\\_RAIterIterator](#), [\\_RAIter3](#), [\\_DifferenceTp](#), [\\_Compare](#) >  
*Switch for 4-way merging with `__sentinels` turned on.*
- struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)  
*Switch for k-way merging with `__sentinels` turned on.*
- struct [\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch](#)< false, [\\_\\_stable](#), [\\_RAIterIterator](#), [\\_RAIter3](#), [\\_DifferenceTp](#), [\\_Compare](#) >  
*Switch for k-way merging with `__sentinels` turned off.*
- struct [\\_\\_replace\\_if\\_selector](#)  
*`std::replace()` selector.*
- struct [\\_\\_replace\\_selector](#)  
*`std::replace()` selector.*
- struct [\\_\\_transform1\\_selector](#)  
*`std::transform()` `__selector`, one input sequence variant.*
- struct [\\_\\_transform2\\_selector](#)  
*`std::transform()` `__selector`, two input sequences variant.*
- class [\\_\\_unary\\_negate](#)  
*Similar to [std::unary\\_negate](#), but giving the argument types explicitly.*
- struct [\\_DRandomShufflingGlobalData](#)  
*Data known to every thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).*
- struct [\\_DRSSorterPU](#)  
*Local data for a thread participating in [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\(\)](#).*
- struct [\\_DummyReduct](#)  
*Reduction function doing nothing.*
- class [\\_EqualFromLess](#)  
*Constructs predicate for equality from strict weak ordering predicate.*

- struct `_EqualTo`  
*Similar to `std::equal_to`, but allows two different types.*
- class `_GuardedIterator`  
*`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*
- class `_IteratorPair`  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class `_IteratorTriple`  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*
- struct `_Job`  
*One `__job` for a certain thread.*
- struct `_Less`  
*Similar to `std::less`, but allows two different types.*
- class `_Lexicographic`  
*Compare `__a` pair of types lexicographically, ascending.*
- class `_LexicographicReverse`  
*Compare `__a` pair of types lexicographically, descending.*
- class `_LoserTree`  
*Stable `_LoserTree` variant.*
- class `_LoserTree< false, _Tp, _Compare >`  
*Unstable `_LoserTree` variant.*
- class `_LoserTreeBase`  
*Guarded loser/tournament tree.*
- class `_LoserTreePointer`  
*Stable `_LoserTree` implementation.*
- class `_LoserTreePointer< false, _Tp, _Compare >`  
*Unstable `_LoserTree` implementation.*
- class `_LoserTreePointerBase`

Base class of [\\_Loser](#) Tree implementation using pointers.

- class [\\_LoserTreePointerUnguarded](#)  
Stable unguarded [\\_LoserTree](#) variant storing pointers.
- class [\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)  
Unstable unguarded [\\_LoserTree](#) variant storing pointers.
- class [\\_LoserTreePointerUnguardedBase](#)  
Unguarded loser tree, keeping only pointers to the elements in the tree structure.
- struct [\\_LoserTreeTraits](#)  
Traits for determining whether the loser tree should use pointers or copies.
- class [\\_LoserTreeUnguarded](#)  
Stable implementation of unguarded [\\_LoserTree](#).
- class [\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)  
Non-Stable implementation of unguarded [\\_LoserTree](#).
- class [\\_LoserTreeUnguardedBase](#)  
Base class for unguarded [\\_LoserTree](#) implementation.
- struct [\\_Multiplies](#)  
Similar to [std::multiplies](#), but allows two different types.
- struct [\\_Nothing](#)  
Functor doing nothing.
- struct [\\_Piece](#)  
Subsequence description.
- struct [\\_Plus](#)  
Similar to [std::plus](#), but allows two different types.
- struct [\\_PMWMSSortingData](#)  
Data accessed by all threads.
- class [\\_PseudoSequence](#)  
Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.
- class [\\_PseudoSequenceIterator](#)

*\_Iterator associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.*

- struct `_QSBThreadLocal`  
*Information local to one thread in the parallel quicksort run.*
- class `_RandomNumber`  
*Random number generator, based on the Mersenne twister.*
- class `_RestrictedBoundedConcurrentQueue`  
*Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.*
- struct `_SamplingSorter`  
*Stable sorting functor.*
- struct `_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`  
*Non-`__stable` sorting functor.*
- struct `_Settings`  
*class `_Settings` /// Run-time settings for the parallel mode including all tunable parameters.*
- struct `_SplitConsistently`  
*Split consistently.*
- struct `_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >`  
*Split by sampling.*
- struct `_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >`  
*Split by exact splitting.*
- struct `balanced_quicksort_tag`  
*Forces parallel sorting using balanced quicksort at compile time.*
- struct `balanced_tag`  
*Recommends parallel execution using dynamic load-balancing at compile time.*
- struct `constant_size_blocks_tag`  
*Selects the constant block size variant for `std::find()`.*

- struct [default\\_parallel\\_tag](#)  
*Recommends parallel execution using the default parallel algorithm.*
- struct [equal\\_split\\_tag](#)  
*Selects the equal splitting variant for `std::find()`.*
- struct [exact\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [find\\_tag](#)  
*Base class for `std::find()` variants.*
- struct [growing\\_blocks\\_tag](#)  
*Selects the growing block size variant for `std::find()`.*
- struct [multiway\\_mergesort\\_exact\\_tag](#)  
*Forces parallel sorting using multiway mergesort with exact splitting at compile time.*
- struct [multiway\\_mergesort\\_sampling\\_tag](#)  
*Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.*
- struct [multiway\\_mergesort\\_tag](#)  
*Forces parallel sorting using multiway mergesort at compile time.*
- struct [omp\\_loop\\_static\\_tag](#)  
*Recommends parallel execution using OpenMP static load-balancing at compile time.*
- struct [omp\\_loop\\_tag](#)  
*Recommends parallel execution using OpenMP dynamic load-balancing at compile time.*
- struct [parallel\\_tag](#)  
*Recommends parallel execution at compile time, optionally using a user-specified number of threads.*
- struct [quicksort\\_tag](#)  
*Forces parallel sorting using unbalanced quicksort at compile time.*
- struct [sampling\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*



- struct [sequential\\_tag](#)  
*Forces sequential execution at compile time.*
- struct [unbalanced\\_tag](#)  
*Recommends parallel execution using static load-balancing at compile time.*

## Typedefs

- typedef unsigned short [\\_BinIndex](#)
- typedef int64\_t [\\_CASable](#)
- typedef uint64\_t [\\_SequenceIndex](#)
- typedef uint16\_t [\\_ThreadIndex](#)

## Enumerations

- enum [\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum [\\_FindAlgorithm](#) { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum [\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_Parallelism](#) {  
    [sequential](#), [parallel\\_unbalanced](#), [parallel\\_balanced](#), [parallel\\_omp\\_loop](#),  
    [parallel\\_omp\\_loop\\_static](#), [parallel\\_taskqueue](#) }
- enum [\\_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [\\_SortAlgorithm](#) { **MWMS**, **QS**, **QS\_BALANCED** }
- enum [\\_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

## Functions

- template<typename [\\_RAIter](#), typename [\\_DifferenceTp](#)>  
void [\\_\\_calc\\_borders](#) ([\\_RAIter](#) \_\_elements, [\\_DifferenceTp](#) \_\_length, [\\_DifferenceTp](#) \*\_\_off)
- template<typename [\\_Tp](#)>  
bool [\\_\\_compare\\_and\\_swap](#) (volatile [\\_Tp](#) \*\_\_ptr, [\\_Tp](#) \_\_comparand, [\\_Tp](#) \_\_replacement)
- bool [\\_\\_compare\\_and\\_swap\\_32](#) (volatile int32\_t \*\_\_ptr, int32\_t \_\_comparand, int32\_t \_\_replacement)
- bool [\\_\\_compare\\_and\\_swap\\_64](#) (volatile int64\_t \*\_\_ptr, int64\_t \_\_comparand, int64\_t \_\_replacement)
- template<typename [\\_Iter](#), typename [\\_OutputIterator](#)>  
[\\_OutputIterator](#) [\\_\\_copy\\_tail](#) (std::pair< [\\_Iter](#), [\\_Iter](#) > \_\_b, std::pair< [\\_Iter](#), [\\_Iter](#) > \_\_e, [\\_OutputIterator](#) \_\_r)

- [void \\_\\_decode2](#) ([\\_CASable](#) \_\_x, int &\_\_a, int &\_\_b)
- [template](#)<typename [\\_RAIter](#) , typename [\\_DifferenceTp](#) >  
[void \\_\\_determine\\_samples](#) ([\\_PMWMSortingData](#)< [\\_RAIter](#) > \*\_\_sd, [\\_DifferenceTp](#) \_\_num\_samples)
- [\\_CASable \\_\\_encode2](#) (int \_\_a, int \_\_b)
- [template](#)<typename [\\_Tp](#) >  
[\\_Tp \\_\\_fetch\\_and\\_add](#) (volatile [\\_Tp](#) \*\_\_ptr, [\\_Tp](#) \_\_addend)
- [int32\\_t \\_\\_fetch\\_and\\_add\\_32](#) (volatile [int32\\_t](#) \*\_\_ptr, [int32\\_t](#) \_\_addend)
- [int64\\_t \\_\\_fetch\\_and\\_add\\_64](#) (volatile [int64\\_t](#) \*\_\_ptr, [int64\\_t](#) \_\_addend)
- [template](#)<typename [\\_RAIter1](#) , typename [\\_RAIter2](#) , typename [\\_Pred](#) , typename [\\_Selector](#) >  
[std::pair](#)< [\\_RAIter1](#) , [\\_RAIter2](#) > [\\_\\_find\\_template](#) ([\\_RAIter1](#) \_\_begin1, [\\_RAIter1](#) \_\_end1, [\\_RAIter2](#) \_\_begin2, [\\_Pred](#) \_\_pred, [\\_Selector](#) \_\_selector)
- [template](#)<typename [\\_RAIter1](#) , typename [\\_RAIter2](#) , typename [\\_Pred](#) , typename [\\_Selector](#) >  
[std::pair](#)< [\\_RAIter1](#) , [\\_RAIter2](#) > [\\_\\_find\\_template](#) ([\\_RAIter1](#) \_\_begin1, [\\_RAIter1](#) \_\_end1, [\\_RAIter2](#) \_\_begin2, [\\_Pred](#) \_\_pred, [\\_Selector](#) \_\_selector, [equal\\_split\\_tag](#))
- [template](#)<typename [\\_RAIter1](#) , typename [\\_RAIter2](#) , typename [\\_Pred](#) , typename [\\_Selector](#) >  
[std::pair](#)< [\\_RAIter1](#) , [\\_RAIter2](#) > [\\_\\_find\\_template](#) ([\\_RAIter1](#) \_\_begin1, [\\_RAIter1](#) \_\_end1, [\\_RAIter2](#) \_\_begin2, [\\_Pred](#) \_\_pred, [\\_Selector](#) \_\_selector, [growing\\_blocks\\_tag](#))
- [template](#)<typename [\\_RAIter1](#) , typename [\\_RAIter2](#) , typename [\\_Pred](#) , typename [\\_Selector](#) >  
[std::pair](#)< [\\_RAIter1](#) , [\\_RAIter2](#) > [\\_\\_find\\_template](#) ([\\_RAIter1](#) \_\_begin1, [\\_RAIter1](#) \_\_end1, [\\_RAIter2](#) \_\_begin2, [\\_Pred](#) \_\_pred, [\\_Selector](#) \_\_selector, [constant\\_size\\_blocks\\_tag](#))
- [template](#)<typename [\\_Iter](#) , typename [\\_UserOp](#) , typename [\\_Functionality](#) , typename [\\_Red](#) , typename [\\_Result](#) >  
[\\_UserOp](#) [\\_\\_for\\_each\\_template\\_random\\_access](#) ([\\_Iter](#) \_\_begin, [\\_Iter](#) \_\_end, [\\_UserOp](#) \_\_user\_op, [\\_Functionality](#) &\_\_functionality, [\\_Red](#) \_\_reduction, [\\_Result](#) \_\_reduction\_start, [\\_Result](#) &\_\_output, typename [std::iterator\\_traits](#)< [\\_Iter](#) >::difference\_type \_\_bound, [\\_Parallelism](#) \_\_parallelism\_tag)
- [template](#)<typename [\\_RAIter](#) , typename [\\_Op](#) , typename [\\_Fu](#) , typename [\\_Red](#) , typename [\\_Result](#) >  
[\\_Op](#) [\\_\\_for\\_each\\_template\\_random\\_access\\_ed](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_Op](#) \_\_o, [\\_Fu](#) &\_\_f, [\\_Red](#) \_\_r, [\\_Result](#) \_\_base, [\\_Result](#) &\_\_output, typename [std::iterator\\_traits](#)< [\\_RAIter](#) >::difference\_type \_\_bound)
- [template](#)<typename [\\_RAIter](#) , typename [\\_Op](#) , typename [\\_Fu](#) , typename [\\_Red](#) , typename [\\_Result](#) >  
[\\_Op](#) [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_Op](#) \_\_o, [\\_Fu](#) &\_\_f, [\\_Red](#) \_\_r, [\\_Result](#) \_\_base, [\\_Result](#) &\_\_output, typename [std::iterator\\_traits](#)< [\\_RAIter](#) >::difference\_type \_\_bound)
- [template](#)<typename [\\_RAIter](#) , typename [\\_Op](#) , typename [\\_Fu](#) , typename [\\_Red](#) , typename [\\_Result](#) >  
[\\_Op](#) [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\\_static](#) ([\\_RAIter](#) \_\_begin, [\\_RAIter](#) \_\_end, [\\_Op](#) \_\_o, [\\_Fu](#) &\_\_f, [\\_Red](#) \_\_r, [\\_Result](#) \_\_base, [\\_Result](#) &\_\_output, typename [std::iterator\\_traits](#)< [\\_RAIter](#) >::difference\_type \_\_bound)

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`>`  
`_Op __for_each_template_random_access_workstealing (_RAIter __begin, _`  
`RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__`  
`output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `__ThreadIndex __get_max_threads ()`
- `bool __is_parallel (const __Parallelism __p)`
- `template<typename _Iter, typename _Compare >`  
`bool __is_sorted (_Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter __median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c,`  
`_Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _`  
`RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp`  
`__max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_movc (_RAIter1 &__begin1, _RAIter1 _`  
`__end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _`  
`DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_usual (_RAIter1 &__begin1, _RAIter1 _`  
`__end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _`  
`DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __`  
`end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename`  
`std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare _`  
`__comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __`  
`end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename`  
`std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare _`  
`__comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end,`  
`_Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __`  
`end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum (_Iter __begin, _Iter __end, _`  
`OutputIterator __result, _BinaryOperation __bin_op)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end,`  
`_OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_  
traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _`  
`_OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_  
traits< _Iter >::difference_type __n)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator_traits< _RAIter >::difference_type \_\_parallel\_partition (_RAIter`  
`__begin, _RAIter __end, _Predicate __pred, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle (_RAIter __begin, _RAIter __end, _`  
`RandomNumberGenerator __rng=\_RandomNumber())`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs (_RAIter __begin, _RAIter __end, type-`  
`name std::iterator_traits< _RAIter >::difference_type __n, \_ThreadIndex __`  
`num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs\_pu (\_DRSSorterPU< _RAIter, _`  
`RandomNumberGenerator > *__pus)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_difference (_Iter __begin1, _Iter __end1, _`  
`_Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_intersection (_Iter __begin1, _Iter __end1, _`  
`_Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`  
`_OutputIterator \_\_parallel\_set\_operation (_Iter __begin1, _Iter __end1, _`  
`_Iter __begin2, _Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_symmetric\_difference (_Iter __begin1, _Iter`  
`__end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __`  
`comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_union (_Iter __begin1, _Iter __end1, _Iter _`  
`__begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`parallel\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`balanced\_quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp,`  
`multiway\_mergesort\_sampling\_tag __parallelism)`

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway\_mergesort\_exact\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default\_parallel\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort\_qs (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort\_qs\_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type \_\_parallel\_sort\_qs\_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_parallel\_sort\_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator \_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator \_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_qsb\_conquer (\_QSBThreadLocal< _RAIter > ** __tls, _RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __iam, \_ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type \_\_qsb\_divide (_RAIter __begin, _RAIter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_qsb\_local\_sort\_with\_helping (\_QSBThreadLocal< _RAIter > ** __tls, _Compare & __comp, \_ThreadIndex __iam, bool __wait)`

- `template<typename _RandomNumberGenerator >`  
`int \_\_random\_number\_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size >`  
`_Size \_\_rd\_log2 (_Size __n)`
- `template<typename _Tp >`  
`_Tp \_\_round\_up\_to\_pow2 (_Tp __x)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred >`  
`__RAIter1 \_\_search\_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_sequential\_multiway\_merge (_RAIterIterator __seqs_begin, __RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void \_\_sequential\_random\_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)`
- `template<typename _Iter >`  
`void \_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >`  
`void \_\_shrink\_and\_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `void \_\_yield ()`
- `template<typename _DifferenceType, typename _OutputIterator >`  
`_OutputIterator equally\_split (_DifferenceType __n, \_ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType >`  
`_DifferenceType equally\_split\_point (_DifferenceType __n, \_ThreadIndex __num_threads, \_ThreadIndex __thread_no)`
- `template<typename _Iter, typename _FunctorType >`  
`size_t list\_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)`
- `template<typename _Tp >`  
`const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare >`  
`void multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())`

- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`  
`_Tp multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _`  
`RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp`  
`>())`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp,`  
`typename _Compare >`  
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _`  
`RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length,`  
`_Compare __comp, __gnu_parallel::sampling_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator,`  
`typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _`  
`RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length,`  
`_Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator,`  
`typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _`  
`RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length,`  
`_Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _`  
`DifferenceType >`  
`void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _`  
`RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType`  
`__total_length, _Compare __comp, std::vector< std::pair< _DifferenceType,`  
`_DifferenceType > > *__pieces)`

- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway\_merge\_loser\_tree\_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >`  
`_RAIter3 multiway\_merge\_loser\_tree\_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >`  
`void multiway\_merge\_sampling\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut multiway\_merge\_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`



- `_RAIterOut` **multiway\_merge\_sentinels** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, `sampling_tag` \_\_tag)
- `template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Splitter , typename _Compare >`  
`_RAIter3` **parallel\_multiway\_merge** (`_RAIterIterator` \_\_seqs\_begin, `_RAIterIterator` \_\_seqs\_end, `_RAIter3` \_\_target, `_Splitter` \_\_splitter, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, `_ThreadIndex` \_\_num\_threads)
- `template<bool __stable, bool __exact, typename _RAIter , typename _Compare >`  
`void` **parallel\_sort\_mwms** (`_RAIter` \_\_begin, `_RAIter` \_\_end, `_Compare` \_\_comp, `_ThreadIndex` \_\_num\_threads)
- `template<bool __stable, bool __exact, typename _RAIter , typename _Compare >`  
`void` **parallel\_sort\_mwms\_pu** (`_PMWMSortingData< _RAIter > *` \_\_sd, `_Compare` &\_\_comp)
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, `parallel_tag` \_\_tag=`parallel_tag`(0))
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, `sampling_tag` \_\_tag)
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, `default_parallel_tag` \_\_tag)
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, `__gnu_parallel::sequential_tag`)
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, `__gnu_parallel::exact_tag` \_\_tag)
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >`  
`_RAIterOut` **stable\_multiway\_merge\_sentinels** (`_RAIterPairIterator` \_\_seqs\_begin, `_RAIterPairIterator` \_\_seqs\_end, `_RAIterOut` \_\_target, `_DifferenceTp` \_\_length, `_Compare` \_\_comp, `__gnu_parallel::exact_tag` \_\_tag)

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp _`  
`_length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp _`  
`_length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp _`  
`_length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_`  
`begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp _`  
`_length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`

## Variables

- static const int [\\_CASable\\_bits](#)
- static const [\\_CASable\\_CASable\\_mask](#)

### 4.6.1 Detailed Description

GNU parallel code for public use.

### 4.6.2 Typedef Documentation

#### 4.6.2.1 `typedef unsigned short __gnu_parallel::_BinIndex`

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

#### 4.6.2.2 `typedef int64_t __gnu_parallel::_CASable`

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file types.h.

#### 4.6.2.3 `typedef uint64_t __gnu_parallel::_SequenceIndex`

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file types.h.

#### 4.6.2.4 `typedef uint16_t __gnu_parallel::_ThreadIndex`

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file types.h.

### 4.6.3 Enumeration Type Documentation

#### 4.6.3.1 `enum __gnu_parallel::_AlgorithmStrategy`

Strategies for run-time algorithm selection:

Definition at line 67 of file types.h.

#### 4.6.3.2 `enum __gnu_parallel::_FindAlgorithm`

Find algorithms:

Definition at line 106 of file types.h.

#### 4.6.3.3 `enum __gnu_parallel::_MultiwayMergeAlgorithm`

Merging algorithms:

Definition at line 85 of file types.h.

#### 4.6.3.4 `enum __gnu_parallel::_Parallelism`

Run-time equivalents for the compile-time tags.

**Enumerator:**

*sequential* Not parallel.

*parallel\_unbalanced* Parallel unbalanced (equal-sized chunks).

*parallel\_balanced* Parallel balanced (work-stealing).  
*parallel\_omp\_loop* Parallel with OpenMP dynamic load-balancing.  
*parallel\_omp\_loop\_static* Parallel with OpenMP static load-balancing.  
*parallel\_taskqueue* Parallel with OpenMP taskqueue construct.

Definition at line 44 of file types.h.

#### 4.6.3.5 enum \_\_gnu\_parallel::\_PartialSumAlgorithm

Partial sum algorithms: recursive, linear.

Definition at line 91 of file types.h.

#### 4.6.3.6 enum \_\_gnu\_parallel::\_SortAlgorithm

Sorting algorithms:

Definition at line 76 of file types.h.

#### 4.6.3.7 enum \_\_gnu\_parallel::\_SplittingAlgorithm

Sorting/merging algorithms: sampling, \_\_exact.

Definition at line 98 of file types.h.

### 4.6.4 Function Documentation

**4.6.4.1** template<typename \_RAIter, typename \_DifferenceTp> void  
 \_\_gnu\_parallel::\_\_calc\_borders ( \_RAIter \_\_elements, \_DifferenceTp  
 \_\_length, \_DifferenceTp \* \_\_off )

Precalculate \_\_advances for Knuth-Morris-Pratt algorithm.

#### Parameters

*\_\_elements* Begin iterator of sequence to search for.  
*\_\_length* Length of sequence to search for.  
*\_\_advances* Returned \_\_offsets.

Definition at line 51 of file search.h.

Referenced by \_\_search\_template().

**4.6.4.2** `template<typename _Tp> bool __gnu_parallel::__compare_and_swap  
( volatile _Tp * __ptr, _Tp __comparand, _Tp __replacement )  
[inline]`

Compare \*\_\_ptr and \_\_comparand. If equal, let \*\_\_ptr=\_\_replacement and return true, return false otherwise.

Implementation is heavily platform-dependent.

#### Parameters

**\_\_ptr** Pointer to signed integer.

**\_\_comparand** Compare value.

**\_\_replacement** Replacement value.

Definition at line 337 of file parallel/compatibility.h.

References \_\_compare\_and\_swap\_32(), and \_\_compare\_and\_swap\_64().

Referenced by \_\_parallel\_partition(), \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue< \_Piece >::pop\_back(), and \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue< \_Piece >::pop\_front().

**4.6.4.3** `bool __gnu_parallel::__compare_and_swap_32 ( volatile int32_t *  
__ptr, int32_t __comparand, int32_t __replacement ) [inline]`

Compare \*\_\_ptr and \_\_comparand. If equal, let \*\_\_ptr=\_\_replacement and return true, return false otherwise.

Implementation is heavily platform-dependent.

#### Parameters

**\_\_ptr** Pointer to 32-bit signed integer.

**\_\_comparand** Compare value.

**\_\_replacement** Replacement value.

Definition at line 240 of file parallel/compatibility.h.

Referenced by \_\_compare\_and\_swap().

**4.6.4.4** `bool __gnu_parallel::__compare_and_swap_64 ( volatile int64_t *  
__ptr, int64_t __comparand, int64_t __replacement ) [inline]`

Compare \*\_\_ptr and \_\_comparand. If equal, let \*\_\_ptr=\_\_replacement and return true, return false otherwise.

Implementation is heavily platform-dependent.

**Parameters**

**\_\_ptr** Pointer to 64-bit signed integer.

**\_\_comparand** Compare value.

**\_\_replacement** Replacement value.

Definition at line 283 of file parallel/compatibility.h.

Referenced by `__compare_and_swap()`.

**4.6.4.5** `void __gnu_parallel::__decode2 ( _CASable __x, int & __a, int & __b ) [inline]`

Decode two integers from one `gnu_parallel::_CASable`.

**Parameters**

**\_\_x** [\\_\\_gnu\\_parallel::\\_CASable](#) to decode integers from.

**\_\_a** First integer, to be decoded from the most-significant `_CASable_bits/2` bits of `__x`.

**\_\_b** Second integer, to be encoded in the least-significant `_CASable_bits/2` bits of `__x`.

**See also**

[\\_\\_encode2](#)

Definition at line 133 of file parallel/base.h.

References `_CASable_bits`, and `_CASable_mask`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_back()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_front()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::push_front()`.

**4.6.4.6** `template<typename _RAIter, typename _DifferenceTp> void __gnu_parallel::__determine_samples ( _PMWMSSortingData< _RAIter> * __sd, _DifferenceTp __num_samples )`

Select `_M_samples` from a sequence.

**Parameters**

**\_\_sd** Pointer to algorithm data. `_Result` will be placed in `__sd->_M_samples`.

**\_\_num\_samples** Number of `_M_samples` to select.

Definition at line 97 of file multiway\_mergesort.h.

References `__gnu_parallel::PMWMSSortingData< _RAIter >::_M_samples`, `__gnu_parallel::PMWMSSortingData< _RAIter >::_M_source`, `__gnu_parallel::PMWMSSortingData< _RAIter >::_M_starts`, and `equally_split()`.

#### 4.6.4.7 `_CASable __gnu_parallel::_encode2 ( int __a, int __b ) [inline]`

Encode two integers into one `gnu_parallel::_CASable`.

##### Parameters

- `__a` First integer, to be encoded in the most-significant `_CASable_bits/2` bits.
- `__b` Second integer, to be encoded in the least-significant `_CASable_bits/2` bits.

##### Returns

value encoding `__a` and `__b`.

##### See also

[\\_\\_decode2](#)

Definition at line 119 of file parallel/base.h.

References `_CASable_bits`.

Referenced by `__gnu_parallel::RestrictedBoundedConcurrentQueue< _Piece >::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::RestrictedBoundedConcurrentQueue< _Piece >::pop_back()`, `__gnu_parallel::RestrictedBoundedConcurrentQueue< _Piece >::pop_front()`, and `__gnu_parallel::RestrictedBoundedConcurrentQueue< _Piece >::push_front()`.

#### 4.6.4.8 `template<typename _Tp> _Tp __gnu_parallel::_fetch_and_add ( volatile _Tp * __ptr, _Tp __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

##### Parameters

- `__ptr` Pointer to a signed integer.
- `__addend` Value to add.

Definition at line 186 of file parallel/compatibility.h.

References `__fetch_and_add_32()`, and `__fetch_and_add_64()`.

Referenced by `__parallel_partition()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Piece>::push_front()`.

**4.6.4.9** `int32_t __gnu_parallel::__fetch_and_add_32 ( volatile int32_t * __ptr, int32_t __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

#### Parameters

`__ptr` Pointer to a 32-bit signed integer.

`__addend` Value to add.

Definition at line 95 of file `parallel/compatibility.h`.

Referenced by `__fetch_and_add()`.

**4.6.4.10** `int64_t __gnu_parallel::__fetch_and_add_64 ( volatile int64_t * __ptr, int64_t __addend ) [inline]`

Add a value to a variable, atomically.

Implementation is heavily platform-dependent.

#### Parameters

`__ptr` Pointer to a 64-bit signed integer.

`__addend` Value to add.

Definition at line 134 of file `parallel/compatibility.h`.

Referenced by `__fetch_and_add()`.

**4.6.4.11** `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector ) [inline]`

Parallel `std::find`, switch for different algorithms.

#### Parameters

`__begin1` Begin iterator of first sequence.



`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence. Must have same length as first sequence.  
`__pred` Find predicate.  
`__selector` \_Functionality (e. g. `std::find_if()`, `std::equal()`,...)

**Returns**

Place of finding in both sequences.

Definition at line 60 of file `find.h`.

References `__gnu_parallel::_Settings::get()`, and `std::make_pair()`.

**4.6.4.12** `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag )`

Parallel `std::find`, equal splitting variant.

**Parameters**

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence. Second \_\_sequence must have same length as first sequence.  
`__pred` Find predicate.  
`__selector` \_Functionality (e. g. `std::find_if()`, `std::equal()`,...)

**Returns**

Place of finding in both sequences.

Definition at line 97 of file `find.h`.

References `_GLIBCXX_CALL`, and `equally_split()`.

**4.6.4.13** `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag )`

Parallel `std::find`, growing block size variant.

**Parameters**

**\_\_begin1** Begin iterator of first sequence.  
**\_\_end1** End iterator of first sequence.  
**\_\_begin2** Begin iterator of second sequence. Second \_\_sequence must have same length as first sequence.  
**\_\_pred** Find predicate.  
**\_\_selector** \_Functionality (e. g. std::find\_if(), std::equal(),...)

**Returns**

Place of finding in both sequences.

**See also**

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)  
[\\_\\_gnu\\_parallel::\\_Settings::find\\_scale\\_factor](#)

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_scale_factor`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.14** `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair<_RAIter1, _RAIter2> __gnu_parallel::_find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag )`

Parallel std::find, constant block size variant.

**Parameters**

**\_\_begin1** Begin iterator of first sequence.  
**\_\_end1** End iterator of first sequence.  
**\_\_begin2** Begin iterator of second sequence. Second \_\_sequence must have same length as first sequence.  
**\_\_pred** Find predicate.  
**\_\_selector** \_Functionality (e. g. std::find\_if(), std::equal(),...)

**Returns**

Place of finding in both sequences.

**See also**

[\\_\\_gnu\\_parallel::\\_Settings::find\\_sequential\\_search\\_size](#)

[\\_\\_gnu\\_parallel::\\_Settings::find\\_block\\_size](#) There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

**4.6.4.15** `template<typename _Iter , typename _UserOp , typename _Functionality , typename _Red , typename _Result > _UserOp  
__gnu_parallel::_for_each_template_random_access ( _Iter  
__begin, _Iter __end, _UserOp __user_op, _Functionality &  
__functionality, _Red __reduction, _Result __reduction_start, _Result  
& __output, typename std::iterator_traits<_Iter >::difference_type  
__bound, _Parallelism __parallelism_tag )`

Chose the desired algorithm by evaluating `__parallelism_tag`.

**Parameters**

`__begin` Begin iterator of input sequence.

`__end` End iterator of input sequence.

`__user_op` A user-specified functor (comparator, predicate, associative operator,...)

`__functionality` functor to *process* an element with `__user_op` (depends on desired functionality, e. g. accumulate, for\_each,...)

`__reduction` Reduction functor.

`__reduction_start` Initial value for reduction.

`__output` Output iterator.

`__bound` Maximum number of elements processed.

`__parallelism_tag` Parallelization method

Definition at line 61 of file for\_each.h.

References `__for_each_template_random_access_ed()`, `__for_each_template_random_access_omp_loop()`, `__for_each_template_random_access_workstealing()`, `parallel_omp_loop`, `parallel_omp_loop_static`, and `parallel_unbalanced`.

**4.6.4.16** `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op  
__gnu_parallel::__for_each_template_random_access_ed ( _RAIter  
__begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result  
__base, _Result & __output, typename std::iterator_traits< _RAIter  
>::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

#### Parameters

`__begin` Begin iterator of element sequence.  
`__end` End iterator of element sequence.  
`__o` User-supplied functor (comparator, predicate, adding functor, ...)  
`__f` Functor to "process" an element with `__op` (depends on desired functionality, e. g. for `std::for_each()`, ...).  
`__r` Functor to "add" a single `__result` to the already processed elements (depends on functionality).  
`__base` Base value for reduction.  
`__output` Pointer to position where final result is written to  
`__bound` Maximum number of elements processed (e. g. for `std::count_n()`).

#### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `par_loop.h`.

References `equally_split_point()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.17** `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op  
__gnu_parallel::__for_each_template_random_access_omp_loop (   
_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,   
_Result __base, _Result & __output, typename std::iterator_traits<   
_RAIter >::difference_type __bound )`

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

**Parameters**

- \_\_begin** Begin iterator of element sequence.
- \_\_end** End iterator of element sequence.
- \_\_o** User-supplied functor (comparator, predicate, adding functor, etc.).
- \_\_f** Functor to *process* an element with **\_\_op** (depends on desired functionality, e. g. for `std::for_each()`, ...).
- \_\_r** Functor to *add* a single **\_\_result** to the already processed elements (depends on functionality).
- \_\_base** Base value for reduction.
- \_\_output** Pointer to position where final result is written to
- \_\_bound** Maximum number of elements processed (e. g. for `std::count_n()`).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `omp_loop.h`.

Referenced by `__for_each_template_random_access()`.

```
4.6.4.18 template<typename _RAIter, typename _Op, typename _Fu,
 typename _Red, typename _Result> _Op __gnu_parallel::__-
 for_each_template_random_access_omp_loop_static (_RAIter
 __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result
 __base, _Result & __output, typename std::iterator_traits< _RAIter
 >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

**Parameters**

- \_\_begin** Begin iterator of element sequence.
- \_\_end** End iterator of element sequence.
- \_\_o** User-supplied functor (comparator, predicate, adding functor, ...).
- \_\_f** Functor to *process* an element with **\_\_op** (depends on desired functionality, e. g. for `std::for_each()`, ...).
- \_\_r** Functor to *add* a single **\_\_result** to the already processed **\_\_elements** (depends on functionality).
- \_\_base** Base value for reduction.
- \_\_output** Pointer to position where final result is written to
- \_\_bound** Maximum number of elements processed (e. g. for `std::count_n()`).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file `omp_loop_static.h`.

**4.6.4.19** `template<typename _RAIter, typename _Op, typename  
_Fu, typename _Red, typename _Result > _Op  
__gnu_parallel::__for_each_template_random_access_workstealing (  
_RAIter __begin, _RAIter __end, _Op __op, _Fu & __f, _Red __r,  
_Result __base, _Result & __output, typename std::iterator_traits<  
_RAIter >::difference_type __bound )`

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

**Parameters**

**\_\_begin** Begin iterator of element sequence.

**\_\_end** End iterator of element sequence.

**\_\_op** User-supplied functor (comparator, predicate, adding functor, ...).

**\_\_f** Functor to *process* an element with **\_\_op** (depends on desired functionality, e. g. for `std::for_each()`, ...).

**\_\_r** Functor to *add* a single **\_\_result** to the already processed elements (depends on functionality).

**\_\_base** Base value for reduction.

**\_\_output** Pointer to position where final result is written to

**\_\_bound** Maximum number of elements processed (e. g. for `std::count_n()`).

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file `workstealing.h`.

References `__yield()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Job< _DifferenceTp >::_M_first`, `__gnu_parallel::_Job< _DifferenceTp >::_M_last`, `__gnu_parallel::_Job< _DifferenceTp >::_M_load`, `__gnu_parallel::_Settings::cache_line_size`, `__gnu_parallel::_Settings::get()`, and `min()`.

Referenced by `__for_each_template_random_access()`.

**4.6.4.20** `template<typename _Iter , typename _Compare > bool  
__gnu_parallel::__is_sorted ( _Iter __begin, _Iter __end,  
_Compare __comp )`

Check whether [`__begin`, `__end`) is sorted according to `__comp`.

#### Parameters

`__begin` Begin iterator of sequence.

`__end` End iterator of sequence.

`__comp` Comparator.

#### Returns

`true` if sorted, `false` otherwise.

Definition at line 51 of file `checkers.h`.

Referenced by `__sequential_multiway_merge()`, `multiway_merge_loser_tree_sentinel()`, and `parallel_multiway_merge()`.

**4.6.4.21** `template<typename _RAIter , typename _Compare > _RAIter  
__gnu_parallel::__median_of_three_iterators ( _RAIter __a, _RAIter  
__b, _RAIter __c, _Compare __comp )`

Compute the median of three referenced elements, according to `__comp`.

#### Parameters

`__a` First iterator.

`__b` Second iterator.

`__c` Third iterator.

`__comp` Comparator.

Definition at line 398 of file `parallel/base.h`.

Referenced by `__qsb_divide()`.

**4.6.4.22** `template<typename _RAIter1 , typename _RAIter2 , typename  
_OutputIterator , typename _DifferenceTp , typename _Compare  
> _OutputIterator __gnu_parallel::__merge_advance ( _RAIter1  
& __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
__end2, _OutputIterator __target, _DifferenceTp __max_length,  
_Compare __comp ) [inline]`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.  
`__target` Target begin iterator.  
`__max_length` Maximum number of elements to merge.  
`__comp` Comparator.

#### Returns

Output end iterator.

Definition at line 171 of file `merge.h`.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`, and `__sequential_multiway_merge()`.

**4.6.4.23** `template<typename _RAIter1, typename _RAIter2, typename  
 _OutputIterator, typename _DifferenceTp, typename _Compare >  
 _OutputIterator __gnu_parallel::__merge_advance_movc ( _RAIter1  
 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
 __end2, _OutputIterator __target, _DifferenceTp __max_length,  
 _Compare __comp )`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.  
`__target` Target begin iterator.  
`__max_length` Maximum number of elements to merge.



`__comp` Comparator.

#### Returns

Output end iterator.

Definition at line 105 of file `merge.h`.

Referenced by `__merge_advance()`.

**4.6.4.24** `template<typename _RAIter1, typename _RAIter2, typename  
_OutputIterator, typename _DifferenceTp, typename _Compare >  
_OutputIterator __gnu_parallel::__merge_advance_usual ( _RAIter1  
& __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2  
__end2, _OutputIterator __target, _DifferenceTp __max_length,  
_Compare __comp )`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

#### Parameters

`__begin1` Begin iterator of first sequence.

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence.

`__end2` End iterator of second sequence.

`__target` Target begin iterator.

`__max_length` Maximum number of elements to merge.

`__comp` Comparator.

#### Returns

Output end iterator.

Definition at line 57 of file `merge.h`.

**4.6.4.25** `template<typename _RAIter1, typename _RAIter2,  
typename _RAIter3, typename _Compare > _RAIter3  
__gnu_parallel::__parallel_merge_advance ( _RAIter1 & __begin1,  
_RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2,  
_RAIter3 __target, typename std::iterator_traits< _RAIter1  
>::difference_type __max_length, _Compare __comp ) [inline]`

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

**Parameters**

**\_\_begin1** Begin iterator of first sequence.  
**\_\_end1** End iterator of first sequence.  
**\_\_begin2** Begin iterator of second sequence.  
**\_\_end2** End iterator of second sequence.  
**\_\_target** Target begin iterator.  
**\_\_max\_length** Maximum number of elements to merge.  
**\_\_comp** Comparator.

**Returns**

Output end iterator.

Definition at line 195 of file merge.h.

References `__merge_advance()`.

**4.6.4.26** `template<typename _RAIter1 , typename _RAIter3 , typename  
 _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance (   
 _RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2,  
 _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits<  
 _RAIter1 >::difference_type __max_length, _Compare __comp )  
 [inline]`

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

**Parameters**

**\_\_begin1** Begin iterator of first sequence.  
**\_\_end1** End iterator of first sequence.  
**\_\_begin2** Begin iterator of second sequence.  
**\_\_end2** End iterator of second sequence.  
**\_\_target** Target begin iterator.  
**\_\_max\_length** Maximum number of elements to merge.  
**\_\_comp** Comparator.

**Returns**

Output end iterator.

Definition at line 223 of file `merge.h`.

References `std::make_pair()`, `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

**4.6.4.27** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__parallel_nth_element ( _RAIter __begin, _RAIter  
__nth, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::nth_element()`.

#### Parameters

***\_\_begin*** Begin iterator of input sequence.

***\_\_nth*** Iterator of element that must be in position afterwards.

***\_\_end*** End iterator of input sequence.

***\_\_comp*** Comparator.

Definition at line 332 of file `partition.h`.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, `max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, `__gnu_parallel::_Settings::partition_minimal_n`, and `std::swap()`.

Referenced by `__parallel_partial_sort()`.

**4.6.4.28** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__parallel_partial_sort ( _RAIter __begin, _RAIter  
__middle, _RAIter __end, _Compare __comp )`

Parallel implementation of `std::partial_sort()`.

#### Parameters

***\_\_begin*** Begin iterator of input sequence.

***\_\_middle*** Sort until this position.

***\_\_end*** End iterator of input sequence.

***\_\_comp*** Comparator.

Definition at line 422 of file `partition.h`.

References `__parallel_nth_element()`.

**4.6.4.29** `template<typename _Iter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::__parallel_partial_sum ( _Iter __begin, _Iter  
__end, _OutputIterator __result, _BinaryOperation __bin_op )`

Parallel partial sum front-\_\_end.

#### Parameters

**\_\_begin** Begin iterator of input sequence.  
**\_\_end** End iterator of input sequence.  
**\_\_result** Begin iterator of output sequence.  
**\_\_bin\_op** Associative binary function.

#### Returns

End iterator of output sequence.

Definition at line 203 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::__Settings::get()`.

**4.6.4.30** `template<typename _Iter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::__parallel_partial_sum_basecase ( _Iter __begin,  
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iter >::value_type __value )`

Base case prefix sum routine.

#### Parameters

**\_\_begin** Begin iterator of input sequence.  
**\_\_end** End iterator of input sequence.  
**\_\_result** Begin iterator of output sequence.  
**\_\_bin\_op** Associative binary function.  
**\_\_value** Start value. Must be passed since the neutral element is unknown in general.

#### Returns

End iterator of output sequence.

Definition at line 58 of file `partial_sum.h`.

Referenced by `__parallel_partial_sum_linear()`.

**4.6.4.31** `template<typename _Iter , typename _OutputIterator  
, typename _BinaryOperation > _OutputIterator  
__gnu_parallel::__parallel_partial_sum_linear ( _Iter __begin,  
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
typename std::iterator_traits< _Iter >::difference_type __n )`

Parallel partial sum implementation, two-phase approach, no recursion.

#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator of input sequence.  
`__result` Begin iterator of output sequence.  
`__bin_op` Associative binary function.  
`__n` Length of sequence.  
`__num_threads` Number of threads to use.

#### Returns

End iterator of output sequence.

Definition at line 90 of file `partial_sum.h`.

References `__parallel_partial_sum_basecase()`, `equally_split()`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::partial_sum_dilation`.

Referenced by `__parallel_partial_sum()`.

**4.6.4.32** `template<typename _RAIter , typename _Predicate  
> std::iterator_traits<_RAIter>::difference_type  
__gnu_parallel::__parallel_partition ( _RAIter __begin, _RAIter  
__end, _Predicate __pred, _ThreadIndex __num_threads )`

Parallel implementation of `std::partition`.

#### Parameters

`__begin` Begin iterator of input sequence to split.  
`__end` End iterator of input sequence to split.  
`__pred` Partition predicate, possibly including some kind of pivot.  
`__num_threads` Maximum number of threads to use for this task.

#### Returns

Number of elements not fulfilling the predicate.

Definition at line 56 of file partition.h.

References `__compare_and_swap()`, `__fetch_and_add()`, `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::partition_chunk_share`, `__gnu_parallel::_Settings::partition_chunk_size`, and `std::swap()`.

Referenced by `__parallel_nth_element()`, `__parallel_sort_qs_divide()`, and `__qsb_divide()`.

```
4.6.4.33 template<typename _RAIter, typename _RandomNumberGenerator
> void __gnu_parallel::__parallel_random_shuffle (_RAIter
__begin, _RAIter __end, _RandomNumberGenerator __rng =
__RandomNumber()) [inline]
```

Parallel random public call.

#### Parameters

`__begin` Begin iterator of sequence.  
`__end` End iterator of sequence.  
`__rng` Random number generator to use.

Definition at line 517 of file random\_shuffle.h.

References `__parallel_random_shuffle_drs()`.

```
4.6.4.34 template<typename _RAIter, typename _RandomNumberGenerator
> void __gnu_parallel::__parallel_random_shuffle_drs (_RAIter
__begin, _RAIter __end, typename std::iterator_traits<
_RAIter >::difference_type __n, _ThreadIndex __num_threads,
_RandomNumberGenerator & __rng)
```

Main parallel random shuffle step.

#### Parameters

`__begin` Begin iterator of sequence.  
`__end` End iterator of sequence.  
`__n` Length of sequence.  
`__num_threads` Number of threads to use.  
`__rng` Random number generator to use.

Definition at line 263 of file random\_shuffle.h.

References \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_bins\_end, \_\_parallel\_random\_shuffle\_drs\_pu(), \_\_rd\_log2(), \_\_round\_up\_to\_pow2(), \_\_sequential\_random\_shuffle(), \_GLIBCXX\_CALL, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_bin\_proc, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_bins\_begin, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_dist, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bins, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bits, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_num\_threads, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_sd, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_seed, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_starts, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_temporaries, \_\_gnu\_parallel::\_Settings::L2\_cache\_size, std::min(), and \_\_gnu\_parallel::\_Settings::TLB\_size.

Referenced by \_\_parallel\_random\_shuffle().

**4.6.4.35** `template<typename _RAIter, typename _RandomNumberGenerator  
> void __gnu_parallel::_parallel_random_shuffle_drs_pu (  
_DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus )`

Random shuffle code executed by each thread.

#### Parameters

*\_\_pus* Array of thread-local data records.

Definition at line 122 of file random\_shuffle.h.

References \_\_random\_number\_pow2(), \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_dist, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bins, \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_num\_bits, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_num\_threads, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_sd, \_\_gnu\_parallel::\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator >::\_M\_seed, and \_\_gnu\_parallel::\_DRandomShufflingGlobalData< \_RAIter >::\_M\_starts.

Referenced by \_\_parallel\_random\_shuffle\_drs().

**4.6.4.36** `template<bool __stable, typename _RAIter, typename _Compare >  
void __gnu_parallel::_parallel_sort ( _RAIter __begin, _RAIter  
__end, _Compare __comp, parallel_tag __parallelism ) [inline]`

Choose a parallel sorting algorithm.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.  
***stable*** Sort `stable`.

References `__gnu_parallel::parallel_tag::get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `GLIBCXX_CALL`, and `__gnu_parallel::Settings::get()`.

[illegible]

Choose balanced quicksort for parallel sorting.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.  
***\_\_stable*** Sort *\_\_stable*.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qsb()`, and `GLIBCXX_CALL`.



[illegible]

Choose multiway mergesort with splitting by sampling, for parallel sorting.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

```
graph LR; A["_gnu_parallel::__parallel_sort"] --> B["_gnu_parallel::parallel_tag::__get_num_threads"]
```

**4.6.4.39** `template<bool __stable, typename _RAIter, typename _Compare >  
 void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, multiway_mergesort_tag __parallelism )  
 [inline]`

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

#### Parameters

**\_\_begin** Begin iterator of input sequence.

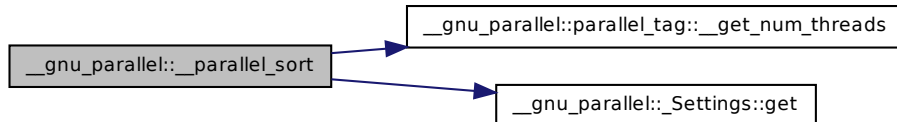
**\_\_end** End iterator of input sequence.

**\_\_comp** Comparator.

Definition at line 74 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `_GLIBCXX_CALL`,  
 and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



**4.6.4.40** `template<bool __stable, typename _RAIter, typename _Compare >  
 void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, multiway_mergesort_exact_tag  
 __parallelism ) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

#### Parameters

**\_\_begin** Begin iterator of input sequence.

**\_\_end** End iterator of input sequence.

**\_\_comp** Comparator.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

```
__gnu_parallel::__parallel_sort → __gnu_parallel::parallel_tag::__get_num_threads
```

Choose quicksort for parallel sorting.

***\_\_begin*** Begin iterator of input sequence.  
***\_\_end*** End iterator of input sequence.  
***\_\_comp*** Comparator.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, and `GLIBCXX CALL`.

```

graph LR
 parallel_sort[parallel_sort] --> parallel_sort_0[parallel_sort_0]
 parallel_sort_0 --> parallel_sort_0_comparator[parallel_sort_0_comparator]
 parallel_sort_0_comparator --> parallel_sort_0_divide[parallel_sort_0_divide]
 parallel_sort_0_divide --> parallel_sort_0_merge[parallel_sort_0_merge]
 parallel_sort_0_merge --> parallel_partition[parallel_partition]
 parallel_partition --> compare_and_swap[compare_and_swap]
 parallel_partition --> fetch_and_add_32[fetch_and_add_32]
 parallel_partition --> fetch_and_add_64[fetch_and_add_64]
 compare_and_swap --> compare_and_swap_2[compare_and_swap_2]
 compare_and_swap_2 --> compare_and_swap_3[compare_and_swap_3]
 fetch_and_add_32 --> fetch_and_add_32_2[fetch_and_add_32_2]
 fetch_and_add_64 --> fetch_and_add_64_2[fetch_and_add_64_2]

```

**4.6.4.42** `template<bool __stable, typename _RAIter, typename _Compare >  
 void __gnu_parallel::__parallel_sort ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, default_parallel_tag __parallelism )  
 [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

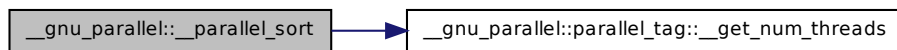
#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator of input sequence.  
`__comp` Comparator.

Definition at line 178 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**4.6.4.43** `template<typename _RAIter, typename _Compare > void  
 __gnu_parallel::__parallel_sort_qs ( _RAIter __begin, _RAIter  
 __end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort main call.

#### Parameters

`__begin` Begin iterator of input sequence.  
`__end` End iterator input sequence, ignored.  
`__comp` Comparator.  
`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 152 of file `quicksort.h`.

References `__parallel_sort_qs_conquer()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.

**4.6.4.44** `template<typename _RAIter, typename _Compare > void  
 __gnu_parallel::__parallel_sort_qs_conquer ( _RAIter __begin,  
 _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

Unbalanced quicksort conquer step.

#### Parameters

`__begin` Begin iterator of subsequence.

`__end` End iterator of subsequence.

`__comp` Comparator.

`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 99 of file quicksort.h.

References `__parallel_sort_qs_divide()`, and `__gnu_parallel::_Settings::get()`.

Referenced by `__parallel_sort_qs()`.

**4.6.4.45** `template<typename _RAIter, typename _Compare  
 > std::iterator_traits<_RAIter>::difference_type  
 __gnu_parallel::__parallel_sort_qs_divide ( _RAIter __begin,  
 _RAIter __end, _Compare __comp, typename std::iterator_traits<  
 _RAIter >::difference_type __pivot_rank, typename  
 std::iterator_traits<_RAIter >::difference_type __num_samples,  
 _ThreadIndex __num_threads )`

Unbalanced quicksort divide step.

#### Parameters

`__begin` Begin iterator of subsequence.

`__end` End iterator of subsequence.

`__comp` Comparator.

`__pivot_rank` Desired \_\_rank of the pivot.

`__num_samples` Choose pivot from that many samples.

`__num_threads` Number of threads that are allowed to work on this part.

Definition at line 51 of file quicksort.h.

References `__parallel_partition()`, and `min()`.

Referenced by `__parallel_sort_qs_conquer()`.

**4.6.4.46** `template<typename _RAIter, typename _Compare> void  
     __gnu_parallel::__parallel_sort_qsb ( _RAIter __begin, _RAIter  
     __end, _Compare __comp, _ThreadIndex __num_threads )`

Top-level quicksort routine.

#### Parameters

***\_\_begin*** Begin iterator of sequence.  
     ***\_\_end*** End iterator of sequence.  
     ***\_\_comp*** Comparator.  
     ***\_\_num\_threads*** Number of threads that are allowed to work on this part.

Definition at line 430 of file `balanced_quicksort.h`.

References `__qsb_conquer()`, `__rd_log2()`, `_GLIBCXX_CALL`, and `std::make_pair()`.

Referenced by `__parallel_sort()`.

**4.6.4.47** `template<typename _IIter, class _OutputIterator> _OutputIterator  
     __gnu_parallel::__parallel_unique_copy ( _IIter __first, _IIter  
     __last, _OutputIterator __result ) [inline]`

Parallel `std::unique_copy()`, without explicit equality predicate.

#### Parameters

***\_\_first*** Begin iterator of input sequence.  
     ***\_\_last*** End iterator of input sequence.  
     ***\_\_result*** Begin iterator of result \_\_sequence.

#### Returns

End iterator of result \_\_sequence.

Definition at line 186 of file `unique_copy.h`.

References `__parallel_unique_copy()`.

**4.6.4.48** `template<typename _IIter, class _OutputIterator, class  
     _BinaryPredicate> _OutputIterator __gnu_parallel::__parallel_  
     unique_copy ( _IIter __first, _IIter __last, _OutputIterator __result,  
     _BinaryPredicate __binary_pred )`

Parallel `std::unique_copy()`, w/ \_\_o explicit equality predicate.

**Parameters**

- \_\_first* Begin iterator of input sequence.
- \_\_last* End iterator of input sequence.
- \_\_result* Begin iterator of result *\_\_sequence*.
- \_\_binary\_pred* Equality predicate.

**Returns**

End iterator of result *\_\_sequence*.

Definition at line 50 of file `unique_copy.h`.

References `_GLIBCXX_CALL`, and `equally_split()`.

Referenced by `__parallel_unique_copy()`.

**4.6.4.49** `template<typename _RAIter, typename _Compare> void  
 __gnu_parallel::__qsb_conquer ( _QSBThreadLocal< _RAIter >  
 ** __tls, _RAIter __begin, _RAIter __end, _Compare __comp,  
 _ThreadIndex __iam, _ThreadIndex __num_threads, bool  
 __parent_wait )`

Quicksort conquer step.

**Parameters**

- \_\_tls* Array of thread-local storages.
- \_\_begin* Begin iterator of subsequence.
- \_\_end* End iterator of subsequence.
- \_\_comp* Comparator.
- \_\_iam* Number of the thread processing this function.
- \_\_num\_threads* Number of threads that are allowed to work on this part.

Definition at line 171 of file `balanced_quicksort.h`.

References `__qsb_divide()`, `__qsb_local_sort_with_helping()`, `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover`, and `__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial`.

Referenced by `__parallel_sort_qsb()`.

**4.6.4.50** `template<typename _RAIter, typename _Compare  
> std::iterator_traits<_RAIter>::difference_type  
__gnu_parallel::__qsb_divide ( _RAIter __begin, _RAIter __end,  
_Compare __comp, _ThreadIndex __num_threads )`

Balanced quicksort divide step.

#### Parameters

`__begin` Begin iterator of subsequence.  
`__end` End iterator of subsequence.  
`__comp` Comparator.  
`__num_threads` Number of threads that are allowed to work on this part.

#### Precondition

`(__end-__begin)>=1`

Definition at line 100 of file `balanced_quicksort.h`.

References `__median_of_three_iterators()`, `__parallel_partition()`, and `std::swap()`.

Referenced by `__qsb_conquer()`.

**4.6.4.51** `template<typename _RAIter, typename _Compare> void  
__gnu_parallel::__qsb_local_sort_with_helping ( _QSBThreadLocal<  
_RAIter> ** __tls, _Compare & __comp, _ThreadIndex __iam,  
bool __wait )`

Quicksort step doing load-balanced local sort.

#### Parameters

`__tls` Array of thread-local storages.  
`__comp` Comparator.  
`__iam` Number of the thread processing this function.

Definition at line 247 of file `balanced_quicksort.h`.

References `__yield()`, `_GLIBCXX_ASSERTIONS`, `__gnu_parallel::_QSBThreadLocal<_RAIter>::M_elements_leftover`, `__gnu_parallel::_QSBThreadLocal<_RAIter>::M_initial`, `__gnu_parallel::_QSBThreadLocal<_RAIter>::M_leftover_parts`, `__gnu_parallel::_QSBThreadLocal<_RAIter>::M_num_threads`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::pop_front()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>::push_front()`, `__gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`, and `std::swap()`.

Referenced by `__qsb_conquer()`.



**4.6.4.52** `template<typename _RandomNumberGenerator > int  
__gnu_parallel::__random_number_pow2 ( int __logp,  
_RandomNumberGenerator & __rng ) [inline]`

Generate a random number in  $[0, 2^{\text{__logp}})$ .

#### Parameters

*\_\_logp* Logarithm (basis 2) of the upper range *\_\_bound*.

*\_\_rng* Random number generator to use.

Definition at line 115 of file `random_shuffle.h`.

Referenced by `__parallel_random_shuffle_drs_pu()`, and `__sequential_random_shuffle()`.

**4.6.4.53** `template<typename _Size > _Size __gnu_parallel::__rd_log2 ( _Size  
__n ) [inline]`

Calculates the rounded-down logarithm of *\_\_n* for base 2.

#### Parameters

*\_\_n* Argument.

#### Returns

Returns 0 for any argument  $< 1$ .

Definition at line 102 of file `parallel/base.h`.

Referenced by `__parallel_random_shuffle_drs()`, `__parallel_sort_qsb()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.54** `template<typename _Tp > _Tp __gnu_parallel::__round_up_to_pow2  
( _Tp __x )`

Round up to the next greater power of 2.

#### Parameters

*\_\_x* Integer to round up

Definition at line 246 of file `random_shuffle.h`.

References `__rd_log2()`.

Referenced by `__parallel_random_shuffle_drs()`, `__sequential_random_shuffle()`, and `multiseq_selection()`.

**4.6.4.55** `template<typename __RAIter1 , typename __RAIter2 , typename  
__Pred > __RAIter1 __gnu_parallel::__search_template ( __RAIter1  
__begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2  
__end2, __Pred __pred )`

Parallel `std::search`.

#### Parameters

`__begin1` Begin iterator of first sequence.  
`__end1` End iterator of first sequence.  
`__begin2` Begin iterator of second sequence.  
`__end2` End iterator of second sequence.  
`__pred` Find predicate.

#### Returns

Place of finding in first sequences.

Definition at line 81 of file `search.h`.

References `__calc_borders()`, `_GLIBCXX_CALL`, `equally_split()`, and `min()`.

**4.6.4.56** `template<bool __stable, bool __sentinels, typename _RAIterIterator ,  
typename _RAIter3 , typename _DifferenceTp , typename _Compare  
> _RAIter3 __gnu_parallel::__sequential_multiway_merge (   
_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, const typename std::iterator_traits< typename  
std::iterator_traits< _RAIterIterator >::value_type::first_type  
>::value_type & __sentinel, _DifferenceTp __length, _Compare  
__comp )`

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and run-time settings.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.  
`__seqs_end` End iterator of iterator pair input sequence.  
`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, possibly larger than the number of elements available.

`__stable` Stable merging incurs a performance penalty.

`__sentinel` The sequences have `__a` `__sentinel` element.

### Returns

End iterator of output sequence.

Definition at line 917 of file `multiway_merge.h`.

References `__is_sorted()`, `__merge_advance()`, `_GLIBCXX_CALL`, and `__GLIBCXX_PARALLEL_LENGTH`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

**4.6.4.57** `template<typename _RAIter, typename _RandomNumberGenerator  
> void __gnu_parallel::__sequential_random_shuffle ( _RAIter  
__begin, _RAIter __end, _RandomNumberGenerator & __rng )`

Sequential cache-efficient random shuffle.

### Parameters

`__begin` Begin iterator of sequence.

`__end` End iterator of sequence.

`__rng` Random number generator to use.

Definition at line 408 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`.

**4.6.4.58** `template<typename _Iter > void __gnu_parallel::__shrink (   
std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t  
& __range_length )`

Combines two ranges into one and thus halves the number of ranges.

### Parameters

`__os_starts` Start positions worked on (oversampled).

***\_\_count\_to\_two*** Counts up to 2.  
***\_\_range\_length*** Current length of a chunk.

Definition at line 70 of file list\_partition.h.

References `std::vector< _Tp, _Alloc >::size()`.

Referenced by `__shrink_and_double()`.

**4.6.4.59** `template<typename _Iter > void __gnu_parallel::__shrink_and_double ( std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice )`

Shrinks and doubles the ranges.

#### Parameters

***\_\_os\_starts*** Start positions worked on (oversampled).  
***\_\_count\_to\_two*** Counts up to 2.  
***\_\_range\_length*** Current length of a chunk.  
***\_\_make\_twice*** Whether the `__os_starts` is allowed to be grown or not

Definition at line 50 of file list\_partition.h.

References `__shrink()`, `std::vector< _Tp, _Alloc >::resize()`, and `std::vector< _Tp, _Alloc >::size()`.

Referenced by `list_partition()`.

**4.6.4.60** `void __gnu_parallel::__yield ( ) [inline]`

Yield the control to another thread, without waiting for the end to the time slice.

Definition at line 354 of file parallel/compatibility.h.

Referenced by `__for_each_template_random_access_workstealing()`, and `__qsb_local_sort_with_helping()`.

**4.6.4.61** `template<typename _DifferenceType, typename _OutputIterator > _OutputIterator __gnu_parallel::equally_split ( _DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s )`

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0,__n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

**Parameters**

`__n` Number of elements  
`__num_threads` Number of parts  
`__s` Splitters

**Returns**

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, `__search_template()`, and `multiway_merge_exact_splitting()`.

**4.6.4.62** `template<typename _DifferenceType > _DifferenceType  
 __gnu_parallel::equally_split_point ( _DifferenceType __n,  
 _ThreadIndex __num_threads, _ThreadIndex __thread_no )`

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

**Parameters**

`__n` Number of elements  
`__num_threads` Number of parts

**Returns**

splitting point

Definition at line 74 of file `equally_split.h`.

Referenced by `__for_each_template_random_access_ed()`.

**4.6.4.63** `template<typename _IIter , typename _FunctorType > size_t  
 __gnu_parallel::list_partition ( const _IIter __begin, const _IIter  
 __end, _IIter * __starts, size_t * __lengths, const int __num_parts,  
 _FunctorType & __f, int __oversampling = 0 )`

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

**Parameters**

- \_\_begin** Begin iterator of input sequence.
- \_\_end** End iterator of input sequence.
- \_\_starts** Start iterators for the resulting parts, dimension `__num_parts+1`. For convenience, `__starts [__num_parts]` contains the end iterator of the sequence.
- \_\_lengths** Length of the resulting parts.
- \_\_num\_parts** Number of parts to split the sequence into.
- \_\_f** Functor to be applied to each element by traversing `__it`
- \_\_oversampling** Oversampling factor. If 0, then the partitions will differ in at most  $\{ \{ \text{__end} \} - \{ \text{__begin} \} \}$  `__elements`. Otherwise, the ratio between the longest and the shortest part is bounded by  $1/(\{ \text{__oversampling} \} \{ \text{num} \})$

**Returns**

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector<_Tp, _Alloc>::size()`.

**4.6.4.64** `template<typename _Tp> const _Tp& __gnu_parallel::max ( const _Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::max`.

Definition at line 150 of file `parallel/base.h`.

Referenced by `__parallel_nth_element()`, `multiseq_partition()`, and `multiseq_selection()`.

**4.6.4.65** `template<typename _Tp> const _Tp& __gnu_parallel::min ( const _Tp & __a, const _Tp & __b ) [inline]`

Equivalent to `std::min`.

Definition at line 144 of file `parallel/base.h`.

Referenced by `__for_each_template_random_access_workstealing()`, `__parallel_sort_qs_divide()`, `__search_template()`, `multiseq_partition()`, and `multiseq_selection()`.

```

4.6.4.66 template<typename _RanSeqs , typename _RankType ,
 typename _RankIterator , typename _Compare > void __gnu_
parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs
__end_seqs, _RankType __rank, _RankIterator __begin_offsets,
_Compare __comp = std::less< typename std::iterator_
traits<typename std::iterator_traits<_
RanSeqs>::value_type:: first_type>::value_type> ()
)

```

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

#### Parameters

`__begin_seqs` Begin of the sequence of iterator pairs.

`__end_seqs` End of the sequence of iterator pairs.

`__rank` The global rank to partition at.

`__begin_offsets` A random-access `__sequence` `__begin` where the `__result` will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective `__sequence`.

`__comp` The ordering functor, defaults to `std::less<_Tp>`.

Definition at line 124 of file `multiseq_selection.h`.

References `__rd_log2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc>::begin()`, `std::distance()`, `__gnu_cxx::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::empty()`, `std::vector<_Tp, _Alloc>::end()`, `std::make_pair()`, `max()`, `min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<_Tp, _Alloc>::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

Referenced by `multiway_merge_exact_splitting()`.

```

4.6.4.67 template<typename _Tp , typename _RanSeqs , typename _RankType
, typename _Compare > _Tp __gnu_parallel::multiseq_selection (
 _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,
 _RankType & __offset, _Compare __comp = std::less<_Tp> ())

```

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

**Parameters**

- `__begin_seqs`** Begin of the sequence of iterator pairs.
- `__end_seqs`** End of the sequence of iterator pairs.
- `__rank`** The global rank to partition at.
- `__offset`** The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
- `__comp`** The ordering functor, defaults to [std::less](#).

Definition at line 390 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc>::begin()`, `std::distance()`, `__gnu_cxx::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::empty()`, `std::vector<_Tp, _Alloc>::end()`, `std::make_pair()`, `max()`, `min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<_Tp, _Alloc>::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

**4.6.4.68** `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut  
__gnu_parallel::multiway_merge ( _RAIterPairIterator  
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut  
__target, _DifferenceTp __length, _Compare __comp,  
__gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels



Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __i < 10; ++__j)
 sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

#### See also

`stable_multiway_merge`

#### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

#### Postcondition

[`__target`, return `__value`) contains merged `__elements` from the input sequences.  
`return __value - __target = min(__length, number of elements in all sequences).`

#### Parameters

*`_RAIterPairIterator`* iterator over sequence of pairs of iterators

*`_RAIterOut`* iterator over target sequence

*`_DifferenceTp`* difference type for the sequence

*`_Compare`* strict weak ordering type to compare elements in sequences

*`__seqs_begin`* `__begin` of sequence `__sequence`

*`__seqs_end`* `_M_end` of sequence `__sequence`

*`__target`* target sequence to merge to.

*`__comp`* strict weak ordering to use for element comparison.

*`__length`* Maximum length to merge, possibly larger than the number of elements available.

#### Returns

`_M_end` iterator of output sequence

Definition at line 1410 of file multiway\_merge.h.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

**4.6.4.69** `template<template< typename RAI, typename C > class  
iterator, typename _RAIterIterator , typename _RAIter3 ,  
typename _DifferenceTp , typename _Compare > _RAIter3  
__gnu_parallel::multiway_merge_3_variant ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
_DifferenceTp __length, _Compare __comp )`

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 234 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

**4.6.4.70** `template<template< typename RAI, typename C > class  
iterator, typename _RAIterIterator , typename _RAIter3 ,  
typename _DifferenceTp , typename _Compare > _RAIter3  
__gnu_parallel::multiway_merge_4_variant ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
_DifferenceTp __length, _Compare __comp )`

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.  
`__seqs_end` End iterator of iterator pair input sequence.  
`__target` Begin iterator of output sequence.  
`__comp` Comparator.  
`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 353 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

**4.6.4.71** `template<bool __stable, typename _RAIterIterator ,  
typename _Compare , typename _DifferenceType > void  
__gnu_parallel::multiway_merge_exact_splitting ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _DifferenceType  
__length, _DifferenceType __total_length, _Compare __comp,  
std::vector< std::pair< _DifferenceType, _DifferenceType > > *  
__pieces )`

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1112 of file multiway\_merge.h.

References `_GLIBCXX_PARALLEL_LENGTH`, `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `equally_split()`, `multiseq_partition()`, and `std::vector< _Tp, _Alloc >::resize()`.

Referenced by `__parallel_merge_advance()`.

**4.6.4.72** `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3  
__gnu_parallel::multiway_merge_loser_tree ( _RAIterIterator  
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
_DifferenceTp __length, _Compare __comp )`

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `_LT`.

Stability is selected through the used `LoserTree` class `_LT`.

At least one non-empty sequence is required.

#### Parameters

`__seqs_begin` Begin iterator of iterator pair input sequence.

`__seqs_end` End iterator of iterator pair input sequence.

`__target` Begin iterator of output sequence.

`__comp` Comparator.

`__length` Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 484 of file multiway\_merge.h.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

**4.6.4.73** `template<typename UnguardedLoserTree , typename _RAIterIterator  
, typename _RAIter3 , typename _DifferenceTp , typename _Compare  
> _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel  
( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, const typename std::iterator_traits< typename  
std::iterator_traits< _RAIterIterator >::value_type::first_type  
>::value_type & __sentinel, _DifferenceTp __length, _Compare  
__comp )`

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

#### Parameters

- \_\_stable* The value must be the same as for the used LoserTrees.
- UnguardedLoserTree* \_Loser Tree variant to use for the unguarded merging.
- GuardedLoserTree* \_Loser Tree variant to use for the guarded merging.
- \_\_seqs\_begin* Begin iterator of iterator pair input sequence.
- \_\_seqs\_end* End iterator of iterator pair input sequence.
- \_\_target* Begin iterator of output sequence.
- \_\_comp* Comparator.
- \_\_length* Maximum length to merge, less equal than the total number of elements available.

#### Returns

End iterator of output sequence.

Definition at line 658 of file `multiway_merge.h`.

References `__is_sorted()`, and `_GLIBCXX_CALL`.

**4.6.4.74** `template<typename _LT , typename _RAIterIterator , typename  
_RAIter3 , typename _DifferenceTp , typename _Compare >  
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded  
( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
_RAIter3 __target, const typename std::iterator_traits< typename  
std::iterator_traits< _RAIterIterator >::value_type::first_type  
>::value_type & __sentinel, _DifferenceTp __length, _Compare  
__comp )`

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the LoserTree class `_LT`.

Stability is selected by the used LoserTrees.

**Precondition**

No input will run out of elements during the merge.

**Parameters**

***\_\_seqs\_begin*** Begin iterator of iterator pair input sequence.  
***\_\_seqs\_end*** End iterator of iterator pair input sequence.  
***\_\_target*** Begin iterator of output sequence.  
***\_\_comp*** Comparator.  
***\_\_length*** Maximum length to merge, less equal than the total number of elements available.

**Returns**

End iterator of output sequence.

Definition at line 567 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

**4.6.4.75** `template<bool __stable, typename _RAIterIterator ,  
 typename _Compare , typename _DifferenceType > void  
 __gnu_parallel::multiway_merge_sampling_splitting (   
 _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,  
 _DifferenceType __length, _DifferenceType __total_length,  
 _Compare __comp, std::vector< std::pair< _DifferenceType,  
 _DifferenceType > > * __pieces )`

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1032 of file multiway\_merge.h.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`,  
 and `__gnu_parallel::_Settings::merge_oversampling`.

**4.6.4.76** `template<typename _RAIterPairIterator , typename _RAIterOut ,  
 typename _DifferenceTp , typename _Compare > _RAIterOut  
 __gnu_parallel::multiway_merge_sentinels ( _RAIterPairIterator  
 __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut  
 __target, _DifferenceTp __length, _Compare __comp,  
 __gnu_parallel::sequential_tag )`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_`-  
`begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an

iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __j < 11; ++__j)
 sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

### Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.  
`return __value - __target = min(__length, number of elements in all sequences).`

**See also**

`stable_multiway_merge_sentinels`

**Parameters**

***\_RAIterPairIterator*** iterator over sequence of pairs of iterators  
***\_RAIterOut*** iterator over target sequence  
***\_DifferenceTp*** difference type for the sequence  
***\_Compare*** strict weak ordering type to compare elements in sequences  
***\_\_seqs\_begin*** \_\_begin of sequence \_\_sequence  
***\_\_seqs\_end*** \_\_M\_end of sequence \_\_sequence  
***\_\_target*** target sequence to merge to.  
***\_\_comp*** strict weak ordering to use for element comparison.  
***\_\_length*** Maximum length to merge, possibly larger than the number of elements available.

**Returns**

`__M_end` iterator of output sequence

Definition at line 1774 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

**4.6.4.77** `template<bool __stable, bool __sentinels, typename  
     _RAIterIterator, typename _RAIter3, typename _DifferenceTp  
     , typename _Splitter, typename _Compare > _RAIter3  
     __gnu_parallel::parallel_multiway_merge ( _RAIterIterator  
     __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target,  
     _Splitter __splitter, _DifferenceTp __length, _Compare __comp,  
     _ThreadIndex __num_threads )`

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and run-time settings.

Must not be called if the number of sequences is 1.

**Parameters**

***\_Splitter*** functor to split input (either `__exact` or sampling based)  
***\_\_seqs\_begin*** Begin iterator of iterator pair input sequence.  
***\_\_seqs\_end*** End iterator of iterator pair input sequence.



- \_\_target** Begin iterator of output sequence.
- \_\_comp** Comparator.
- \_\_length** Maximum length to merge, possibly larger than the number of elements available.
- \_\_stable** Stable merging incurs a performance penalty.
- \_\_sentinel** Ignored.

### Returns

End iterator of output sequence.

Definition at line 1217 of file multiway\_merge.h.

References `__is_sorted()`, `__GLIBCXX_CALL`, `__GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, and `__gnu_parallel::_Settings::merge_oversampling`.

Referenced by `__parallel_merge_advance()`.

**4.6.4.78** `template<bool __stable, bool __exact, typename _RAIter, typename _Compare> void __gnu_parallel::parallel_sort_mwms ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads )`

PMWMS main call.

### Parameters

- \_\_begin** Begin iterator of sequence.
- \_\_end** End iterator of sequence.
- \_\_comp** Comparator.
- \_\_n** Length of sequence.
- \_\_num\_threads** Number of threads to use.

Definition at line 394 of file multiway\_mergesort.h.

References `__GLIBCXX_CALL`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_num_threads`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_offsets`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_pieces`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_samples`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_source`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_starts`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_temporary`, `__gnu_parallel::_Settings::get()`, `std::vector< _Tp, _Alloc >::resize()`, and `__gnu_parallel::_Settings::sort_mwms_oversampling`.

**4.6.4.79** `template<bool __stable, bool __exact, typename _RAIter, typename  
_Compare > void __gnu_parallel::parallel_sort_mwms_pu (  
_PMWMSortingData< _RAIter > * __sd, _Compare & __comp )`

PMWMS code executed by each thread.

#### Parameters

`__sd` Pointer to algorithm data.

`__comp` Comparator.

Definition at line 308 of file multiway\_mergesort.h.

References `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_num_`  
`threads`, `__gnu_parallel::_PMWMSortingData< _RAIter >::_M_pieces`,  
`__gnu_parallel::_PMWMSortingData< _RAIter >::_M_source`, `__gnu_`  
`parallel::_PMWMSortingData< _RAIter >::_M_starts`, `__gnu_parallel::_`  
`PMWMSortingData< _RAIter >::_M_temporary`, `__gnu_parallel::_Settings::get()`,  
`std::make_pair()`, `__gnu_parallel::_Settings::sort_mwms_oversampling`, and  
`std::uninitialized_copy()`.

### 4.6.5 Variable Documentation

**4.6.5.1** `const int __gnu_parallel::_CASable_bits [static]`

Number of bits of `_CASable`.

Definition at line 130 of file types.h.

Referenced by `__decode2()`, and `__encode2()`.

**4.6.5.2** `const _CASable __gnu_parallel::_CASable_mask [static]`

`_CASable` with the right half of bits set to 1.

Definition at line 133 of file types.h.

Referenced by `__decode2()`.

## 4.7 \_\_gnu\_pbds Namespace Reference

GNU extensions for policy-based data structures for public use.

## Classes

- struct [associative\\_container\\_tag](#)  
*Basic associative-container.*
- class [basic\\_hash\\_table](#)  
*An abstract basic hash-based associative container.*
- struct [basic\\_hash\\_tag](#)  
*Basic hash.*
- class [basic\\_tree](#)  
*An abstract basic tree-like (tree, trie) associative container.*
- struct [basic\\_tree\\_tag](#)  
*Basic tree.*
- struct [binary\\_heap\\_tag](#)  
*Binary-heap (array-based).*
- struct [binomial\\_heap\\_tag](#)  
*Binomial-heap.*
- class [cc\\_hash\\_table](#)  
*A concrete collision-chaining hash-based associative container.*
- struct [cc\\_hash\\_tag](#)  
*Collision-chaining hash.*
- class [container\\_base](#)  
*An abstract basic associative container.*
- struct [container\\_tag](#)  
*Base data structure tag.*
- struct [container\\_traits](#)  
*[container\\_traits](#)*
- class [gp\\_hash\\_table](#)  
*A concrete general-probing hash-based associative container.*
- struct [gp\\_hash\\_tag](#)  
*General-probing hash.*

- class [list\\_update](#)  
*A list-update based associative container.*
- struct [list\\_update\\_tag](#)  
*List-update.*
- struct [null\\_mapped\\_type](#)  
*A mapped-policy indicating that an associative container is a set.*
- struct [ov\\_tree\\_tag](#)  
*Ordered-vector tree.*
- struct [pairing\\_heap\\_tag](#)  
*Pairing-heap.*
- struct [pat\\_trie\\_tag](#)  
*PATRICIA trie.*
- struct [priority\\_queue\\_tag](#)  
*Basic priority-queue.*
- struct [rb\\_tree\\_tag](#)  
*Red-black tree.*
- struct [rc\\_binomial\\_heap\\_tag](#)  
*Redundant-counter binomial-heap.*
- struct [sequence\\_tag](#)  
*Basic sequence.*
- struct [splay\\_tree\\_tag](#)  
*Splay tree.*
- struct [string\\_tag](#)  
*Basic string container, inclusive of strings, ropes, etc.*
- struct [thin\\_heap\\_tag](#)  
*Thin heap.*
- class [tree](#)  
*A concrete basic tree-based associative container.*

- struct [tree\\_tag](#)  
*tree.*
- class [trie](#)  
*A concrete basic trie-based associative container.*
- struct [trie\\_tag](#)  
*trie.*

## Typedefs

- typedef void `trivial_iterator_difference_type`

## Functions

- void `__throw_container_error` (void)
- void `__throw_insert_error` (void)
- void `__throw_join_error` (void)
- void `__throw_resize_error` (void)

### 4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 4.8 `__gnu_profile` Namespace Reference

GNU profile code for public use.

## Classes

- class [\\_\\_container\\_size\\_info](#)  
*A container size instrumentation line in the object table.*
- class [\\_\\_container\\_size\\_stack\\_info](#)  
*A container size instrumentation line in the stack table.*
- class [\\_\\_hashfunc\\_info](#)

*A hash performance instrumentation line in the object table.*

- class [\\_\\_hashfunc\\_stack\\_info](#)

*A hash performance instrumentation line in the stack table.*

- class [\\_\\_list2vector\\_info](#)

*A list-to-vector instrumentation line in the object table.*

- class [\\_\\_map2umap\\_info](#)

*A map-to-unordered\_map instrumentation line in the object table.*

- class [\\_\\_map2umap\\_stack\\_info](#)

*A map-to-unordered\_map instrumentation line in the stack table.*

- class [\\_\\_object\\_info\\_base](#)

*Base class for a line in the object table.*

- struct [\\_\\_reentrance\\_guard](#)

*Reentrance guard.*

- class [\\_\\_stack\\_hash](#)

*Hash function for summary trace using call stack as index.*

- class [\\_\\_stack\\_info\\_base](#)

*Base class for a line in the stack table.*

- class [\\_\\_trace\\_base](#)

*Base class for all trace producers.*

- class [\\_\\_trace\\_container\\_size](#)

*Container size instrumentation trace producer.*

- class [\\_\\_trace\\_hash\\_func](#)

*Hash performance instrumentation producer.*

- class [\\_\\_trace\\_hashtable\\_size](#)

*Hashtable size instrumentation trace producer.*

- class [\\_\\_trace\\_map2umap](#)

*Map-to-unordered\_map instrumentation producer.*

- class [\\_\\_trace\\_vector\\_size](#)

*Hashtable size instrumentation trace producer.*

- class `__trace_vector_to_list`  
*Vector-to-list instrumentation producer.*
- class `__vector2list_info`  
*A vector-to-list instrumentation line in the object table.*
- class `__vector2list_stack_info`  
*A vector-to-list instrumentation line in the stack table.*
- struct `__warning_data`  
*Representation of a warning.*

## Typedefs

- typedef `std::vector< __cost_factor * >` `__cost_factor_vector`
- typedef `std::unordered_map< std::string, std::string >` `__env_t`
- typedef `void *` `__instruction_address_t`
- typedef `const void *` `__object_t`
- typedef `std::vector< __instruction_address_t >` `__stack_npt`
- typedef `__stack_npt *` `__stack_t`
- typedef `std::vector< __warning_data >` `__warning_vector_t`

## Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

## Functions

- `std::size_t __env_to_size_t (const char * __env_var, std::size_t __default_value)`
- `template<typename _InputIterator, typename _Function >`  
`_Function __for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `__cost_factor_vector *& __get__cost_factors ()`
- `__env_t & __get__env ()`
- `__gnu_cxx::__mutex & __get__global_lock ()`
- `__cost_factor & __get__list_iterate_cost_factor ()`
- `__cost_factor & __get__list_resize_cost_factor ()`
- `__cost_factor & __get__list_shift_cost_factor ()`

- `__cost_factor & __get__map_erase_cost_factor ()`
- `__cost_factor & __get__map_find_cost_factor ()`
- `__cost_factor & __get__map_insert_cost_factor ()`
- `__cost_factor & __get__map_iterate_cost_factor ()`
- `__cost_factor & __get__umap_erase_cost_factor ()`
- `__cost_factor & __get__umap_find_cost_factor ()`
- `__cost_factor & __get__umap_insert_cost_factor ()`
- `__cost_factor & __get__umap_iterate_cost_factor ()`
- `__cost_factor & __get__vector_iterate_cost_factor ()`
- `__cost_factor & __get__vector_resize_cost_factor ()`
- `__cost_factor & __get__vector_shift_cost_factor ()`
- `__trace_hash_func *& __get__S_hash_func ()`
- `__trace_hashtable_size *& __get__S_hashtable_size ()`
- `__trace_list_to_slist *& __get__S_list_to_slist ()`
- `__trace_list_to_vector *& __get__S_list_to_vector ()`
- `__trace_map2umap *& __get__S_map2umap ()`
- `std::size_t & __get__S_max_mem ()`
- `std::size_t & __get__S_max_stack_depth ()`
- `std::size_t & __get__S_max_warn_count ()`
- `const char *& __get__S_trace_file_name ()`
- `__trace_vector_size *& __get__S_vector_size ()`
- `__trace_vector_to_list *& __get__S_vector_to_list ()`
- `__stack_t __get_stack ()`
- `template<typename _Container >  
void __insert_top_n (_Container &__output, const typename _-  
Container::value_type &__value, typename _Container::size_type __n)`
- `bool __is_invalid ()`
- `bool __is_off ()`
- `bool __is_on ()`
- `int __log2 (std::size_t __size)`
- `int __log_magnitude (float __f)`
- `float __map_erase_cost (std::size_t __size)`
- `float __map_find_cost (std::size_t __size)`
- `float __map_insert_cost (std::size_t __size)`
- `std::size_t __max_mem ()`
- `FILE * __open_output_file (const char *__extension)`
- `bool __profcxx_init ()`
- `void __profcxx_init_unconditional ()`
- `void __read_cost_factors ()`
- `template<typename _ForwardIterator, typename _Tp >  
_ForwardIterator __remove (_ForwardIterator __first, _ForwardIterator __last,  
const _Tp &__value)`



- void `__report` (void)
- void `__set_cost_factors` ()
- void `__set_max_mem` ()
- void `__set_max_stack_trace_depth` ()
- void `__set_max_warn_count` ()
- void `__set_trace_path` ()
- `std::size_t` `__size` (`__stack_t` `__stack`)
- `std::size_t` `__stack_max_depth` ()
- `template<typename _Container >`  
`void` `__top_n` (`const _Container &__input`, `_Container &__output`, `typename _Container::size_type __n`)
- void `__trace_hash_func_construct` (`const void *`)
- void `__trace_hash_func_destruct` (`const void *`, `std::size_t`, `std::size_t`, `std::size_t`)
- void `__trace_hash_func_init` ()
- void `__trace_hash_func_report` (`FILE *__f`, `__warning_vector_t &__warnings`)
- void `__trace_hashtable_size_construct` (`const void *`, `std::size_t`)
- void `__trace_hashtable_size_destruct` (`const void *`, `std::size_t`, `std::size_t`)
- void `__trace_hashtable_size_init` ()
- void `__trace_hashtable_size_report` (`FILE *__f`, `__warning_vector_t &__warnings`)
- void `__trace_hashtable_size_resize` (`const void *`, `std::size_t`, `std::size_t`)
- void `__trace_list_to_set_construct` (`const void *`)
- void `__trace_list_to_set_destruct` (`const void *`)
- void `__trace_list_to_set_find` (`const void *`, `std::size_t`)
- void `__trace_list_to_set_insert` (`const void *`, `std::size_t`, `std::size_t`)
- void `__trace_list_to_set_invalid_operator` (`const void *`)
- void `__trace_list_to_set_iterate` (`const void *`, `std::size_t`)
- void `__trace_list_to_slist_construct` (`const void *`)
- void `__trace_list_to_slist_destruct` (`const void *`)
- void `__trace_list_to_slist_init` ()
- void `__trace_list_to_slist_operation` (`const void *`)
- void `__trace_list_to_slist_report` (`FILE *__f`, `__warning_vector_t &__warnings`)
- void `__trace_list_to_slist_rewind` (`const void *`)
- void `__trace_list_to_vector_construct` (`const void *`)
- void `__trace_list_to_vector_destruct` (`const void *`)
- void `__trace_list_to_vector_init` ()
- void `__trace_list_to_vector_insert` (`const void *`, `std::size_t`, `std::size_t`)
- void `__trace_list_to_vector_invalid_operator` (`const void *`)
- void `__trace_list_to_vector_iterate` (`const void *`, `std::size_t`)

- void **\_\_trace\_list\_to\_vector\_report** (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void **\_\_trace\_list\_to\_vector\_resize** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_construct** (const void \*)
- void **\_\_trace\_map\_to\_unordered\_map\_destruct** (const void \*)
- void **\_\_trace\_map\_to\_unordered\_map\_erase** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_find** (const void \*, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_init** ()
- void **\_\_trace\_map\_to\_unordered\_map\_insert** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_invalidate** (const void \*)
- void **\_\_trace\_map\_to\_unordered\_map\_iterate** (const void \*, std::size\_t)
- void **\_\_trace\_map\_to\_unordered\_map\_report** (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void **\_\_trace\_vector\_size\_construct** (const void \*, std::size\_t)
- void **\_\_trace\_vector\_size\_destruct** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_vector\_size\_init** ()
- void **\_\_trace\_vector\_size\_report** (FILE \*, \_\_warning\_vector\_t &)
- void **\_\_trace\_vector\_size\_resize** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_vector\_to\_list\_construct** (const void \*)
- void **\_\_trace\_vector\_to\_list\_destruct** (const void \*)
- void **\_\_trace\_vector\_to\_list\_find** (const void \*, std::size\_t)
- void **\_\_trace\_vector\_to\_list\_init** ()
- void **\_\_trace\_vector\_to\_list\_insert** (const void \*, std::size\_t, std::size\_t)
- void **\_\_trace\_vector\_to\_list\_invalid\_operator** (const void \*)
- void **\_\_trace\_vector\_to\_list\_iterate** (const void \*, std::size\_t)
- void **\_\_trace\_vector\_to\_list\_report** (FILE \*, \_\_warning\_vector\_t &)
- void **\_\_trace\_vector\_to\_list\_resize** (const void \*, std::size\_t, std::size\_t)
- bool **\_\_turn** (\_\_state\_type \_\_s)
- bool **\_\_turn\_off** ()
- bool **\_\_turn\_on** ()
- void **\_\_write** (FILE \*\_\_f, \_\_stack\_t \_\_stack)
- void **\_\_write\_cost\_factors** ()
- **\_GLIBCXX\_PROFILE\_DEFINE\_DATA** (\_\_state\_type, \_\_state, \_\_INVALID)

### 4.8.1 Detailed Description

GNU profile code for public use.

## 4.8.2 Typedef Documentation

### 4.8.2.1 `typedef std:: std ::unordered_map<std::string, std::string>` `__gnu_profile::__env_t`

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is libstdcxx-profile.conf. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a map like this: `export __map_iterate_cost_factor=5.0`. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 72 of file `profiler_trace.h`.

## 4.8.3 Function Documentation

### 4.8.3.1 `__gnu_cxx::__mutex& __gnu_profile::__get__global_lock ( )` `[inline]`

Master lock.

Definition at line 77 of file `profiler_trace.h`.

### 4.8.3.2 `bool __gnu_profile::__profcxx_init ( ) [inline]`

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 656 of file `profiler_trace.h`.

### 4.8.3.3 `void __gnu_profile::__report ( void ) [inline]`

Final report method, registered with `atexit`.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in `profiler.h`. However, when called from a signal handler that triggers while within `__gnu_profile` (under the guarded zone), no output will be produced.

Definition at line 447 of file `profiler_trace.h`.

References `std::min()`.

## 4.9 `__gnu_sequential` Namespace Reference

GNU sequential classes for public use.

### 4.9.1 Detailed Description

GNU sequential classes for public use.

## 4.10 `abi` Namespace Reference

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`.

### 4.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`. A brief overview of an ABI is given in the `libstdc++` FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq/index.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq/index.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 4.11 `std` Namespace Reference

ISO C++ entities toplevel namespace is `std`.

### Namespaces

- namespace `__debug`
- namespace `__detail`

- namespace [\\_\\_parallel](#)
- namespace [\\_\\_profile](#)
- namespace [chrono](#)
- namespace [decimal](#)
- namespace [placeholders](#)
- namespace [regex\\_constants](#)
- namespace [rel\\_ops](#)
- namespace [this\\_thread](#)
- namespace [tr1](#)

## Classes

- class [\\_\\_basic\\_future](#)  
*Common implementation for future and [shared\\_future](#).*
- class [\\_\\_codecvt\\_abstract\\_base](#)  
*Common base for codecvt functions.*
- class [\\_\\_ctype\\_abstract\\_base](#)  
*Common base for ctype facet.*
- struct [\\_\\_declval\\_protector](#)  
*declval*
- struct [\\_\\_future\\_base](#)  
*Base class and enclosing scope.*
- struct [\\_\\_is\\_location\\_invariant](#)
- struct [\\_\\_iterator\\_traits](#)  
*Traits class for iterators.*
- struct [\\_\\_numeric\\_limits\\_base](#)  
*Part of `std::numeric_limits`.*
- struct [\\_Base\\_bitset](#)
- struct [\\_Base\\_bitset< 0 >](#)
- struct [\\_Base\\_bitset< 1 >](#)
- struct [\\_Build\\_index\\_tuple](#)  
*Builds an [\\_Index\\_tuple](#)<0, 1, 2, ..., \_Num-1>.*
- class [\\_Deque\\_base](#)
- struct [\\_Deque\\_iterator](#)

*A deque::iterator.*

- struct [\\_Derives\\_from\\_binary\\_function](#)  
*Determines if the type `_Tp` derives from [binary\\_function](#).*
- struct [\\_Derives\\_from\\_unary\\_function](#)  
*Determines if the type `_Tp` derives from [unary\\_function](#).*
- class [\\_Function\\_base](#)  
*Base class of all polymorphic function object wrappers.*
- struct [\\_Function\\_to\\_function\\_pointer](#)  
*Turns a function type into a function pointer type.*
- struct [\\_Fwd\\_list\\_base](#)  
*Base class for forward\_list.*
- struct [\\_Fwd\\_list\\_const\\_iterator](#)  
*A forward\_list::const\_iterator.*
- struct [\\_Fwd\\_list\\_iterator](#)  
*A forward\_list::iterator.*
- struct [\\_Fwd\\_list\\_node](#)  
*A helper node class for forward\_list. This is just a linked list with a data value in each node. There is a sorting utility method.*
- struct [\\_Fwd\\_list\\_node\\_base](#)  
*A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*
- struct [\\_Index\\_tuple](#)
- class [\\_List\\_base](#)  
*See [bits/stl\\_deque.h](#)'s [\\_Deque\\_base](#) for an explanation.*
- struct [\\_List\\_const\\_iterator](#)  
*A list::const\_iterator.*
- struct [\\_List\\_iterator](#)  
*A list::iterator.*
- struct [\\_List\\_node](#)  
*An actual node in the list.*

- struct [\\_List\\_node\\_base](#)  
*Common part of a node in the list.*
- struct [\\_Maybe\\_get\\_result\\_type](#)  
*If we have found a result\_type, extract it.*
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function](#)
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1 >](#)  
*Derives from [unary\\_function](#), as appropriate.*
- struct [\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1, \\_T2 >](#)  
*Derives from [binary\\_function](#), as appropriate.*
- struct [\\_Maybe\\_wrap\\_member\\_pointer](#)
- struct [\\_Maybe\\_wrap\\_member\\_pointer< \\_Tp \\_Class::\\* >](#)
- class [\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Implementation of `mem_fn` for const member function pointers.*
- class [\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Implementation of `mem_fn` for const volatile member function pointers.*
- class [\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)  
*Implementation of `mem_fn` for volatile member function pointers.*
- class [\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) >](#)  
*Implementation of `mem_fn` for member function pointers.*
- class [\\_Mu< \\_Arg, false, false >](#)
- class [\\_Mu< \\_Arg, false, true >](#)
- class [\\_Mu< \\_Arg, true, false >](#)
- class [\\_Mu< reference\\_wrapper< \\_Tp >, false, false >](#)
- struct [\\_Placeholder](#)  
*The type of placeholder objects defined by `libstdc++.`*
- struct [\\_Reference\\_wrapper\\_base](#)
- struct [\\_Safe\\_tuple\\_element](#)
- struct [\\_Safe\\_tuple\\_element\\_impl](#)
- struct [\\_Safe\\_tuple\\_element\\_impl< \\_\\_i, \\_Tuple, false >](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Tuple\\_impl< \\_Idx >](#)
- struct [\\_Tuple\\_impl< \\_Idx, \\_Head, \\_Tail...>](#)

- struct [\\_Vector\\_base](#)  
*See [bits/stl\\_deque.h](#)'s [\\_Deque\\_base](#) for an explanation.*
- struct [\\_Weak\\_result\\_type](#)
- struct [\\_Weak\\_result\\_type\\_impl](#)
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(&\)\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function reference.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\*\)\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_ArgTypes...\)>](#)  
*Retrieve the result type for a function type.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Retrieve result type for a const member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Retrieve result type for a const volatile member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)  
*Retrieve result type for a volatile member function pointer.*
- struct [\\_Weak\\_result\\_type\\_impl< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\)>](#)  
*Retrieve result type for a member function pointer.*
- struct [add\\_lvalue\\_reference](#)  
*[add\\_lvalue\\_reference](#)*
- struct [add\\_rvalue\\_reference](#)  
*[add\\_rvalue\\_reference](#)*
- struct [adopt\\_lock\\_t](#)  
*Assume the calling thread has already obtained mutex ownership /// and manage it.*
- struct [aligned\\_storage](#)  
*Alignment type.*
- class [allocator](#)  
*The standard allocator, as per [20.4].  
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*



- class `allocator< void >`  
*allocator<void> specialization.*
- struct `allocator_arg_t`  
*[allocator.tag]*
- struct `atomic`  
*atomic /// 29.4.3, Generic atomic type, primary class template.*
- struct `atomic< _Tp * >`  
*Partial specialization for pointer types.*
- struct `atomic< bool >`  
*Explicit specialization for bool.*
- struct `atomic< char >`  
*Explicit specialization for char.*
- struct `atomic< char16_t >`  
*Explicit specialization for char16\_t.*
- struct `atomic< char32_t >`  
*Explicit specialization for char32\_t.*
- struct `atomic< int >`  
*Explicit specialization for int.*
- struct `atomic< long >`  
*Explicit specialization for long.*
- struct `atomic< long long >`  
*Explicit specialization for long long.*
- struct `atomic< short >`  
*Explicit specialization for short.*
- struct `atomic< signed char >`  
*Explicit specialization for signed char.*
- struct `atomic< unsigned char >`  
*Explicit specialization for unsigned char.*

- struct [atomic< unsigned int >](#)  
*Explicit specialization for unsigned int.*
- struct [atomic< unsigned long >](#)  
*Explicit specialization for unsigned long.*
- struct [atomic< unsigned long long >](#)  
*Explicit specialization for unsigned long long.*
- struct [atomic< unsigned short >](#)  
*Explicit specialization for unsigned short.*
- struct [atomic< void \\* >](#)  
*Explicit specialization for void\*.*
- struct [atomic< wchar\\_t >](#)  
*Explicit specialization for wchar\_t.*
- class [auto\\_ptr](#)  
*A simple smart pointer providing strict ownership semantics.*
- struct [auto\\_ptr\\_ref](#)
- class [back\\_insert\\_iterator](#)  
*Turns assignment into insertion.*
- class [bad\\_alloc](#)  
*Exception possibly thrown by new.  
[bad\\_alloc](#) (or classes derived from it) is used to report allocation errors from the throwing forms of new.*
- class [bad\\_cast](#)  
*Thrown during incorrect typecasting.  
If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class [bad\\_exception](#)
- class [bad\\_function\\_call](#)  
*Exception class thrown when class template function's `operator()` is called with an empty target.*
- class [bad\\_typeid](#)  
*Thrown when a NULL pointer in a `typeid` expression is used.*

- class `basic_filebuf`  
*The actual work of input and output (for files).  
This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.*
- class `basic_fstream`  
*Controlling input and output for files.  
This class supports reading from and writing to named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*
- class `basic_ifstream`  
*Controlling input for files.  
This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*
- class `basic_ios`  
*Virtual base class for all stream classes.  
Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*
- class `basic_iostream`  
*Merging istream and ostream capabilities.  
This class multiply inherits from the input and output stream classes simply to provide a single interface.*
- class `basic_istream`  
*Controlling input.  
This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.*
- class `basic_istreamstream`  
*Controlling input for `std::string`.  
This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.*
- class `basic_ofstream`  
*Controlling output for files.  
This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.*

- class [basic\\_ostream](#)

*Controlling output.*

*This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic\\_streambuf](#) to do the actual output.*

- class [basic\\_ostringstream](#)

*Controlling output for `std::string`.*

*This class supports writing to objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.*

- class [basic\\_regex](#)

- class [basic\\_streambuf](#)

*The actual work of input and output (interface).*

*This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.*

- class [basic\\_string](#)

*Managing sequences of characters and character-like objects.*

- class [basic\\_stringbuf](#)

*The actual work of input and output (for `std::string`).*

*This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a [std::basic\\_string](#). (Paraphrased from [27.7.1]/1.).*

- class [basic\\_stringstream](#)

*Controlling input and output for `std::string`.*

*This class supports reading from and writing to objects of type [std::basic\\_string](#), using the inherited functions from [std::basic\\_iostream](#). To control the associated sequence, an instance of [std::basic\\_stringbuf](#) is used, which this page refers to as *sb*.*

- class [bernoulli\\_distribution](#)

*A Bernoulli random number distribution.*

- struct [bidirectional\\_iterator\\_tag](#)

*Bidirectional iterators support a superset of forward iterator /// operations.*

- struct [binary\\_function](#)

- class [binary\\_negate](#)

*One of the *negation* functors.*

- class [binder1st](#)

One of the *binder functors*.

- class `binder2nd`

One of the *binder functors*.

- class `binomial_distribution`

A discrete binomial random number distribution.

- class `bitset`

The `bitset` class represents a fixed-size sequence of bits.

- class `cauchy_distribution`

A *cauchy\_distribution* random number distribution.

- struct `char_traits`

Basis for explicit traits specializations.

- struct `char_traits< __gnu_cxx::character< V, I, S > >`

`char_traits<__gnu_cxx::character>` specialization.

- struct `char_traits< char >`

21.1.3.1 *char\_traits* specializations

- struct `char_traits< wchar_t >`

21.1.3.2 *char\_traits* specializations

- class `chi_squared_distribution`

A *chi\_squared\_distribution* random number distribution.

- class `codecvt`

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

- class `codecvt< _InternT, _ExternT, encoding_state >`

`codecvt<InternT, _ExternT, encoding_state>` specialization.

- class `codecvt< char, char, mbstate_t >`

class `codecvt<char, char, mbstate_t>` specialization.

- class `codecvt< wchar_t, char, mbstate_t >`

class `codecvt<wchar_t, char, mbstate_t>` specialization.

- class `codecvt_base`

*Empty base class for codecvt facet [22.2.1.5].*

- class [codecvt\\_byname](#)

*class [codecvt\\_byname](#) [22.2.1.6].*

- class [collate](#)

*Facet for localized string comparison.*

- class [collate\\_byname](#)

*class [collate\\_byname](#) [22.2.4.2].*

- struct [complex](#)

- class [condition\\_variable](#)

*[condition\\_variable](#)*

- class [condition\\_variable\\_any](#)

*[condition\\_variable\\_any](#)*

- struct [conditional](#)

*[conditional](#)*

- class [const\\_mem\\_fun1\\_ref\\_t](#)

*One of the [adaptors](#) for member /// pointers.*

- class [const\\_mem\\_fun1\\_t](#)

*One of the [adaptors](#) for member /// pointers.*

- class [const\\_mem\\_fun\\_ref\\_t](#)

*One of the [adaptors](#) for member /// pointers.*

- class [const\\_mem\\_fun\\_t](#)

*One of the [adaptors](#) for member /// pointers.*

- class [ctype](#)

*Primary class template [ctype](#) facet.*

*This template class defines classification and conversion functions for character sets. It wraps [cctype](#) functionality. [Ctype](#) gets used by streams for many I/O operations.*

- class [ctype<char>](#)

*The [ctype<char>](#) specialization.*

*This class defines classification and conversion functions for the [char](#) type. It gets used by [char](#) streams for many I/O operations. The [char](#) specialization provides a number of optimizations as well.*

- class `ctype< wchar_t >`  
*The `ctype<wchar_t>` specialization.  
This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well.*
- struct `ctype_base`  
*Base class for `ctype`.*
- class `ctype_byname`  
*class `ctype_byname` [22.2.1.2].*
- class `ctype_byname< char >`  
*22.2.1.4 Class `ctype_byname` specializations.*
- class `decay`  
*decay*
- struct `default_delete`  
*Primary template, `default_delete`.*
- struct `default_delete< _Tp[] >`  
*Specialization, `default_delete`.*
- struct `defer_lock_t`  
*Do not acquire ownership of the mutex.*
- class `deque`  
*A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.*
- class `discard_block_engine`
- class `discrete_distribution`  
*A `discrete_distribution` random number distribution.*
- struct `divides`  
*One of the `math` functors.*
- class `domain_error`
- struct `enable_if`  
*`enable_if`*

- class [enable\\_shared\\_from\\_this](#)  
*Base class allowing use of member function `shared_from_this`.*
- struct [equal\\_to](#)  
*One of the `comparison` functors.*
- class [error\\_category](#)  
*`error_category`*
- struct [error\\_code](#)  
*`error_code`*
- struct [error\\_condition](#)  
*`error_condition`*
- class [exception](#)  
*Base class for all library exceptions.*
- class [exponential\\_distribution](#)  
*An exponential continuous distribution for random numbers.*
- class [extreme\\_value\\_distribution](#)  
*A `extreme_value_distribution` random number distribution.*
- class [fisher\\_f\\_distribution](#)  
*A `fisher_f_distribution` random number distribution.*
- struct [forward\\_iterator\\_tag](#)  
*Forward iterators support a superset of input iterator operations.*
- class [forward\\_list](#)  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*
- class [fpos](#)  
*Class representing stream positions.*
- class [front\\_insert\\_iterator](#)  
*Turns assignment into insertion.*
- class [function<\\_Res\(\\_ArgTypes...\)>](#)  
*Primary class template for `std::function`.  
Polymorphic function wrapper.*



- class `future`  
*Primary template for future.*
- class `future< _Res & >`  
*Partial specialization for `future<R&>`*
- class `future< void >`  
*Explicit specialization for `future<void>`*
- class `future_error`  
*Exception type thrown by futures.*
- class `gamma_distribution`  
*A gamma continuous distribution for random numbers.*
- class `geometric_distribution`  
*A discrete geometric random number distribution.*
- struct `greater`  
*One of the *comparison functors*.*
- struct `greater_equal`  
*One of the *comparison functors*.*
- class `gslice`  
*Class defining multi-dimensional subset of an array.*
- class `gslice_array`  
*Reference to multi-dimensional subset of an array.*
- struct `has_nothrow_copy_assign`  
*`has_nothrow_copy_assign`*
- struct `has_nothrow_copy_constructor`  
*`has_nothrow_copy_constructor`*
- struct `has_nothrow_default_constructor`  
*`has_nothrow_default_constructor`*
- struct `has_trivial_copy_assign`  
*`has_trivial_copy_assign`*

- struct `has_trivial_copy_constructor`  
*has\_trivial\_copy\_constructor*
- struct `has_trivial_default_constructor`  
*has\_trivial\_default\_constructor*
- struct `has_trivial_destructor`  
*has\_trivial\_destructor*
- struct `hash`  
*Primary class template hash.*
- struct `hash< __debug::bitset< _Nb > >`  
*std::hash specialization for bitset.*
- struct `hash< __debug::vector< bool, _Alloc > >`  
*std::hash specialization for vector<bool>.*
- struct `hash< __gnu_cxx::throw_value_limit >`  
*Explicit specialization of std::hash for \_\_gnu\_cxx::throw\_value\_limit.*
- struct `hash< __gnu_cxx::throw_value_random >`  
*Explicit specialization of std::hash for \_\_gnu\_cxx::throw\_value\_limit.*
- struct `hash< __profile::bitset< _Nb > >`  
*std::hash specialization for bitset.*
- struct `hash< __profile::vector< bool, _Alloc > >`  
*std::hash specialization for vector<bool>.*
- struct `hash< __shared_ptr< _Tp, _Lp > >`  
*std::hash specialization for \_\_shared\_ptr.*
- struct `hash< _Tp * >`  
*Partial specializations for pointer types.*
- struct `hash< error_code >`  
*std::hash specialization for error\_code.*
- struct `hash< shared_ptr< _Tp > >`  
*std::hash specialization for shared\_ptr.*

- struct `hash< string >`  
*std::hash specialization for string.*
- struct `hash< thread::id >`  
*std::hash specialization for thread::id.*
- struct `hash< u16string >`  
*std::hash specialization for u16string.*
- struct `hash< u32string >`  
*std::hash specialization for u32string.*
- struct `hash< unique_ptr< _Tp, _Dp > >`  
*std::hash specialization for unique\_ptr.*
- struct `hash< wstring >`  
*std::hash specialization for wstring.*
- struct `hash<::bitset< _Nb > >`  
*std::hash specialization for bitset.*
- struct `hash<::vector< bool, _Alloc > >`  
*std::hash specialization for vector<bool>.*
- class `independent_bits_engine`
- class `indirect_array`  
*Reference to arbitrary subset of an array.*
- class `initializer_list`  
*initializer\_list*
- struct `input_iterator_tag`  
*Marking input iterators.*
- class `insert_iterator`  
*Turns assignment into insertion.*
- class `invalid_argument`
- class `ios_base`  
*The base of the I/O class hierarchy.  
This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see ios\_base when they need to specify the full name of the various I/O flags (e.g., the openmodes).*

- struct [is\\_base\\_of](#)  
*is\_base\_of*
- struct [is\\_bind\\_expression](#)  
*Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by `bind()`. [TR1 3.6.1].*
- struct [is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)  
*Class template `_Bind` is always a bind expression.*
- struct [is\\_constructible](#)  
*is\_constructible*
- struct [is\\_convertible](#)  
*is\_convertible*
- struct [is\\_error\\_code\\_enum](#)  
*is\_error\_code\_enum*
- struct [is\\_error\\_condition\\_enum](#)  
*is\_error\_condition\_enum*
- struct [is\\_explicitly\\_convertible](#)  
*is\_explicitly\_convertible*
- struct [is\\_lvalue\\_reference](#)  
*is\_lvalue\_reference*
- struct [is\\_nothrow\\_constructible](#)  
*is\_nothrow\_constructible*
- struct [is\\_placeholder](#)  
*Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is. [TR1 3.6.2].*
- struct [is\\_placeholder< \\_Placeholder< \\_Num > >](#)
- struct [is\\_pod](#)  
*is\_pod*

- struct `is_reference`  
*is\_reference*
- struct `is_rvalue_reference`  
*is\_rvalue\_reference*
- struct `is_signed`  
*is\_signed*
- struct `is_standard_layout`  
*is\_standard\_layout*
- struct `is_trivial`  
*is\_trivial*
- struct `is_unsigned`  
*is\_unsigned*
- class `istream_iterator`  
*Provides input iterator semantics for streams.*
- class `istreambuf_iterator`  
*Provides input iterator semantics for streambufs.*
- struct `iterator`  
*Common iterator class.*
- struct `iterator_traits< _Tp * >`  
*Partial specialization for pointer types.*
- struct `iterator_traits< const _Tp * >`  
*Partial specialization for const pointer types.*
- class `length_error`
- struct `less`  
*One of the *comparison functors*.*
- struct `less_equal`  
*One of the *comparison functors*.*
- class `linear_congruential_engine`  
*A model of a linear congruential random number generator.*

- class [list](#)

*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

- class [locale](#)

*Container class for localization functionality.*

*The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.*

- class [lock\\_guard](#)

*Scoped lock idiom.*

- class [logic\\_error](#)

*One of two subclasses of exception.*

- struct [logical\\_and](#)

*One of the [Boolean operations functors](#).*

- struct [logical\\_not](#)

*One of the [Boolean operations functors](#).*

- struct [logical\\_or](#)

*One of the [Boolean operations functors](#).*

- class [lognormal\\_distribution](#)

*A [lognormal\\_distribution](#) random number distribution.*

- struct [make\\_signed](#)

*[make\\_signed](#)*

- struct [make\\_unsigned](#)

*[make\\_unsigned](#)*

- class [map](#)

*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

- class [mask\\_array](#)

*Reference to selected subset of an array.*

- class [match\\_results](#)

*The results of a match or search operation.*

- class `mem_fun1_ref_t`  
*One of the [adaptors](#) for member /// pointers.*
- class `mem_fun1_t`  
*One of the [adaptors](#) for member /// pointers.*
- class `mem_fun_ref_t`  
*One of the [adaptors](#) for member /// pointers.*
- class `mem_fun_t`  
*One of the [adaptors](#) for member /// pointers.*
- class `messages`  
*Primary class template [messages](#).  
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct `messages_base`  
*Messages facet base class providing catalog typedef.*
- class `messages_byname`  
*class [messages\\_byname](#) [22.2.7.2].*
- struct `minus`  
*One of the [math functors](#).*
- struct `modulus`  
*One of the [math functors](#).*
- class `money_base`  
*Money format ordering data.  
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the `part` enum. `symbol`, `sign`, and `value` must be present and the remaining field must contain either none or space.*
- class `money_get`  
*Primary class template [money\\_get](#).  
This facet encapsulates the code to parse and return a monetary amount from a string.*
- class `money_put`

Primary class template [money\\_put](#).

*This facet encapsulates the code to format and output a monetary amount.*

- class [moneypunct](#)

Primary class template [moneypunct](#).

*This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*

- class [moneypunct\\_byname](#)

class [moneypunct\\_byname](#) [22.2.6.4].

- class [move\\_iterator](#)

- class [multimap](#)

*A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.*

- struct [multiplies](#)

*One of the [math functors](#).*

- class [multiset](#)

*A standard container made up of elements, which can be retrieved in logarithmic time.*

- class [mutex](#)

*mutex*

- struct [negate](#)

*One of the [math functors](#).*

- class [negative\\_binomial\\_distribution](#)

*A [negative\\_binomial\\_distribution](#) random number distribution.*

- class [nested\\_exception](#)

*Exception class with `exception_ptr` data member.*

- class [normal\\_distribution](#)

*A normal continuous distribution for random numbers.*

- struct [not\\_equal\\_to](#)

*One of the [comparison functors](#).*

- class [num\\_get](#)

Primary class template [num\\_get](#).

*This facet encapsulates the code to parse and return a number from a string. It is used by the `istream` numeric extraction operators.*



- class `num_put`  
*Primary class template `num_put`.  
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- struct `numeric_limits`  
*Properties of fundamental types.*
- struct `numeric_limits< bool >`  
*`numeric_limits<bool>` specialization.*
- struct `numeric_limits< char >`  
*`numeric_limits<char>` specialization.*
- struct `numeric_limits< char16_t >`  
*`numeric_limits<char16_t>` specialization.*
- struct `numeric_limits< char32_t >`  
*`numeric_limits<char32_t>` specialization.*
- struct `numeric_limits< double >`  
*`numeric_limits<double>` specialization.*
- struct `numeric_limits< float >`  
*`numeric_limits<float>` specialization.*
- struct `numeric_limits< int >`  
*`numeric_limits<int>` specialization.*
- struct `numeric_limits< long >`  
*`numeric_limits<long>` specialization.*
- struct `numeric_limits< long double >`  
*`numeric_limits<long double>` specialization.*
- struct `numeric_limits< long long >`  
*`numeric_limits<long long>` specialization.*
- struct `numeric_limits< short >`  
*`numeric_limits<short>` specialization.*
- struct `numeric_limits< signed char >`

*numeric\_limits<signed char> specialization.*

- struct [numeric\\_limits< unsigned char >](#)  
*numeric\_limits<unsigned char> specialization.*
- struct [numeric\\_limits< unsigned int >](#)  
*numeric\_limits<unsigned int> specialization.*
- struct [numeric\\_limits< unsigned long >](#)  
*numeric\_limits<unsigned long> specialization.*
- struct [numeric\\_limits< unsigned long long >](#)  
*numeric\_limits<unsigned long long> specialization.*
- struct [numeric\\_limits< unsigned short >](#)  
*numeric\_limits<unsigned short> specialization.*
- struct [numeric\\_limits< wchar\\_t >](#)  
*numeric\_limits<wchar\_t> specialization.*
- class [numpunct](#)  
*Primary class template numpunct.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*
- class [numpunct\\_byname](#)  
*class numpunct\_byname [22.2.3.2].*
- struct [once\\_flag](#)  
*once\_flag*
- class [ostream\\_iterator](#)  
*Provides output iterator semantics for streams.*
- class [ostreambuf\\_iterator](#)  
*Provides output iterator semantics for streambufs.*
- class [out\\_of\\_range](#)
- struct [output\\_iterator\\_tag](#)  
*Marking output iterators.*

- class [overflow\\_error](#)
- struct [owner\\_less< shared\\_ptr< \\_Tp > >](#)  
*Partial specialization of [owner\\_less](#) for [shared\\_ptr](#).*
- struct [owner\\_less< weak\\_ptr< \\_Tp > >](#)  
*Partial specialization of [owner\\_less](#) for [weak\\_ptr](#).*
- class [packaged\\_task< \\_Res\(\\_ArgTypes...\)>](#)  
*[packaged\\_task](#)*
- struct [pair](#)  
*[pair](#) holds two objects of arbitrary type.*
- class [piecewise\\_constant\\_distribution](#)  
*A [piecewise\\_constant\\_distribution](#) random number distribution.*
- class [piecewise\\_linear\\_distribution](#)  
*A [piecewise\\_linear\\_distribution](#) random number distribution.*
- struct [plus](#)  
*One of the [math functors](#).*
- class [pointer\\_to\\_binary\\_function](#)  
*One of the [adaptors for function pointers](#).*
- class [pointer\\_to\\_unary\\_function](#)  
*One of the [adaptors for function pointers](#).*
- class [poisson\\_distribution](#)  
*A discrete Poisson random number distribution.*
- class [priority\\_queue](#)  
*A standard container automatically sorting its contents.*
- class [promise](#)  
*Primary template for [promise](#).*
- class [promise< \\_Res & >](#)  
*Partial specialization for [promise<R&>](#)*
- class [promise< void >](#)  
*Explicit specialization for [promise<void>](#)*

- class [queue](#)  
*A standard container giving FIFO behavior.*
- struct [random\\_access\\_iterator\\_tag](#)  
*Random-access iterators support a superset of bidirectional /// iterator operations.*
- class [random\\_device](#)
- class [range\\_error](#)
- struct [ratio](#)  
*Provides compile-time rational arithmetic.*
- struct [ratio\\_add](#)  
*ratio\_add*
- struct [ratio\\_divide](#)  
*ratio\_divide*
- struct [ratio\\_equal](#)  
*ratio\_equal*
- struct [ratio\\_multiply](#)  
*ratio\_multiply*
- struct [ratio\\_not\\_equal](#)  
*ratio\_not\_equal*
- struct [ratio\\_subtract](#)  
*ratio\_subtract*
- class [raw\\_storage\\_iterator](#)
- class [recursive\\_mutex](#)  
*recursive\_mutex*
- class [recursive\\_timed\\_mutex](#)  
*recursive\_timed\_mutex*
- class [reference\\_wrapper](#)  
*Primary class template for [reference\\_wrapper](#).*
- class [regex\\_error](#)  
*A regular expression exception class.  
The regular expression library throws objects of this class on error.*

- class [regex\\_iterator](#)
- class [regex\\_token\\_iterator](#)
- struct [regex\\_traits](#)  
*Describes aspects of a regular expression.*
- struct [remove\\_reference](#)  
*remove\_reference*
- class [reverse\\_iterator](#)
- class [runtime\\_error](#)  
*One of two subclasses of exception.*
- class [seed\\_seq](#)  
*The [seed\\_seq](#) class generates sequences of seeds for random number generators.*
- class [set](#)  
*A standard container made up of unique keys, which can be retrieved in logarithmic time.*
- class [shared\\_future](#)  
*Primary template for [shared\\_future](#).*
- class [shared\\_future< \\_Res & >](#)  
*Partial specialization for [shared\\_future<R&>](#)*
- class [shared\\_future< void >](#)  
*Explicit specialization for [shared\\_future<void>](#)*
- class [shared\\_ptr](#)  
*A smart pointer with reference-counted copy semantics.*
- class [shuffle\\_order\\_engine](#)  
*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.*
- class [slice](#)  
*Class defining one-dimensional subset of an array.*
- class [slice\\_array](#)  
*Reference to one-dimensional subset of an array.*
- class [stack](#)

*A standard container giving FILO behavior.*

- class [student\\_t\\_distribution](#)

*A [student\\_t\\_distribution](#) random number distribution.*

- class [sub\\_match](#)
- class [system\\_error](#)

*Thrown to indicate error code of underlying system.*

- class [thread](#)

*thread*

- class [time\\_base](#)

*Time format ordering data.*

*This class provides an enum representing different orderings of time: day, month, and year.*

- class [time\\_get](#)

*Primary class template [time\\_get](#).*

*This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*

- class [time\\_get\\_byname](#)

*class [time\\_get\\_byname](#) [22.2.5.2].*

- class [time\\_put](#)

*Primary class template [time\\_put](#).*

*This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*

- class [time\\_put\\_byname](#)

*class [time\\_put\\_byname](#) [22.2.5.4].*

- class [timed\\_mutex](#)

*timed\_mutex*

- struct [try\\_to\\_lock\\_t](#)

*Try to acquire ownership of the mutex without blocking.*

- class [tuple](#)

*tuple*

- class [tuple<\\_T1 >](#)

*tuple (1-element).*

- class `tuple< _T1, _T2 >`  
*tuple (2-element), with construction and assignment from a pair.*
- struct `tuple_element< 0, tuple< _Head, _Tail...> >`
- struct `tuple_element< __i, tuple< _Head, _Tail...> >`
- struct `tuple_size< tuple< _Elements...> >`  
*class tuple\_size*
- class `type_info`  
*Part of RTTI.*
- struct `unary_function`
- class `unary_negate`  
*One of the negation functors.*
- class `underflow_error`
- class `uniform_int_distribution`  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
- class `uniform_real_distribution`  
*Uniform continuous distribution for random numbers.*
- class `unique_lock`  
*unique\_lock*
- class `unique_ptr`  
*20.7.12.2 unique\_ptr for single objects.*
- class `unique_ptr< _Tp[ ], _Dp >`  
*20.7.12.3 unique\_ptr for array objects with a runtime length*
- class `unordered_map`  
*A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.*
- class `unordered_multimap`  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.*
- class `unordered_multiset`

*A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.*

- class [unordered\\_set](#)

*A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.*

- struct [uses\\_allocator](#)

*[allocator.uses.trait]*

- class [valarray](#)

*Smart array designed to support numeric processing.*

- class [vector](#)

*A standard container which offers fixed time access to individual elements in any order.*

- class [vector< bool, \\_Alloc >](#)

*A specialization of vector for booleans which offers fixed time access to individual elements in any order.*

- class [weak\\_ptr](#)

*A smart pointer with weak semantics.*

- class [weibull\\_distribution](#)

*A [weibull\\_distribution](#) random number distribution.*

## Typedefs

- typedef struct std::\_\_atomic\_flag\_base **\_\_atomic\_flag\_base**
- typedef FILE **\_\_c\_file**
- typedef \_\_locale\_t **\_\_c\_locale**
- typedef \_\_pthread\_mutex\_t **\_\_c\_lock**
- typedef unsigned long **\_Bit\_type**
- typedef [atomic\\_short](#) **atomic\_int\_fast16\_t**
- typedef [atomic\\_int](#) **atomic\_int\_fast32\_t**
- typedef [atomic\\_llong](#) **atomic\_int\_fast64\_t**
- typedef [atomic\\_schar](#) **atomic\_int\_fast8\_t**
- typedef [atomic\\_short](#) **atomic\_int\_least16\_t**
- typedef [atomic\\_int](#) **atomic\_int\_least32\_t**
- typedef [atomic\\_llong](#) **atomic\_int\_least64\_t**
- typedef [atomic\\_schar](#) **atomic\_int\_least8\_t**



- typedef [atomic\\_llong](#) **atomic\_intmax\_t**
- typedef [atomic\\_long](#) **atomic\_intptr\_t**
- typedef [atomic\\_long](#) **atomic\_ptrdiff\_t**
- typedef [atomic\\_ulong](#) **atomic\_size\_t**
- typedef [atomic\\_long](#) **atomic\_ssize\_t**
- typedef [atomic\\_ushort](#) **atomic\_uint\_fast16\_t**
- typedef [atomic\\_uint](#) **atomic\_uint\_fast32\_t**
- typedef [atomic\\_ullong](#) **atomic\_uint\_fast64\_t**
- typedef [atomic\\_uchar](#) **atomic\_uint\_fast8\_t**
- typedef [atomic\\_ushort](#) **atomic\_uint\_least16\_t**
- typedef [atomic\\_uint](#) **atomic\_uint\_least32\_t**
- typedef [atomic\\_ullong](#) **atomic\_uint\_least64\_t**
- typedef [atomic\\_uchar](#) **atomic\_uint\_least8\_t**
- typedef [atomic\\_ullong](#) **atomic\_uintmax\_t**
- typedef [atomic\\_ulong](#) **atomic\_uintptr\_t**
- typedef [match\\_results](#)< const char \* > **cmatch**
- typedef [regex\\_iterator](#)< const char \* > **cregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const char \* > **cregex\_token\_iterator**
- typedef [sub\\_match](#)< const char \* > **csub\_match**
- typedef [minstd\\_rand0](#) **default\_random\_engine**
- typedef [basic\\_filebuf](#)< char > **filebuf**
- typedef [basic\\_fstream](#)< char > **fstream**
- typedef [basic\\_ifstream](#)< char > **ifstream**
- typedef [basic\\_ios](#)< char > **ios**
- typedef [basic\\_iostream](#)< char > **iostream**
- typedef [basic\\_istream](#)< char > **istream**
- typedef [basic\\_istream](#)< char > **istream**
- typedef [basic\\_istream](#)< char > **istream**
- typedef [basic\\_istream](#)< char > **istream**
- typedef [shuffle\\_order\\_engine](#)< [minstd\\_rand0](#), 256 > **knuth\_b**
- typedef enum [std::memory\\_order](#) **memory\_order**
- typedef [linear\\_congruential\\_engine](#)< [uint\\_fast32\\_t](#), 48271UL, 0UL, 2147483647UL > **minstd\_rand**
- typedef [linear\\_congruential\\_engine](#)< [uint\\_fast32\\_t](#), 16807UL, 0UL, 2147483647UL > **minstd\_rand0**
- typedef [mersenne\\_twister\\_engine](#)< [uint\\_fast32\\_t](#), 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > **mt19937**
- typedef [mersenne\\_twister\\_engine](#)< [uint\\_fast64\\_t](#), 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > **mt19937\_64**
- typedef void(\* [new\\_handler](#) )()
- typedef [basic\\_ofstream](#)< char > **ofstream**
- typedef [basic\\_ostream](#)< char > **ostream**

- typedef [basic\\_ostringstream](#)< char > [ostringstream](#)
- typedef [\\_\\_PTRDIFF\\_TYPE\\_\\_](#) [ptrdiff\\_t](#)
- typedef [discard\\_block\\_engine](#)< [ranlux24\\_base](#), 223, 23 > [ranlux24](#)
- typedef [subtract\\_with\\_carry\\_engine](#)< [uint\\_fast32\\_t](#), 24, 10, 24 > [ranlux24\\_base](#)
- typedef [discard\\_block\\_engine](#)< [ranlux48\\_base](#), 389, 11 > [ranlux48](#)
- typedef [subtract\\_with\\_carry\\_engine](#)< [uint\\_fast64\\_t](#), 48, 5, 12 > [ranlux48\\_base](#)
  
- typedef [basic\\_regex](#)< char > [regex](#)
- typedef [\\_\\_SIZE\\_TYPE\\_\\_](#) [size\\_t](#)
- typedef [match\\_results](#)< [string::const\\_iterator](#) > [smatch](#)
- typedef [regex\\_iterator](#)< [string::const\\_iterator](#) > [sregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< [string::const\\_iterator](#) > [sregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< [string::const\\_iterator](#) > [ssub\\_match](#)
- typedef [basic\\_streambuf](#)< char > [streambuf](#)
- typedef long long [streamoff](#)
- typedef [fpos](#)< [mbstate\\_t](#) > [streampos](#)
- typedef [ptrdiff\\_t](#) [streamsize](#)
- typedef [basic\\_string](#)< char > [string](#)
- typedef [basic\\_stringbuf](#)< char > [stringbuf](#)
- typedef [basic\\_stringstream](#)< char > [stringstream](#)
- typedef void(\* [terminate\\_handler](#) )()
- typedef [fpos](#)< [mbstate\\_t](#) > [u16streampos](#)
- typedef [basic\\_string](#)< [char16\\_t](#) > [u16string](#)
- typedef [fpos](#)< [mbstate\\_t](#) > [u32streampos](#)
- typedef [basic\\_string](#)< [char32\\_t](#) > [u32string](#)
- typedef void(\* [unexpected\\_handler](#) )()
- typedef [match\\_results](#)< const [wchar\\_t](#) \* > [wcmatch](#)
- typedef [regex\\_iterator](#)< const [wchar\\_t](#) \* > [wcregex\\_iterator](#)
- typedef [regex\\_token\\_iterator](#)< const [wchar\\_t](#) \* > [wcregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const [wchar\\_t](#) \* > [wcs\\_sub\\_match](#)
- typedef [basic\\_filebuf](#)< [wchar\\_t](#) > [wfilebuf](#)
- typedef [basic\\_fstream](#)< [wchar\\_t](#) > [wfstream](#)
- typedef [basic\\_ifstream](#)< [wchar\\_t](#) > [wifstream](#)
- typedef [basic\\_ios](#)< [wchar\\_t](#) > [wios](#)
- typedef [basic\\_iostream](#)< [wchar\\_t](#) > [wiostream](#)
- typedef [basic\\_istream](#)< [wchar\\_t](#) > [wistream](#)
- typedef [basic\\_istreamstream](#)< [wchar\\_t](#) > [wistreamstream](#)
- typedef [basic\\_ofstream](#)< [wchar\\_t](#) > [wofstream](#)
- typedef [basic\\_ostream](#)< [wchar\\_t](#) > [wostream](#)
- typedef [basic\\_ostreamstream](#)< [wchar\\_t](#) > [wostreamstream](#)
- typedef [basic\\_regex](#)< [wchar\\_t](#) > [wregex](#)
- typedef [match\\_results](#)< [wstring::const\\_iterator](#) > [wsmatch](#)

- typedef [regex\\_iterator](#)< wstring::const\_iterator > **wsregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< wstring::const\_iterator > **wsregex\_token\_iterator**
- typedef [sub\\_match](#)< wstring::const\_iterator > **wssub\_match**
- typedef [basic\\_streambuf](#)< wchar\_t > **wstreambuf**
- typedef [fpos](#)< mbstate\_t > **wstreampos**
- typedef [basic\\_string](#)< wchar\_t > **wstring**
- typedef [basic\\_stringbuf](#)< wchar\_t > **wstringbuf**
- typedef [basic\\_stringstream](#)< wchar\_t > **wstringstream**

## Enumerations

- enum { **\_S\_threshold** }
- enum { **\_S\_chunk\_size** }
- enum { **\_S\_word\_bit** }
- enum **\_Ios\_Fmtflags** {  
**\_S\_boolalpha, \_S\_dec, \_S\_fixed, \_S\_hex,**  
**\_S\_internal, \_S\_left, \_S\_oct, \_S\_right,**  
**\_S\_scientific, \_S\_showbase, \_S\_showpoint, \_S\_showpos,**  
**\_S\_skipws, \_S\_unitbuf, \_S\_uppercase, \_S\_adjustfield,**  
**\_S\_basefield, \_S\_floatfield, \_S\_ios\_fmtflags\_end** }
- enum **\_Ios\_Iostate** {  
**\_S\_goodbit, \_S\_badbit, \_S\_eofbit, \_S\_failbit,**  
**\_S\_ios\_iostate\_end** }
- enum **\_Ios\_Openmode** {  
**\_S\_app, \_S\_ate, \_S\_bin, \_S\_in,**  
**\_S\_out, \_S\_trunc, \_S\_ios\_openmode\_end** }
- enum **\_Ios\_Seekdir** { **\_S\_beg, \_S\_cur, \_S\_end, \_S\_ios\_seekdir\_end** }
- enum **\_Manager\_operation** { **\_\_get\_type\_info, \_\_get\_functor\_ptr, \_\_clone\_functor, \_\_destroy\_functor** }
- enum **\_Rb\_tree\_color** { **\_S\_red, \_S\_black** }
- enum **cv\_status** { **no\_timeout, timeout** }
- enum **errc** {  
**address\_family\_not\_supported, address\_in\_use, address\_not\_available,**  
**already\_connected,**  
**argument\_list\_too\_long, argument\_out\_of\_domain, bad\_address, bad\_file\_descriptor,**  
**bad\_message, broken\_pipe, connection\_aborted, connection\_already\_in\_progress,**

connection\_refused, connection\_reset, cross\_device\_link, destination\_ -  
 address\_required,  
 device\_or\_resource\_busy, directory\_not\_empty, executable\_format\_error,  
 file\_exists,  
 file\_too\_large, filename\_too\_long, function\_not\_supported, host\_ -  
 unreachable,  
 identifier\_removed, illegal\_byte\_sequence, inappropriate\_io\_control\_ -  
 operation, interrupted,  
 invalid\_argument, invalid\_seek, io\_error, is\_a\_directory,  
 message\_size, network\_down, network\_reset, network\_unreachable,  
 no\_buffer\_space, no\_child\_process, no\_link, no\_lock\_available,  
 no\_message\_available, no\_message, no\_protocol\_option, no\_space\_on\_ -  
 device,  
 no\_stream\_resources, no\_such\_device\_or\_address, no\_such\_device, no\_ -  
 such\_file\_or\_directory,  
 no\_such\_process, not\_a\_directory, not\_a\_socket, not\_a\_stream,  
 not\_connected, not\_enough\_memory, not\_supported, operation\_canceled,  
 operation\_in\_progress, operation\_not\_permitted, operation\_not\_ -  
 supported, operation\_would\_block,  
 owner\_dead, permission\_denied, protocol\_error, protocol\_not\_supported,  
 read\_only\_file\_system, resource\_deadlock\_would\_occur, resource\_ -  
 unavailable\_try\_again, result\_out\_of\_range,  
 state\_not\_recoverable, stream\_timeout, text\_file\_busy, timed\_out,  
 too\_many\_files\_open\_in\_system, too\_many\_files\_open, too\_many\_links,  
 too\_many\_symbolic\_link\_levels,  
 value\_too\_large, wrong\_protocol\_type }

- enum `float_denorm_style` { `denorm_indeterminate`, `denorm_absent`, `denorm_ -`  
`present` }
- enum `float_round_style` {  
`round_indeterminate`, `round_toward_zero`, `round_to_nearest`, `round_toward_ -`  
`infinity`,  
`round_toward_neg_infinity` }
- enum `future_errc` { `broken_promise`, `future_already_retrieved`, `promise_ -`  
`already_satisfied`, `no_state` }
- enum `launch` { `any`, `async`, `sync` }
- enum `memory_order` {  
`memory_order_relaxed`, `memory_order_consume`, `memory_order_ -`  
`acquire`, `memory_order_release`,  
`memory_order_acq_rel`, `memory_order_seq_cst` }

## Functions

- `template<typename _CharT >`  
`_CharT * __add_grouping (_CharT *__s, _CharT __sep, const char *__gbeg,`  
`size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _Tp >`  
`_Tp * __addressof (_Tp &__r)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void __adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex,`  
`_Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _`  
`Compare >`  
`void __adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex,`  
`_Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _InputIterator, typename _Distance >`  
`void __advance (_InputIterator &__i, _Distance __n, input\_iterator\_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`  
`void __advance (_BidirectionalIterator &__i, _Distance __n, bidirectional\_`  
`iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __advance (_RandomAccessIterator &__i, _Distance __n, random\_`  
`access\_iterator\_tag)`
- `template<typename _Tp, _Lock_policy _Lp, typename _Alloc, typename... _Args>`  
`__shared_ptr<_Tp, _Lp> __allocate_shared (_Alloc __a, _Args &&... __args)`
- `void __atomic_flag_wait_explicit (__atomic_flag_base *, memory\_order) _`  
`GLIBCXX_NOTHROW`
- `__attribute__((__pure__)) _Rb_tree_node_base * _Rb_tree_increment(_Rb_`  
`tree_node_base *__x) throw ()`
- `__attribute__((__const__)) __atomic_flag_base * __atomic_flag_for_`  
`address(const void *__z) _GLIBCXX_NOTHROW`
- `memory\_order __calculate_memory_order (memory\_order __m)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > __callable_functor (_Member _`  
`Class::*const &__p)`
- `template<typename _Functor >`  
`_Functor & __callable_functor (_Functor &__f)`
- `template<typename _Member, typename _Class >`  
`_Mem_fn< _Member _Class::* > __callable_functor (_Member _Class::*&__`  
`__p)`
- `template<typename _Facet >`  
`const _Facet & __check_facet (const _Facet *__f)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _`  
`RandomAccessIterator __last, _Distance __chunk_size)`

- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`void __chunk_insertion_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)`
- `template<typename _Tp >`  
`_Tp __complex_abs (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp __complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_pow (const complex< _Tp > &__x, const com-`  
`plex< _Tp > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tanh (const complex< _Tp > &__z)`
- `int __convert_from_v (const __c_locale &__cloc __attribute__((__unused__)),`  
`char *__out, const int __size __attribute__((__unused__)), const char *__fmt,...)`
- `template<typename _Tp >`  
`void __convert_to_v (const char *, _Tp &, ios\_base::iostate &, const __c_locale`  
`&) throw ()`
- `template<>`  
`void __convert_to_v (const char *, float &, ios\_base::iostate &, const __c_-`  
`locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, double &, ios\_base::iostate &, const __c_-`  
`locale &) throw ()`

- `template<>`  
`void __convert_to_v (const char *, long double &, ios_base::iostate &, const`  
`__c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_-`  
`iterator< _CharT > >::__type __copy_move_a2 (_CharT *__first, _CharT *_-`  
`__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_-`  
`iterator< _CharT > >::__type __copy_move_a2 (const _CharT *__first, const`  
`_CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__-`  
`type __copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_-`  
`iterator< _CharT > __last, _CharT *__result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_-`  
`iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (_-`  
`CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_-`  
`iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (const`  
`_CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT`  
`> >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__-`  
`type __copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT >`  
`>, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`_OI __copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 __copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`_BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator _-`  
`__result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`_OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _-`  
`OutputIterator __result, random_access_iterator_tag)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs (basic_streambuf< _CharT, _Traits > *__sbin,`  
`basic_streambuf< _CharT, _Traits > *__sbout)`

- `template<>`  
[streamsize](#) **\_\_copy\_streambufs\_eof** ([basic\\_streambuf](#)< char > \*\_\_sbin, [basic\\_streambuf](#)< char > \*\_\_sbout, bool &\_\_ineof)
- `template<>`  
[streamsize](#) **\_\_copy\_streambufs\_eof** ([basic\\_streambuf](#)< wchar\_t > \*\_\_sbin, [basic\\_streambuf](#)< wchar\_t > \*\_\_sbout, bool &\_\_ineof)
- `template<typename _CharT, typename _Traits >`  
[streamsize](#) **\_\_copy\_streambufs\_eof** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*, [basic\\_streambuf](#)< \_CharT, \_Traits > \*, bool &)
- `size_t` **\_\_deque\_buf\_size** (size\_t \_\_size)
- `template<typename _RandomAccessIterator >`  
`iterator_traits< _RandomAccessIterator >::difference_type` **\_\_distance** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, [random\\_access\\_iterator\\_tag](#))
- `template<typename _InputIterator >`  
`iterator_traits< _InputIterator >::difference_type` **\_\_distance** (\_InputIterator \_\_first, \_InputIterator \_\_last, [input\\_iterator\\_tag](#))
- `template<_Lock_policy _Lp, typename _Tp1, typename _Tp2 >`  
void **\_\_enable\_shared\_from\_this\_helper** (const \_\_shared\_count< \_Lp > &, const \_\_enable\_shared\_from\_this< \_Tp1, \_Lp > \*, const \_Tp2 \*)
- `template<typename _Tp1, typename _Tp2 >`  
void **\_\_enable\_shared\_from\_this\_helper** (const \_\_shared\_count<> &, const [enable\\_shared\\_from\\_this](#)< \_Tp1 > \*, const \_Tp2 \*)
- `template<_Lock_policy _Lp>`  
void **\_\_enable\_shared\_from\_this\_helper** (const \_\_shared\_count< \_Lp > &,...)
  
- `template<typename _II1, typename _II2 >`  
bool **\_\_equal\_aux** (\_II1 \_\_first1, \_II1 \_\_last1, \_II2 \_\_first2)
- `template<typename _ForwardIterator, typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)<!\_\_is\_scalar< \_Tp >::\_\_value, void >::\_\_type **\_\_fill\_a** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_value)
  
- `template<typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< \_\_is\_byte< \_Tp >::\_\_value, void >::\_\_type **\_\_fill\_a** (\_Tp \*\_\_first, \_Tp \*\_\_last, const \_Tp &\_\_c)
- `void` **\_\_fill\_bvector** (\_Bit\_iterator \_\_first, \_Bit\_iterator \_\_last, bool \_\_x)
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)<!\_\_is\_scalar< \_Tp >::\_\_value, \_OutputIterator >::\_\_type **\_\_fill\_n\_a** (\_OutputIterator \_\_first, \_Size \_\_n, const \_Tp &\_\_value)
- `template<typename _Size, typename _Tp >`  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< \_\_is\_byte< \_Tp >::\_\_value, \_Tp \* >::\_\_type **\_\_fill\_n\_a** (\_Tp \*\_\_first, \_Size \_\_n, const \_Tp &\_\_c)
- `template<typename _RandomAccessIterator >`  
void **\_\_final\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)



- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_final\_insertion\_sort (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator \_\_find (_InputIterator __first, _InputIterator __last, const _Tp &_-`  
`__val, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator \_\_find (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, const _Tp &__val, random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1 \_\_find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_-`  
`iterator\_tag, forward\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >`  
`_BidirectionalIterator1 \_\_find\_end (_BidirectionalIterator1 __first1,`  
`_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_-`  
`iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`  
`BinaryPredicate >`  
`_BidirectionalIterator1 \_\_find\_end (_BidirectionalIterator1 __first1,`  
`_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`  
`BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_-`  
`iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`  
`>`  
`_ForwardIterator1 \_\_find\_end (_ForwardIterator1 __first1, _ForwardIterator1`  
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_-`  
`iterator\_tag, forward\_iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator \_\_find\_if (_InputIterator __first, _InputIterator __last, _Predicate`  
`__pred, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator \_\_find\_if (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator \_\_find\_if\_not (_InputIterator __first, _InputIterator __last, _-`  
`Predicate __pred, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`_RandomAccessIterator \_\_find\_if\_not (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _EuclideanRingElement >`  
`_EuclideanRingElement \_\_gcd (_EuclideanRingElement __m, _-`  
`EuclideanRingElement __n)`

- `template<std::size_t __i, typename _Head, typename... _Tail>  
__add_ref< _Head >::type __get_helper (_Tuple_impl< __i, _Head, _Tail...>  
&__t)`
- `template<std::size_t __i, typename _Head, typename... _Tail>  
__add_c_ref< _Head >::type __get_helper (const _Tuple_impl< __i, _Head,  
_Tail...> &__t)`
- `template<typename _Ex >  
const nested_exception * __get_nested_exception (const _Ex &__ex)`
- `mutex & __get_once_mutex ()`
- `template<typename _RandomAccessIterator >  
void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator __-  
_middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >  
void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator __-  
_middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp >  
size_t __iconv_adapter (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t  
*, iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t  
*__outbytes)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >  
_ForwardIterator __inplace_stable_partition (_ForwardIterator __first, _-  
ForwardIterator __last, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator >  
void __inplace_stable_sort (_RandomAccessIterator __first, _-  
RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >  
void __inplace_stable_sort (_RandomAccessIterator __first, _-  
RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >  
void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator  
__last)`
- `template<typename _RandomAccessIterator, typename _Compare >  
void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator  
__last, _Compare __comp)`
- `template<typename _CharT, typename _ValueT >  
_GLIBCXX_END_LDBL_NAMESPACE int __int_to_char (_CharT *__-  
bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __-  
_dec)`
- `template<typename _RandomAccessIterator, typename _Size >  
void __introspect (_RandomAccessIterator __first, _RandomAccessIterator __-  
_nth, _RandomAccessIterator __last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >  
void __introspect (_RandomAccessIterator __first, _RandomAccessIterator __-  
_nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size >`  
`void \_\_introsort\_loop (_RandomAccessIterator __first, _RandomAccessIterator`  
`__last, _Size __depth_limit)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _Compare __comp, _Distance`  
`__n)`
- `template<typename _RandomAccessIterator >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`bool \_\_is\_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`_Distance \_\_is\_heap\_until (_RandomAccessIterator __first, _Distance __n, _-`  
`Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`_Distance \_\_is\_heap\_until (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _Iter >`  
`iterator_traits< _Iter >::iterator_category \_\_iterator\_category (const _Iter &)`
- `template<typename _II1, typename _II2 >`  
`bool \_\_lexicographical\_compare\_aux (_II1 __first1, _II1 __last1, _II2 __first2,`  
`__II2 __last2)`
- `template<typename _Size >`  
`_Size \_\_lg (_Size __n)`
- `int \_\_lg (int __n)`
- `long long \_\_lg (long long __n)`
- `long \_\_lg (long __n)`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args>`  
`__shared_ptr< _Tp, _Lp > \_\_make\_shared (_Args &&... __args)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`  
`void \_\_merge\_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`__middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _-`  
`Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _-`  
`Compare >`  
`void \_\_merge\_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`__middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _-`  
`Pointer __buffer, _Distance __buffer_size, _Compare __comp)`

- `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 , typename _Compare >`  
`__merge_backward` (`_BidirectionalIterator1` \_\_first1, `_BidirectionalIterator1` \_\_last1, `_BidirectionalIterator2` \_\_first2, `_BidirectionalIterator2` \_\_last2, `_BidirectionalIterator3` \_\_result, `_Compare` \_\_comp)
- `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 >`  
`__merge_backward` (`_BidirectionalIterator1` \_\_first1, `_BidirectionalIterator1` \_\_last1, `_BidirectionalIterator2` \_\_first2, `_BidirectionalIterator2` \_\_last2, `_BidirectionalIterator3` \_\_result)
- `template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance >`  
`void __merge_sort_loop` (`_RandomAccessIterator1` \_\_first, `_RandomAccessIterator1` \_\_last, `_RandomAccessIterator2` \_\_result, `_Distance` \_\_step\_size)
- `template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance , typename _Compare >`  
`void __merge_sort_loop` (`_RandomAccessIterator1` \_\_first, `_RandomAccessIterator1` \_\_last, `_RandomAccessIterator2` \_\_result, `_Distance` \_\_step\_size, `_Compare` \_\_comp)
- `template<typename _RandomAccessIterator , typename _Pointer >`  
`void __merge_sort_with_buffer` (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last, `_Pointer` \_\_buffer)
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Compare >`  
`void __merge_sort_with_buffer` (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last, `_Pointer` \_\_buffer, `_Compare` \_\_comp)
- `template<typename _BidirectionalIterator , typename _Distance >`  
`void __merge_without_buffer` (`_BidirectionalIterator` \_\_first, `_BidirectionalIterator` \_\_middle, `_BidirectionalIterator` \_\_last, `_Distance` \_\_len1, `_Distance` \_\_len2)
- `template<typename _BidirectionalIterator , typename _Distance , typename _Compare >`  
`void __merge_without_buffer` (`_BidirectionalIterator` \_\_first, `_BidirectionalIterator` \_\_middle, `_BidirectionalIterator` \_\_last, `_Distance` \_\_len1, `_Distance` \_\_len2, `_Compare` \_\_comp)
- `template<typename _Iterator >`  
`_Miter_base< _Iterator >::iterator_type __miter_base` (`_Iterator` \_\_it)
- `template<typename _Iterator >`  
`void __move_median_first` (`_Iterator` \_\_a, `_Iterator` \_\_b, `_Iterator` \_\_c)
- `template<typename _Iterator , typename _Compare >`  
`void __move_median_first` (`_Iterator` \_\_a, `_Iterator` \_\_b, `_Iterator` \_\_c, `_Compare` \_\_comp)
- `template<typename _Iterator >`  
`_Niter_base< _Iterator >::iterator_type __niter_base` (`_Iterator` \_\_it)
- `void __once_proxy` ()

- `template<typename _CharT, typename _Traits >`  
`void __ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize _-`  
`__n)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & __ostream_insert (basic_ostream< _-`  
`CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template ostream & __ostream_insert (ostream &, const char *, streamsize)`
- `template wostream & __ostream_insert (wostream &, const wchar_t *, stream-`  
`size)`
- `template<typename _CharT, typename _Traits >`  
`void __ostream_write (basic_ostream< _CharT, _Traits > &__out, const _-`  
`CharT *__s, streamsize __n)`
- `template<typename _BidirectionalIterator, typename _Predicate >`  
`_BidirectionalIterator __partition (_BidirectionalIterator __first, _-`  
`BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last,`  
`_Predicate __pred, forward_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _RandomAccessIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`  
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _-`  
`Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _-`  
`Compare >`  
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _-`  
`Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,`  
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last,`  
`random_access_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`void __rotate (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`  
`void __rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`  
`ForwardIterator __last, forward_iterator_tag)`

- `template<typename _BidirectionalIterator >`  
`void \_\_rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _-`  
`BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _-`  
`Distance >`  
`_BidirectionalIterator1 \_\_rotate\_adaptive (_BidirectionalIterator1 __first, _-`  
`BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __-`  
`len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_-`  
`size)`
- `template<typename _ForwardIterator , typename _Integer , typename _Tp >`  
`_ForwardIterator \_\_search\_n (_ForwardIterator __first, _ForwardIterator __last,`  
`_Integer __count, const _Tp &__val, std::forward\_iterator\_tag)`
- `template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _-`  
`BinaryPredicate >`  
`_ForwardIterator \_\_search\_n (_ForwardIterator __first, _ForwardIterator __-`  
`last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`  
`std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter , typename _Integer , typename _Tp >`  
`_RandomAccessIter \_\_search\_n (_RandomAccessIter __first, _-`  
`RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random\_-`  
`access\_iterator\_tag)`
- `template<typename _RandomAccessIter , typename _Integer , typename _Tp , typename _-`  
`BinaryPredicate >`  
`_RandomAccessIter \_\_search\_n (_RandomAccessIter __first, _-`  
`RandomAccessIter __last, _Integer __count, const _Tp &__val, _-`  
`BinaryPredicate __binary_pred, std::random\_access\_iterator\_tag)`
- `void \_\_set\_once\_functor\_lock\_ptr (unique\_lock< mutex > *)`
- `template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _-`  
`Distance >`  
`_ForwardIterator \_\_stable\_partition\_adaptive (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer,`  
`_Distance __buffer_size)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Distance >`  
`void \_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Distance , typename`  
`_Compare >`  
`void \_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size,`  
`_Compare __comp)`
- `void \_\_throw\_bad\_alloc (void) __attribute__((__noreturn__))`
- `void \_\_throw\_bad\_cast (void) __attribute__((__noreturn__))`
- `void \_\_throw\_bad\_exception (void) __attribute__((__noreturn__))`
- `void \_\_throw\_bad\_function\_call () __attribute__((__noreturn__))`

- void **\_\_throw\_bad\_typeid** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_domain\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_future\_error** (int) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_invalid\_argument** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_ios\_failure** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_length\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_logic\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_out\_of\_range** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_overflow\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_range\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_regex\_error** (regex\_constants::error\_type \_\_ecode)
- void **\_\_throw\_runtime\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_system\_error** (int) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **\_\_throw\_underflow\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- template<typename \_Ex >  
void **\_\_throw\_with\_nested** (\_Ex &&, const nested\_exception \*=0) \_\_attribute\_\_((\_\_noreturn\_\_))
- template<typename \_Ex >  
void **\_\_throw\_with\_nested** (\_Ex &&,...) \_\_attribute\_\_((\_\_noreturn\_\_))
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
**tuple**< \_TElements..., \_UElements...> **\_\_tuple\_cat\_helper** (const **tuple**< \_TElements...> &\_\_t, const \_\_index\_holder< \_TIdx...> &, const **tuple**< \_UElements...> &\_\_u, const \_\_index\_holder< \_UIdx...> &)
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
**tuple**< \_TElements..., \_UElements...> **\_\_tuple\_cat\_helper** (**tuple**< \_TElements...> &&\_\_t, const \_\_index\_holder< \_TIdx...> &, const **tuple**< \_UElements...> &\_\_u, const \_\_index\_holder< \_UIdx...> &)
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
**tuple**< \_TElements..., \_UElements...> **\_\_tuple\_cat\_helper** (const **tuple**< \_TElements...> &\_\_t, const \_\_index\_holder< \_TIdx...> &, **tuple**< \_UElements...> &&\_\_u, const \_\_index\_holder< \_UIdx...> &)
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
**tuple**< \_TElements..., \_UElements...> **\_\_tuple\_cat\_helper** (**tuple**< \_TElements...> &&\_\_t, const \_\_index\_holder< \_TIdx...> &, **tuple**< \_UElements...> &&\_\_u, const \_\_index\_holder< \_UIdx...> &)
- template<typename \_RandomAccessIterator >  
void **\_\_unguarded\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- template<typename \_RandomAccessIterator, typename \_Compare >  
void **\_\_unguarded\_insertion\_sort** (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)

- `template<typename _RandomAccessIterator >`  
`void \_\_unguarded\_linear\_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_unguarded\_linear\_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Tp >`  
`_RandomAccessIterator \_\_unguarded\_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator, typename _Tp, typename _Compare >`  
`_RandomAccessIterator \_\_unguarded\_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator \_\_unguarded\_partition\_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator \_\_unguarded\_partition\_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void \_\_uninitialized\_construct\_buf (_ForwardIterator __first, _ForwardIterator __last, _Tp &__value)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_copy\_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator \_\_uninitialized\_copy\_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator<_Tp> &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator \_\_uninitialized\_copy\_move (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator \_\_uninitialized\_copy\_n (_InputIterator __first, _Size __n, _ForwardIterator __result, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator \_\_uninitialized\_copy\_n (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random\_access\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void \_\_uninitialized\_default (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void \_\_uninitialized\_default\_a (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`



- `template<typename _ForwardIterator, typename _Tp >`  
`void __uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __-`  
`last, allocator< _Tp > &)`
- `template<typename _ForwardIterator, typename _Size >`  
`void __uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator >`  
`void __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _-`  
`Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`void __uninitialized_default_n_a (_ForwardIterator __first, _Size __n, allocat-`  
`or< _Tp > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`  
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`  
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _-`  
`Allocator >`  
`_ForwardIterator __uninitialized_fill_move (_ForwardIterator __result, _-`  
`ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator`  
`__last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`  
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp`  
`&__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`  
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp`  
`&__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator __uninitialized_move_a (_InputIterator __first, _InputIterator`  
`__last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`  
`name _Allocator >`  
`_ForwardIterator __uninitialized_move_copy (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _-`  
`Allocator >`  
`void __uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1,`  
`_ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator`  
`&__alloc)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _-`  
`OutputIterator __result, input_iterator_tag, output_iterator_tag)`

- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >  
_OutputIterator \_\_unique\_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward\_iterator\_tag, output\_iterator\_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >  
_OutputIterator \_\_unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input\_iterator\_tag, output\_iterator\_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >  
_ForwardIterator \_\_unique\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input\_iterator\_tag, forward\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator >  
_OutputIterator \_\_unique\_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward\_iterator\_tag, output\_iterator\_tag)`
- `template<typename _InputIterator, typename _ForwardIterator >  
_ForwardIterator \_\_unique\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, input\_iterator\_tag, forward\_iterator\_tag)`
- `template<typename _Bi_iter >  
const sub\_match<_Bi_iter > & \_\_unmatched\_sub ()`
- `template<typename _Tp >  
void \_\_valarray\_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp >  
void \_\_valarray\_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp >  
void \_\_valarray\_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp >  
void \_\_valarray\_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >  
void \_\_valarray\_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >  
void \_\_valarray\_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >  
void \_\_valarray\_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >  
void \_\_valarray\_copy (_Array<_Tp > __a, size_t __n, _Array<_Tp > __b)`

- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`

- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * __valarray_get_memory (size_t __n)`

- `template<typename _Tp >`  
`_Tp *__restrict__ __valarray_get_storage (size_t __n)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `template<typename _Tp >`  
`_Tp __valarray_product (const _Tp *__f, const _Tp *__l)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `_GLIBCXX_PURE bool __verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`
- `template<size_t _Ind, typename... _Tp>`  
`auto __volget (const volatile tuple< _Tp...> &__tuple)-> typename tuple_element< _Ind`
- `template<size_t _Ind, typename... _Tp>`  
`auto __volget (volatile tuple< _Tp...> &__tuple)-> typename tuple_element< _Ind`
- `template<typename _CharT >`  
`ostreambuf_iterator< _CharT > __write (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _OutIter >`  
`_OutIter __write (_OutIter __s, const _CharT *__ws, int __len)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool >`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp`  
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _-`  
`Tp &__t)`

- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`



- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b,`  
`size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t >`  
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __-`  
`m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _-`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __-`  
`m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _-`  
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t >`  
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _-`  
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp`  
`&__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s,`  
`_Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b,`  
`size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m,`  
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i,`  
`_Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array<`  
`_Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _`  
`Array< _Tp > __b)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _-`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t >`  
`__i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t`  
`> __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool >`  
`__m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _T1, typename... _Args>`  
`void _Construct (_T1 *__p, _Args &&...__args)`
- `template<typename _Tp >`  
`void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator`  
`&__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< _-`  
`Tp > &)`
- `size_t _Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t _Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_-`  
`node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header)`  
`throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base`  
`*const __z, _Rb_tree_node_base &__header) throw ()`
- `void abort (void) _GLIBCXX_NORETURN throw ()`

- float **abs** (float \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **abs**  
(\_Tp \_\_x)
- double **abs** (double \_\_x)
- long double **abs** (long double \_\_x)
- template<typename \_Tp >  
\_Tp **abs** (const [complex](#)< \_Tp > &)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Abs, \_Expr, \_Dom >, typename \_Dom::value\_type > **abs**  
(const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Abs, \_ValArray, \_Tp >, \_Tp > **abs** (const [valarray](#)< \_Tp  
> &\_\_v)
- template<typename \_InputIterator, typename \_Tp >  
\_Tp [accumulate](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Tp \_\_init)
- template<typename \_InputIterator, typename \_Tp, typename \_BinaryOperation >  
\_Tp [accumulate](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Tp \_\_init, \_-  
BinaryOperation \_\_binary\_op)
- float **acos** (float \_\_x)
- long double **acos** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**acos** (\_Tp \_\_x)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Acos, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**acos** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Acos, \_ValArray, \_Tp >, \_Tp > **acos** (const [valarray](#)< \_Tp  
> &\_\_v)
- template<typename \_Tp >  
\_Tp \* [addressof](#) (\_Tp &\_\_r)
- template<typename \_InputIterator, typename \_OutputIterator >  
\_OutputIterator [adjacent\\_difference](#) (\_InputIterator \_\_first, \_InputIterator \_\_last,  
\_OutputIterator \_\_result)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_BinaryOperation >  
\_OutputIterator [adjacent\\_difference](#) (\_InputIterator \_\_first, \_InputIterator \_\_last,  
\_OutputIterator \_\_result, \_BinaryOperation \_\_binary\_op)
- template<typename \_Filter >  
\_Filter **adjacent\_find** (\_Filter, \_Filter)
- template<typename \_Filter, typename \_BinaryPredicate >  
\_Filter **adjacent\_find** (\_Filter, \_Filter, \_BinaryPredicate)
- template<typename \_ForwardIterator >  
\_ForwardIterator [adjacent\\_find](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_-  
last)

- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __-`  
`last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Distance >`  
`void advance (_InputIterator &__i, _Distance __n)`
- `template<typename _Iter, typename _Predicate >`  
`bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _Tp > allocate_shared (_Alloc __a, _Args &&...__args)`
- `template<typename _Iter, typename _Predicate >`  
`bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`_Tp arg (const complex< _Tp > &)`
- `float asin (float __x)`
- `long double asin (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`asin (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > asin`  
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp`  
`> &__v)`
- `template<typename _Fn, typename... _Args>`  
`future< typename result_of< _Fn(_Args...)>::type > async (launch __policy,`  
`_Fn &&__fn, _Args &&...__args)`
- `template<typename _Fn, typename... _Args>`  
`enable_if<!is_same< typename decay< _Fn >::type, launch >::value, future<`  
`decltype(std::declval< _Fn >)(std::declval< _Args >...)> >::type async (_Fn`  
`&&__fn, _Args &&...__args)`
- `long double atan (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`atan (_Tp __x)`
- `float atan (float __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type >`  
`atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Tp >, _Tp > atan (const valarray< _Tp`  
`> &__v)`
- `float atan2 (float __y, float __x)`
- `long double atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_`  
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`  
`>::__type, _Up >::__type atan2 (_Tp __y, _Up __x)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _`  
`Dom >, typename _Dom::value_type > atan2 (const typename _Dom::value_`  
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _`  
`Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value_`  
`type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _`  
`Dom, typename _Dom::value_type > &__e, const valarray< typename _`  
`Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type,`  
`_Dom >, typename _Dom::value_type > atan2 (const valarray< typename _`  
`Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_`  
`type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2`  
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > atan2`  
`(const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2`  
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `int atexit (void(*)()) throw ()`
- `bool atomic_compare_exchange_strong (atomic_address *__a, void **__v1,`  
`void *__v2)`



- bool **atomic\_compare\_exchange\_strong** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- template<typename \_ITp >  
bool **atomic\_compare\_exchange\_strong** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool **atomic\_compare\_exchange\_strong\_explicit** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **atomic\_compare\_exchange\_strong\_explicit** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- template<typename \_ITp >  
bool **atomic\_compare\_exchange\_strong\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- template<typename \_ITp >  
bool **atomic\_compare\_exchange\_weak** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool **atomic\_compare\_exchange\_weak** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- bool **atomic\_compare\_exchange\_weak** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- bool **atomic\_compare\_exchange\_weak\_explicit** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **atomic\_compare\_exchange\_weak\_explicit** (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- template<typename \_ITp >  
bool **atomic\_compare\_exchange\_weak\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2)
- bool **atomic\_exchange** (atomic\_bool \*\_\_a, bool \_\_i)
- void \* **atomic\_exchange** (atomic\_address \*\_\_a, void \*\_\_v)
- template<typename \_ITp >  
\_ITp **atomic\_exchange** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- bool **atomic\_exchange\_explicit** (atomic\_bool \*\_\_a, bool \_\_i, [memory\\_order](#) \_\_m)
- template<typename \_ITp >  
\_ITp **atomic\_exchange\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, [memory\\_order](#) \_\_m)
- void \* **atomic\_exchange\_explicit** (atomic\_address \*\_\_a, void \*\_\_v, [memory\\_order](#) \_\_m)
- template<typename \_ITp >  
\_ITp **atomic\_fetch\_add** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i)
- void \* **atomic\_fetch\_add** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- void \* **atomic\_fetch\_add\_explicit** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d, [memory\\_order](#) \_\_m)
- template<typename \_ITp >  
\_ITp **atomic\_fetch\_add\_explicit** (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \_\_i, [memory\\_order](#) \_\_m)

- `template<typename _ITp >`  
`_ITp atomic_fetch_and (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub (__atomic_base< _ITp > *__a, _ITp __i)`
- `void * atomic_fetch_sub (atomic_address *__a, ptrdiff_t __d)`
- `void * atomic_fetch_sub_explicit (atomic_address *__a, ptrdiff_t __d, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor (__atomic_base< _ITp > *__a, _ITp __i)`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, _ITp __i, memory\_order __m)`
- `void atomic_flag_clear (__atomic_flag_base *__a)`
- `void atomic_flag_clear_explicit (atomic_flag *__a, memory\_order __m)`
- `void atomic_flag_clear_explicit (__atomic_flag_base *, memory\_order) \_GLIBCXX\_NOTHROW`
- `bool atomic_flag_test_and_set (__atomic_flag_base *__a)`
- `bool atomic_flag_test_and_set_explicit (atomic_flag *__a, memory\_order __m)`
- `bool atomic_flag_test_and_set_explicit (__atomic_flag_base *, memory\_order) \_GLIBCXX\_NOTHROW`
- `bool atomic_is_lock_free (const atomic_bool *__a)`
- `bool atomic_is_lock_free (const atomic_address *__a)`
- `template<typename _ITp >`  
`bool atomic_is_lock_free (const __atomic_base< _ITp > *__a)`
- `bool atomic_load (const atomic_bool *__a)`
- `void * atomic_load (const atomic_address *__a)`
- `template<typename _ITp >`  
`_ITp atomic_load (const __atomic_base< _ITp > *__a)`
- `bool atomic_load_explicit (const atomic_bool *__a, memory\_order __m)`
- `template<typename _ITp >`  
`_ITp atomic_load_explicit (const __atomic_base< _ITp > *__a, memory\_order __m)`

- void \* **atomic\_load\_explicit** (const atomic\_address \* \_\_a, [memory\\_order](#) \_\_m)
- void **atomic\_store** (atomic\_bool \* \_\_a, bool \_\_i)
- void **atomic\_store** (atomic\_address \* \_\_a, void \* \_\_v)
- template<typename \_ITp >  
void **atomic\_store** (\_\_atomic\_base< \_ITp > \* \_\_a, \_ITp \_\_i)
- void **atomic\_store\_explicit** (atomic\_bool \* \_\_a, bool \_\_i, [memory\\_order](#) \_\_m)
- template<typename \_ITp >  
void **atomic\_store\_explicit** (\_\_atomic\_base< \_ITp > \* \_\_a, \_ITp \_\_i, [memory\\_order](#) \_\_m)
- void **atomic\_store\_explicit** (atomic\_address \* \_\_a, void \* \_\_v, [memory\\_order](#) \_\_m)
- template<typename \_Container >  
[back\\_insert\\_iterator](#)< \_Container > [back\\_inserter](#) (\_Container & \_\_x)
- template<class \_Container >  
auto [begin](#) (\_Container & \_\_cont)-> decltype(\_\_cont.begin())
- template<class \_Tp >  
\_Tp \* [begin](#) (valarray< \_Tp > & \_\_va)
- template<class \_Tp >  
const \_Tp \* [begin](#) (const valarray< \_Tp > & \_\_va)
- template<class \_Container >  
auto [begin](#) (const \_Container & \_\_cont)-> decltype(\_\_cont.begin())
- template<class \_Tp, size\_t \_Nm>  
\_Tp \* [begin](#) (\_Tp(& \_\_arr)[\_Nm])
- template<class \_Tp >  
const \_Tp \* [begin](#) (initializer\_list< \_Tp > \_\_ils)
- template<typename \_Filter, typename \_Tp >  
bool **binary\_search** (\_Filter, \_Filter, const \_Tp &)
- template<typename \_Filter, typename \_Tp, typename \_Compare >  
bool **binary\_search** (\_Filter, \_Filter, const \_Tp &, \_Compare)
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
bool [binary\\_search](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_ForwardIterator, typename \_Tp >  
bool [binary\\_search](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val)
- template<typename \_Functor, typename... \_ArgTypes>  
\_Bind\_helper< \_Functor, \_ArgTypes...>::type [bind](#) (\_Functor && \_\_f, \_ArgTypes &&... \_\_args)
- template<typename \_Result, typename \_Functor, typename... \_ArgTypes>  
\_Bindres\_helper< \_Result, \_Functor, \_ArgTypes...>::type [bind](#) (\_Functor && \_\_f, \_ArgTypes &&... \_\_args)
- template<typename \_Operation, typename \_Tp >  
[binder1st](#)< \_Operation > [binder1st](#) (const \_Operation & \_\_fn, const \_Tp & \_\_x)

- `template<typename _Operation, typename _Tp >`  
`binder2nd< _Operation > bind2nd` (const `_Operation` &\_\_fn, const `_Tp` &\_\_x)
- `ios_base` & `boolalpha` (`ios_base` &\_\_base)
- `template<typename _Callable, typename... _Args>`  
`void call_once` (`once_flag` &\_\_once, `_Callable` &&\_\_f, `_Args` &&...\_\_args)
- `float ceil` (`float` \_\_x)
- `long double ceil` (`long double` \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ceil`  
(`_Tp` \_\_x)
- `template<typename _Tp >`  
`complex< _Tp > conj` (const `complex< _Tp >` &)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type conj` (`_Tp` \_\_x)
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > const_pointer_cast` (const `shared_ptr< _Tp1 >` &\_\_r)
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > const_pointer_cast` (const `__shared_ptr< _Tp1, _Lp >` &\_\_r)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy` (`_Deque_iterator< _Tp, _Tp &, _Tp * >` \_\_first, `_Deque_iterator< _Tp, _Tp &, _Tp * >` \_\_last, `_Deque_iterator< _Tp, _Tp &, _Tp * >` \_\_result)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy` (`_Deque_iterator< _Tp, const _Tp &, const _Tp * >` \_\_first, `_Deque_iterator< _Tp, const _Tp &, const _Tp * >` \_\_last, `_Deque_iterator< _Tp, _Tp &, _Tp * >` \_\_result)
- `template<typename _Iter, typename _OIter >`  
`_OIter copy` (`_Iter`, `_Iter`, `_OIter`)
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type copy` (`istreambuf_iterator< _CharT >` \_\_first, `istreambuf_iterator< _CharT >` \_\_last, `ostreambuf_iterator< _CharT >` \_\_result)
- `template<typename _II, typename _OI >`  
`_OI copy` (`_II` \_\_first, `_II` \_\_last, `_OI` \_\_result)
- `template<typename _BI1, typename _BI2 >`  
`_BI2 copy_backward` (`_BI1` \_\_first, `_BI1` \_\_last, `_BI2` \_\_result)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward` (`_Deque_iterator< _Tp, _Tp &, _Tp * >` \_\_first, `_Deque_iterator< _Tp, _Tp &, _Tp * >` \_\_last, `_Deque_iterator< _Tp, _Tp &, _Tp * >` \_\_result)
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward` (`_Deque_iterator<`

- `_Tp, const _Tp &, const _Tp * > __first, \_Deque\_iterator< _Tp, const _Tp &, const _Tp * > __last, \_Deque\_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BIter1, typename _BIter2 >  
_BIter2 copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Ex >  
exception_ptr copy_exception (_Ex __ex) throw ()`
- `template<typename _IIter, typename _OIter, typename _Predicate >  
_OIter copy_if (_IIter, _IIter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >  
_OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _-  
OutputIterator __result, _Predicate __pred)`
- `template<typename _IIter, typename _Size, typename _OIter >  
_OIter copy_n (_IIter, _Size, _OIter)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >  
_OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __-  
result)`
- `float cos (float __x)`
- `long double cos (long double __x)`
- `template<typename _Tp >  
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type cos  
(_Tp __x)`
- `template<typename _Tp >  
complex< _Tp > cos (const complex< _Tp > &)`
- `template<class _Dom >  
_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > cos  
(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >  
_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > cos (const valarray< _Tp  
> &__v)`
- `float cosh (float __x)`
- `long double cosh (long double __x)`
- `template<typename _Tp >  
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type  
cosh (_Tp __x)`
- `template<typename _Tp >  
complex< _Tp > cosh (const complex< _Tp > &)`
- `template<class _Dom >  
_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type >  
cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >  
_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > cosh (const valarray< _Tp  
> &__v)`
- `template<typename _IIter, typename _Tp >  
iterator_traits< _IIter >::difference_type count (_IIter, _IIter, const _Tp &)`

- `template<typename _InputIterator, typename _Tp >`  
`iterator_traits< _InputIterator >::difference_type` `count` (`_InputIterator __first`,  
`_InputIterator __last`, `const _Tp &__value`)
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type` `count_if` (`_Iter, _Iter, _Predicate`)
- `template<typename _InputIterator, typename _Predicate >`  
`iterator_traits< _InputIterator >::difference_type` `count_if` (`_InputIterator __-`  
`first, _InputIterator __last, _Predicate __pred`)
- `exception_ptr` `current_exception` () `throw` ()
- `ios_base` & `dec` (`ios_base &__base`)
- `typedef decltype` (`nullptr`) `nullptr_t`
- `template<typename _Tp >`  
`add_rvalue_reference< _Tp >::type` `declval` () `noexcept`
- `template<typename _InputIterator >`  
`iterator_traits< _InputIterator >::difference_type` `distance` (`_InputIterator __-`  
`first, _InputIterator __last`)
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp >` `dynamic_pointer_cast` (`const shared_ptr< _Tp1 > &__r`)
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp >` `dynamic_pointer_cast` (`const __shared_ptr< _Tp1,`  
`_Lp > &__r`)
- `template<class _Tp >`  
`_Tp * end` (`valarray< _Tp > &__va`)
- `template<class _Tp >`  
`const _Tp * end` (`const valarray< _Tp > &__va`)
- `template<class _Tp, size_t _Nm>`  
`_Tp * end` (`_Tp(&__arr)[_Nm]`)
- `template<class _Tp >`  
`const _Tp * end` (`initializer_list< _Tp > __ils`)
- `template<class _Container >`  
`auto end` (`_Container &__cont`)-> `decltype(__cont.end())`
- `template<class _Container >`  
`auto end` (`const _Container &__cont`)-> `decltype(__cont.end())`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & endl` (`basic_ostream< _CharT, _Traits >`  
`&__os`)
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & ends` (`basic_ostream< _CharT, _Traits >`  
`&__os`)
- `template<typename _II1, typename _II2 >`  
`bool equal` (`_II1 __first1, _II1 __last1, _II2 __first2`)
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal` (`_Iter1, _Iter1, _Iter2`)

- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`pair< _Filter, _Filter > equal\_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`pair< _Filter, _Filter > equal\_range (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`pair< _ForwardIterator, _ForwardIterator > equal\_range (_ForwardIterator __-`  
`first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > equal\_range (_ForwardIterator __-`  
`first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `void exit (int) _GLIBCXX_NORETURN throw ()`
- `float exp (float __x)`
- `long double exp (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type exp`  
`(_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type > exp`  
`(const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`  
`complex< _Tp > exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > exp (const valarray< _Tp`  
`> & __v)`
- `float fabs (float __x)`
- `long double fabs (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`fabs (_Tp __x)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `void fill (_Bit_iterator __first, _Bit_iterator __last, const bool & __x)`
- `template<typename _Tp >`  
`void fill (const Deque\_iterator< _Tp, _Tp &, _Tp * > & __first, const Deque\_`  
`iterator< _Tp, _Tp &, _Tp * > & __last, const _Tp & __value)`
- `template<typename _Filter, typename _Tp >`  
`void fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`_OI fill\_n (_OI __first, _Size __n, const _Tp & __value)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`_OIter fill\_n (_OIter, _Size, const _Tp &)`

- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type` **find** (`istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val`)
- `template<typename _Iter, typename _Tp >`  
`_Iter` **find** (`_Iter, _Iter, const _Tp &`)
- `template<typename _InputIterator, typename _Tp >`  
`_InputIterator` **find** (`_InputIterator __first, _InputIterator __last, const _Tp &__val`)
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1` **find\_end** (`_Filter1, _Filter1, _Filter2, _Filter2`)
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1` **find\_end** (`_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate`)
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`_ForwardIterator1` **find\_end** (`_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2`)
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`_ForwardIterator1` **find\_end** (`_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp`)
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1` **find\_first\_of** (`_Filter1, _Filter1, _Filter2, _Filter2`)
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`_Filter1` **find\_first\_of** (`_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate`)
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`_InputIterator` **find\_first\_of** (`_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp`)
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_InputIterator` **find\_first\_of** (`_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2`)
- `template<typename _Iter, typename _Predicate >`  
`_Iter` **find\_if** (`_Iter, _Iter, _Predicate`)
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator` **find\_if** (`_InputIterator __first, _InputIterator __last, _Predicate __pred`)
- `template<typename _Iter, typename _Predicate >`  
`_Iter` **find\_if\_not** (`_Iter, _Iter, _Predicate`)
- `template<typename _InputIterator, typename _Predicate >`  
`_InputIterator` **find\_if\_not** (`_InputIterator __first, _InputIterator __last, _Predicate __pred`)
- `ios_base & fixed` (`ios_base &__base`)
- `float floor` (`float __x`)
- `long double floor` (`long double __x`)



- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`floor (_Tp __x)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits >`  
`& __os)`
- `float fmod (float __x, float __y)`
- `long double fmod (long double __x, long double __y)`
- `template<typename _Iter, typename _Funct >`  
`_Funct for_each (_Iter, _Iter, _Funct)`
- `template<typename _InputIterator, typename _Function >`  
`_Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Tp >`  
`enable_if< !is_lvalue_reference< _Tp >::value, _Tp && >::type forward (type-`  
`name std::common_type< _Tp >::type & __t)`
- `template<typename _Tp >`  
`enable_if< !is_lvalue_reference< _Tp >::value, _Tp && >::type forward (type-`  
`name std::common_type< _Tp >::type && __t)`
- `template<typename _Tp >`  
`enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type forward (type-`  
`name std::common_type< _Tp >::type __t)`
- `template<typename _Tp >`  
`enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type forward (type-`  
`name std::remove_reference< _Tp >::type && __t)`
- `template<typename... _Elements>`  
`tuple< _Elements &&...> forward_as_tuple (_Elements &&... __args)`
- `float frexp (float __x, int * __exp)`
- `long double frexp (long double __x, int * __exp)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type fr-`  
`exp (_Tp __x, int * __exp)`
- `template<typename _Container >`  
`front_insert_iterator< _Container > front_inserter (_Container & __x)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _ForwardIterator, typename _Generator >`  
`void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __-`  
`gen)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`  
`_RealType generate_canonical (_UniformRandomNumberGenerator & __g)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __first, _Size __n, _Generator __-`  
`gen)`

- `_GLIBCXX_CONST` const [error\\_category](#) & [generic\\_category](#) () throw ()
- `template<std::size_t __i, typename... _Elements>`  
`__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type`  
`>::type get (const tuple< _Elements...> &__t)`
- `template<std::size_t __i, typename... _Elements>`  
`__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type`  
`>::type get (tuple< _Elements...> &__t)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * get_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _MoneyT >`  
`_Get_money< _MoneyT > get_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _Tp >`  
`pair< _Tp *, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits >`  
`&__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits >`  
`&__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & getline (basic_istream< char > &__in, basic_string<`  
`char > &__str, char __delim)`
- `template<>`  
`basic_istream< wchar_t > & getline (basic_istream< wchar_t > &__in, basic_`  
`string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`  
`typename > class _Base>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits >`  
`&__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str,`  
`_CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`  
`typename > class _Base>`  
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits >`  
`&__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _Facet >`  
`bool has_facet (const locale &__loc) throw ()`
- `ios_base & hex (ios_base &__base)`
- `template<typename _Tp >`  
`_Tp imag (const complex< _Tp > &__z)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`

- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`  
`__first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`  
`__first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`  
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _-`  
`BinaryOperation1, typename _BinaryOperation2 >`  
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _-`  
`BinaryOperation2 __binary_op2)`
- `template<typename _BIter, typename _Compare >`  
`void inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _BIter >`  
`void inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BidirectionalIterator >`  
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __-`  
`middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Container, typename _Iterator >`  
`insert_iterator< _Container > inserter (_Container &__x, _Iterator __i)`
- `ios_base & internal (ios_base &__base)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RandomAccessIterator >`  
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`  
`Compare __comp)`
- `template<typename _RAIter >`  
`bool is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`bool is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _-`  
`RandomAccessIterator __last, _Compare __comp)`

- `template<typename _RAIter >`  
`_RAIter is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Filter >`  
`bool is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`bool is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator, typename _Compare >`  
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Filter >`  
`_Filter is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _CharT >`  
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool ispunct (_CharT __c, const locale &__loc)`

- `template<typename _CharT >`  
`bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _Filter1, typename _Filter2 >`  
`void iter_swap (_Filter1, _Filter2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _Tp >`  
`_Tp kill_dependency (_Tp __y)`
- `float ldexp (float __x, int __exp)`
- `long double ldexp (long double __x, int __exp)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ldexp (_Tp __x, int __exp)`
- `ios_base & left (ios_base &__base)`
- `template<typename _II1, typename _II2 >`  
`bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _Iiter1, typename _Iiter2 >`  
`bool lexicographical_compare (_Iiter1, _Iiter1, _Iiter2, _Iiter2)`
- `template<typename _Iiter1, typename _Iiter2, typename _Compare >`  
`bool lexicographical_compare (_Iiter1, _Iiter1, _Iiter2, _Iiter2, _Compare)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`void lock (_L1 &, _L2 &, _L3 &...)`
- `long double log (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > log (const complex< _Tp > &)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > log (const valarray< _Tp > &__v)`
- `float log (float __x)`

- `template<class _Dom >`  
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type >`  
`log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `float log10 (float __x)`
- `long double log10 (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`log10 (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > log10 (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > log10 (const valarray<`  
`_Tp > &__v)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator lower\_bound (_ForwardIterator __first, _ForwardIterator __-`  
`last, const _Tp &__val)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`_Filter lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`_Filter lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`_ForwardIterator lower\_bound (_ForwardIterator __first, _ForwardIterator __-`  
`last, const _Tp &__val, _Compare __comp)`
- `error\_code make_error_code (errc __e)`
- `error\_code make_error_code (future\_errc __errc)`
- `error\_condition make_error_condition (errc __e)`
- `error\_condition make_error_condition (future\_errc __errc)`
- `template<typename _Ex >`  
`exception_ptr make\_exception\_ptr (_Ex __ex) throw ()`
- `template<typename _RandomAccessIterator >`  
`void make\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void make\_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`  
`last, _Compare __comp)`
- `template<typename _RAIter >`  
`void make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Iterator >`  
`move\_iterator< _Iterator > make_move_iterator (const _Iterator &__i)`
- `template<class _T1, class _T2 >`  
`pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_-`  
`strip< _T2 >::__type > make\_pair (_T1 &&__x, _T2 &&__y)`

- `template<typename _Tp, typename... _Args>`  
`shared_ptr< _Tp > make_shared ( _Args &&... __args )`
- `template<typename... _Elements>`  
`tuple< typename __decay_and_strip< _Elements >::__type...> make_tuple`  
`( _Elements &&... __args )`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`  
`_Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter max_element ( _Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter max_element ( _Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator max_element ( _ForwardIterator __first, _ForwardIterator __-`  
`last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator max_element ( _ForwardIterator __first, _ForwardIterator __-`  
`last)`
- `template<typename _Tp, typename _Class >`  
`_Mem_fn< _Tp _Class::* > mem_fn ( _Tp _Class::* __pm)`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_t< _Ret, _Tp > mem_fun ( _Ret( _Tp::* __f )())`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun ( _Ret( _Tp::* __f )( _Arg ))`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_ref_t< _Ret, _Tp > mem_fun_ref ( _Ret( _Tp::* __f )())`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref ( _Ret( _Tp::* __f )( _Arg ))`
- `void * memchr (void * __s, int __c, size_t __n)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter merge ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter merge ( _Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`_OutputIterator merge ( _InputIterator1 __first1, _InputIterator1 __last1, _-`  
`InputIterator2 __first2, InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`  
`name _Compare >`

```

_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _-
InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-
Compare __comp)

```

- `template<typename _Tp, typename _Compare >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`  
`_Tp min (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`_Tp min (initializer\_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`_Filter min\_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min\_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator min\_element (_ForwardIterator __first, _ForwardIterator __-  
last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`_ForwardIterator min\_element (_ForwardIterator __first, _ForwardIterator __-  
last, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`  
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b,  
_Compare __comp)`
- `template<typename _Tp >`  
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`  
`pair< _Tp, _Tp > minmax (initializer\_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`pair< _Tp, _Tp > minmax (initializer\_list< _Tp >, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`pair< _Filter, _Filter > minmax\_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`pair< _Filter, _Filter > minmax\_element (_Filter, _Filter)`
- `template<typename _ForwardIterator >`  
`pair< _ForwardIterator, _ForwardIterator > minmax\_element (_ForwardIterator  
__first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`pair< _ForwardIterator, _ForwardIterator > minmax\_element (_ForwardIterator  
__first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _-  
InputIterator1 __last1, _InputIterator2 __first2)`



- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1,`  
`_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_`  
`pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `float modf (float __x, float *__iptr)`
- `long double modf (long double __x, long double *__iptr)`
- `template<typename _II, typename _OI >`  
`_OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, _Tp`  
`&, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_`  
`iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp *`  
`> __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`std::remove_reference< _Tp >::type && move (_Tp && __t)`
- `template<typename _BI1, typename _BI2 >`  
`_BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward (_Deque_iterator<`  
`_Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last,`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward (_Deque_iterator<`  
`_Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &,`  
`const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _ForwardIterator >`  
`_ForwardIterator next (_ForwardIterator __x, typename iterator_traits< _`  
`ForwardIterator >::difference_type __n=1)`
- `template<typename _BIter, typename _Compare >`  
`bool next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BIter >`  
`bool next_permutation (_BIter, _BIter)`
- `template<typename _BidirectionalIterator >`  
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __`  
`last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __`  
`last, _Compare __comp)`

- [ios\\_base](#) & [noboolalpha](#) ([ios\\_base](#) & \_\_base)
- `template<typename _Iter, typename _Predicate >`  
`bool none_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`_Tp norm (const complex< _Tp > &)`
- [ios\\_base](#) & [noshowbase](#) ([ios\\_base](#) & \_\_base)
- [ios\\_base](#) & [noshowpoint](#) ([ios\\_base](#) & \_\_base)
- [ios\\_base](#) & [noshowpos](#) ([ios\\_base](#) & \_\_base)
- [ios\\_base](#) & [noskipws](#) ([ios\\_base](#) & \_\_base)
- `template<typename _Predicate >`  
`unary\_negate< _Predicate > not1 (const _Predicate & __pred)`
- `template<typename _Predicate >`  
`binary\_negate< _Predicate > not2 (const _Predicate & __pred)`
- [ios\\_base](#) & [nounitbuf](#) ([ios\\_base](#) & \_\_base)
- [ios\\_base](#) & [nouppercase](#) ([ios\\_base](#) & \_\_base)
- `template<typename _RAIter >`  
`void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __-  
nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __-  
nth, _RandomAccessIterator __last, _Compare __comp)`
- [ios\\_base](#) & [oct](#) ([ios\\_base](#) & \_\_base)
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator!= (const sub\_match< _Bi_iter > & __lhs, const basic\_string<  
typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > & __-  
rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`  
`bool operator!= (const \_Deque\_iterator< _Tp, _Ref, _Ptr > & __x, const \_Deque\_iterator< _Tp, _Ref, _Ptr > & __y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`  
`bool operator!= (const \_Deque\_iterator< _Tp, _RefL, _PtrL > & __x, const \_Deque\_iterator< _Tp, _RefR, _PtrR > & __y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc  
> & __y)`
- `template<typename _Iterator >`  
`bool operator!= (const reverse\_iterator< _Iterator > & __x, const reverse\_-  
iterator< _Iterator > & __y)`

- `template<typename _Tp >`  
`bool operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_`  
`const_iterator< _Tp > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_`  
`iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`  
`_Tp, _Alloc > &__ly)`
- `template<typename _IteratorL, typename _IteratorR >`  
`bool operator!= (const move_iterator< _IteratorL > &__x, const move_`  
`iterator< _IteratorR > &__y)`
- `template<typename _Val >`  
`bool operator!= (const _List_iterator< _Val > &__x, const _List_const_`  
`iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`  
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const`  
`multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const mul-`  
`tiset< _Key, _Compare, _Alloc > &__y)`
- `template<class _T1, class _T2 >`  
`bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key,`  
`_Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`  
`&__y)`
- `template<typename _Val >`  
`bool operator!= (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_`  
`const_iterator< _Val > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`  
`name _Alloc >`

```

bool operator!= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _
Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc
> &__y)
• template<typename _Tp, typename _Alloc >
bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc
> &__y)
• template<class _Tp, class _CharT, class _Traits, class _Dist >
bool operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x,
const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)
• template<typename _Bi_iter >
bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_-
traits< _Bi_iter >::value_type const *__rhs)
• template<typename _Tp, typename _Dp, typename _Up, typename _Ep >
bool operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _-
Up, _Ep > &__y)
• template<typename _Tp, typename _Dp >
bool operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_-
code>
bool operator!= (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc,
__cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _-
Pred, _Alloc, __cache_hash_code > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
> &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &_-
__y)
• template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
bool operator!= (const __unordered_set< _Value, _Hash, _Pred, _Alloc, __-
cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred, _-
Alloc, __cache_hash_code > &__y)
• template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>
bool operator!= (const __unordered_multiset< _Value, _Hash, _Pred, _Alloc,
__cache_hash_code > &__x, const __unordered_multiset< _Value, _Hash, _-
Pred, _Alloc, __cache_hash_code > &__y)
• template<class _Value, class _Hash, class _Pred, class _Alloc >
bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,
const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
• template<class _Value, class _Hash, class _Pred, class _Alloc >
bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >
&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
• template<typename _StateT >
bool operator!= (const fpos< _StateT > &__lhs, const fpos< _StateT > &__-
rhs)

```

- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>`  
`bool operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>`  
`bool operator!= (const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>`  
`bool operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)`

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __not_equal_to, typename _Dom1::value_type >::result_type >`  
`operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _RealType >`  
`bool operator!= (const std::cauchy\_distribution< _RealType > &__d1, const`  
`std::cauchy\_distribution< _RealType > &__d2)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, type-`  
`name _Dom::value_type >::result_type > operator!= (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`  
`&__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename`  
`_Dom::value_type >::result_type > operator!= (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type`  
`> &__e)`
- `template<typename _RealType >`  
`bool operator!= (const std::fisher\_f\_distribution< _RealType > &__d1, const`  
`std::fisher\_f\_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::student\_t\_distribution< _RealType > &__d1, const`  
`std::student\_t\_distribution< _RealType > &__d2)`
- `bool operator!= (const std::bernoulli\_distribution &__d1, const std::bernoulli\_-`  
`distribution &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::binomial\_distribution< _IntType > &__d1, const`  
`std::binomial\_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool operator!= (const std::poisson\_distribution< _IntType > &__d1, const`  
`std::poisson\_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::exponential\_distribution< _RealType > &__d1,`  
`const std::exponential\_distribution< _RealType > &__d2)`
- `bool operator!= (const error\_code &__lhs, const error\_code &__rhs)`
- `template<typename _IntType >`  
`bool operator!= (const std::discrete\_distribution< _IntType > &__d1, const`  
`std::discrete\_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::piecewise\_constant\_distribution< _RealType > &__-`  
`__d1, const std::piecewise\_constant\_distribution< _RealType > &__d2)`
- `bool operator!= (const error\_code &__lhs, const error\_condition &__rhs)`

- `template<typename _IntType >`  
`bool operator!= (const std::negative\_binomial\_distribution< _IntType > &__d1,`  
`const std::negative\_binomial\_distribution< _IntType > &__d2)`
- `bool operator!= (const error\_condition &__lhs, const error\_condition &__rhs)`
- `template<typename _IntType >`  
`bool operator!= (const std::geometric\_distribution< _IntType > &__d1, const`  
`std::geometric\_distribution< _IntType > &__d2)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename`  
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`  
`_Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename`  
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type >`  
`&__v)`
- `template<typename _CharT, typename _Traits >`  
`bool operator!= (const istreambuf\_iterator< _CharT, _Traits > &__a, const`  
`istreambuf\_iterator< _CharT, _Traits > &__b)`
- `template<typename _Tp, typename _Dp >`  
`bool operator!= (const unique\_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename`  
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`  
`_Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename`  
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `bool operator!= (thread::id __x, thread::id __y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_`  
`code>`  
`bool operator!= (const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _-`  
`_Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp,`  
`_Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _RealType >`  
`bool operator!= (const std::extreme\_value\_distribution< _RealType > &__d1,`  
`const std::extreme\_value\_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::weibull\_distribution< _RealType > &__d1, const`  
`std::weibull\_distribution< _RealType > &__d2)`
- `template<typename _Bi_iter >`  
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const *__lhs,`  
`const sub\_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Allocator >`  
`bool operator!= (const match\_results< _Bi_iter, _Allocator > &__m1, const`  
`match\_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool operator!= (const shared\_ptr< _Tp1 > &__a, const shared\_ptr< _Tp2 >`  
`&__b)`

- `template<typename _Tp >`  
`bool operator!= (const shared\_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Tp >`  
`bool operator!= (nullptr_t, const shared\_ptr< _Tp > &__b)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const valar-`  
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`  
`bool operator!= (const basic\_string< typename iterator_traits< _Bi_iter`  
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub\_match< _Bi_iter >`  
`&__rhs)`
- `template<typename _Bi_iter >`  
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const &__-`  
`lhs, const sub\_match< _Bi_iter > &__rhs)`
- `bool operator!= (const error\_condition &__lhs, const error\_code &__rhs)`
- `template<typename... _TElements, typename... _UElements>`  
`bool operator!= (const tuple< _TElements...> &__t, const tuple< _-`  
`UElements...> &__u)`
- `template<typename _RealType >`  
`bool operator!= (const std::piecewise\_linear\_distribution< _RealType > &__-`  
`d1, const std::piecewise\_linear\_distribution< _RealType > &__d2)`
- `template<typename _Bilter >`  
`bool operator!= (const sub\_match< _Bilter > &__lhs, const sub\_match< _-`  
`Bilter > &__rhs)`
- `template<typename _Bi_iter >`  
`bool operator!= (const sub\_match< _Bi_iter > &__lhs, typename iterator_-`  
`traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<typename _T1, typename _T2 >`  
`bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >`  
`bool operator!= (const allocator< _Tp > &, const allocator< _Tp > &)`



- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const valarray`  
`< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __not_equal_to, _Tp >::result_type > operator!= (const _Tp &__`  
`_t, const valarray< _Tp > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const`  
`basic\_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const`  
`_CharT *__rhs)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename _`  
`_fun< __modulus, typename _Dom1::value_type >::result_type > operator%`  
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_`  
`type >::result_type > operator% (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_`  
`type >::result_type > operator% (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`  
`>::result_type > operator% (const valarray< typename _Dom::value_type >`  
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`  
`>::result_type > operator% (const typename _Dom::value_type &__t, const`  
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename`

- ```
__fun< __modulus, _Tp >::result_type > operator% (const valarray< _Tp >
&__v, const valarray< _Tp > &__w)
```
- `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > operator% (const valarray< _Tp >`
`&__v, const _Tp &__t)`
 - `template<typename _Tp >`
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __modulus, _Tp >::result_type > operator% (const _Tp &__t, const`
`valarray< _Tp > &__v)`
 - `_Ios_Fmtflags operator& (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
 - `_Ios_Openmode operator& (_Ios_Openmode __a, _Ios_Openmode __b)`
 - `_Ios_Iostate operator& (_Ios_Iostate __a, _Ios_Iostate __b)`
 - `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename`
`__fun< __bitwise_and, typename _Dom1::value_type >::result_type > opera-`
`tor& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _`
`Expr< _Dom2, typename _Dom2::value_type > &__w)`
 - `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename`
`_Dom::value_type >, typename __fun< __bitwise_and, typename _`
`Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
 - `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename`
`_Dom::value_type >, typename __fun< __bitwise_and, typename _`
`Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type >`
`&__v)`
 - `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value -`
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`
`>::result_type > operator& (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
 - `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value -`
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`
`>::result_type > operator& (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
 - `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_and, _Tp >::result_type > operator& (const _Tp &__`
`__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`
`type >::result_type > operator&& (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`
`__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename`
`__fun< __logical_and, typename _Dom1::value_type >::result_type > opera-`
`tor&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_`
`type >::result_type > operator&& (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`
`>::result_type > operator&& (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type`
`>::result_type > operator&& (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > operator&& (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __logical_and, _Tp >::result_type > operator&& (const _Tp`
`&__t, const valarray< _Tp > &__v)`

- `_Ios_Openmode & operator&= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate & operator&= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Fmtflags & operator&= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `template<class _Dom1, class _Dom2 >
_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, type-
name __fun< __multiplies, typename _Dom1::value_type >::result_type >
operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const
_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >
_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _-
Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_
type >::result_type > operator* (const _Expr< _Dom, typename _-
Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >
_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_
type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type
>::result_type > operator* (const typename _Dom::value_type &__t, const _-
Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >
_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _-
Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_
type >::result_type > operator* (const _Expr< _Dom, typename _-
Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_-
_v)`
- `template<class _Dom >
_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_
type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type
>::result_type > operator* (const valarray< typename _Dom::value_type >
&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >
_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename
__fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp >
&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >
_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename
__fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp >
&__v, const _Tp &__t)`
- `template<typename _Tp >
_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename
__fun< __multiplies, _Tp >::result_type > operator* (const _Tp &__t, const
valarray< _Tp > &__v)`
- `_Bit_const_iterator operator+ (ptrdiff_t __n, const _Bit_const_iterator &__x)`
- `_Bit_iterator operator+ (ptrdiff_t __n, const _Bit_iterator &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >
Deque_iterator< _Tp, _Ref, _Ptr > operator+ (ptrdiff_t __n, const Deque_-
iterator< _Tp, _Ref, _Ptr > &__x)`

- `template<typename _Iterator >`
`reverse_iterator< _Iterator > operator+ (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __plus, typename _Dom1::value_type >::result_type > operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > operator+ (const _Tp &__t, const valarray< _Tp > &__v)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _`
`CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc`
`> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_`
`string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _`
`CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _`
`CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type operator- (const _`
`Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref,`
`_Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type operator- (const _`
`Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _`
`RefR, _PtrR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `ptrdiff_t operator- (const _Bit_iterator_base &__x, const _Bit_iterator_base`
`&__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_`
`type > operator- (const typename _Dom::value_type &__t, const _Expr< _`
`Dom, typename _Dom::value_type > &__v)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto operator- (const move_iterator< _IteratorL > &__x, const move_iterator<`
`_IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >::difference_type operator- (const reverse_`
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > &__x)`

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __-`
`fun< __minus, typename _Dom1::value_type >::result_type > operator- (const`
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __minus, typename _Dom::value_type`
`>::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type`
`> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_`
`type > operator- (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __minus, typename _Dom::value_type`
`>::result_type > operator- (const _Expr< _Dom, typename _Dom::value_type`
`> &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename __-`
`fun< __minus, _Tp >::result_type > operator- (const valarray< _Tp > &__v,`
`const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename __-`
`fun< __minus, _Tp >::result_type > operator- (const valarray< _Tp > &__v,`
`const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _-`
`_fun< __minus, _Tp >::result_type > operator- (const _Tp &__t, const valar-`
`ray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename`
`__fun< __divides, typename _Dom1::value_type >::result_type > operator/`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __divides, typename _Dom::value_type`
`>::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type`
`> &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type,`

```

_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_
type > operator/ (const typename _Dom::value_type &__t, const _Expr< _
Dom, typename _Dom::value_type > &__v)
• template<class _Dom >
_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type,
_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_
type > operator/ (const valarray< typename _Dom::value_type > &__v, const
_Expr< _Dom, typename _Dom::value_type > &__e)
• template<class _Dom >
_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _
Dom::value_type >, typename __fun< __divides, typename _Dom::value_type
>::result_type > operator/ (const _Expr< _Dom, typename _Dom::value_type
> &__e, const valarray< typename _Dom::value_type > &__v)
• template<typename _Tp >
_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _
fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &_
v, const valarray< _Tp > &__w)
• template<typename _Tp >
_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _
fun< __divides, _Tp >::result_type > operator/ (const valarray< _Tp > &_
v, const _Tp &__t)
• template<typename _Tp >
_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _
fun< __divides, _Tp >::result_type > operator/ (const _Tp &__t, const valar-
ray< _Tp > &__v)
• template<typename _CharT, typename _Traits, typename _Alloc >
bool operator< (const basic\_string< _CharT, _Traits, _Alloc > &__lhs, const
_CharT *__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc >
bool operator< (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _
Alloc > &__rhs)
• template<typename _Tp >
_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _
fun< __less, _Tp >::result_type > operator< (const _Tp &__t, const valarray<
_Tp > &__v)
• template<typename _Tp, typename _Ref, typename _Ptr >
bool operator< (const Deque\_iterator< _Tp, _Ref, _Ptr > &__x, const _
Deque_iterator< _Tp, _Ref, _Ptr > &__y)
• template<typename _Tp, typename _Alloc >
bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc
> &__y)
• template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-
name _Alloc >
bool operator< (const Rb\_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc

```


- > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Iterator >
bool **operator**< (const [reverse_iterator](#)< _Iterator > &__x, const [reverse_iterator](#)< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR >
bool **operator**< (const [reverse_iterator](#)< _IteratorL > &__x, const [reverse_iterator](#)< _IteratorR > &__y)
- template<typename _Tp, typename _Alloc >
bool **operator**< (const [forward_list](#)< _Tp, _Alloc > &__lx, const [forward_list](#)< _Tp, _Alloc > &__ly)
- template<typename _IteratorL, typename _IteratorR >
bool **operator**< (const [move_iterator](#)< _IteratorL > &__x, const [move_iterator](#)< _IteratorR > &__y)
- template<typename _Bi_iter >
bool **operator**< (const [sub_match](#)< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Tp, typename _Alloc >
bool **operator**< (const [list](#)< _Tp, _Alloc > &__x, const [list](#)< _Tp, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >
bool **operator**< (const [map](#)< _Key, _Tp, _Compare, _Alloc > &__x, const [map](#)< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >
bool **operator**< (const [multimap](#)< _Key, _Tp, _Compare, _Alloc > &__x, const [multimap](#)< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc >
bool **operator**< (const [multiset](#)< _Key, _Compare, _Alloc > &__x, const [multiset](#)< _Key, _Compare, _Alloc > &__y)
- template<class _T1, class _T2 >
bool **operator**< (const [pair](#)< _T1, _T2 > &__x, const [pair](#)< _T1, _T2 > &__y)
- template<typename _Tp, typename _Seq >
bool **operator**< (const [queue](#)< _Tp, _Seq > &__x, const [queue](#)< _Tp, _Seq > &__y)
- template<typename _Key, typename _Compare, typename _Alloc >
bool **operator**< (const [set](#)< _Key, _Compare, _Alloc > &__x, const [set](#)< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq >
bool **operator**< (const [stack](#)< _Tp, _Seq > &__x, const [stack](#)< _Tp, _Seq > &__y)
- template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >
bool **operator**< (const [Deque_iterator](#)< _Tp, _RefL, _PtrL > &__x, const [Deque_iterator](#)< _Tp, _RefR, _PtrR > &__y)
- bool **operator**< (const [error_code](#) &__lhs, const [error_code](#) &__rhs)

- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_`
`type >, typename __fun< __less, typename _Dom::value_type >::result_type >`
`operator< (const _Expr< _Dom, typename _Dom::value_type > &__v, const`
`typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _`
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> operator< (const typename _Dom::value_type &__t, const _Expr< _Dom,`
`typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_`
`type >, typename __fun< __less, typename _Dom::value_type >::result_type >`
`operator< (const _Expr< _Dom, typename _Dom::value_type > &__e, const`
`valarray< typename _Dom::value_type > &__v)`
- `bool operator< (const error_condition &__lhs, const error_condition &__rhs)`
- `template<class _Dom1 , class _Dom2 >`
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`fun< __less, typename _Dom1::value_type >::result_type > operator< (const`
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`
`typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`
`> operator< (const valarray< typename _Dom::value_type > &__v, const _`
`Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Bi_iter >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp , typename _Dp , typename _Up , typename _Ep >`
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up,`
`_Ep > &__y)`
- `template<typename _Bilter >`
`bool operator< (const sub_match< _Bilter > &__lhs, const sub_match< _`
`Bilter > &__rhs)`
- `template<typename _Tp , typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc`
`> &__y)`
- `template<typename _Bi_iter >`
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs,`
`const sub_match< _Bi_iter > &__rhs)`

- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename __-`
`fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v,`
`const _Tp &__t)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 >`
`&__b)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator< (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr<`
`_Tp2, _Lp > &__b)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__-`
`_rhs)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator< (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator< (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename __-`
`fun< __less, _Tp >::result_type > operator< (const valarray< _Tp > &__v,`
`const valarray< _Tp > &__w)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`
`_CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType >`
`&__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _-`
`_Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, Setiosflags __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)`

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setw __f)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename`
`_Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_-`
`ostream< _CharT, _Traits > &__os, const discard_block_engine< _-`
`RandomNumberEngine, __p, __r > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`
`_CharT, _Traits > &__os, const negative_binomial_distribution< _IntType >`
`&__x)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __shift_left, typename _Dom1::value_type >::result_type >`
`operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT`
`, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_-`
`ostream< _CharT, _Traits > &__os, const std::independent_bits_engine<`
`_RandomNumberEngine, __w, _UIntType > &__x)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`
`type >::result_type > operator<< (const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, const error_code &__e)`

- `template<typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _-`
`Traits > & __os, const complex< _Tp > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > &, const std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > & __os, const fisher_f_distribution< _RealType > & __x)`
- `template<typename _CharT, typename _Traits, typename _Tp >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _-`
`Traits > & __os, const _Tp & __x)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_-`
`type >::result_type > operator<< (const _Expr< _Dom, typename _-`
`Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-`
`CharT, _Traits > & __os, const gamma_distribution< _RealType > & __x)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`
`>::result_type > operator<< (const valarray< typename _Dom::value_type >`
`& __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`
`>::result_type > operator<< (const typename _Dom::value_type & __t, const`
`_Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-`
`UIntType __f, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`
`_CharT, _Traits > & __os, const mersenne_twister_engine< _UIntType, __w,`
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT`
`, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream<`
`_CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a,`
`__c, __m > &__lcr)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,`
`typename > class _Base>`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _`
`Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _`
`Base > &__str)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename`
`_Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w,`
`__s, __r > &__x)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_`
`type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr`
`> &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__os, _Setprecision __f)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)`

- `template<class _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT,`
`_Traits > &__out, thread::id __id)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType >`
`&__x)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > operator<< (const valarray< _Tp`
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > operator<< (const valarray< _Tp`
`> &__v, const _Tp &__t)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_left, _Tp >::result_type > operator<< (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine,`
`__k > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`
`_Alloc > &__rhs)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __less_equal, _Tp >::result_type > operator<= (const valarray< _Tp`
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _`
`Alloc > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`

```

__fun< __less_equal, _Tp >::result_type > operator<= (const _Tp &__t, const
valarray< _Tp > &__v)

```

- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const *__-`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const -`
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const`
`_Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &-`
`__rhs)`
- `template<typename _Iterator >`
`bool operator<= (const reverse_iterator< _Iterator > &__x, const reverse_-`
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_-`
`iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_-`
`list< _Tp, _Alloc > &__ly)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator<= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`
`&__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const mul-`
`tiset< _Key, _Compare, _Alloc > &__y)`

- `template<class _T1, class _T2 >`
`bool operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less_equal, typename _Dom1::value_type >::result_type >`
`operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_type >::result_type >`
`operator<= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less_equal, _Tp >::result_type >`
`operator<= (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type >::result_type >`
`operator<= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_`
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`
`>::result_type > operator<= (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_`
`traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator<= (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr<`
`_Tp, _Lp > &__b)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`
`type >::result_type > operator<= (const _Expr< _Dom, typename _`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _BiIter >`
`bool operator<= (const sub_match< _BiIter > &__lhs, const sub_match< _`
`BiIter > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator<= (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `bool operator<= (thread::id __x, thread::id __y)`
- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const &__`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__`
`rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _`
`Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`
- `bool operator== (const std::bernoulli_distribution &__d1, const std::bernoulli_`
`distribution &__d2)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const mul`
`tiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`

- `template<typename _Tp >`
`bool operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__b)`
- `template<typename _Val >`
`bool operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Tp, typename _Seq >`
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _RealType >`
`bool operator== (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator== (const __unordered_multiset< _Value, _Hash, _Pred, _Alloc,`
`__cache_hash_code > &__x, const __unordered_multiset< _Value, _Hash, _`
`Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator== (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const`
`Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const move_iterator< _IteratorL > &__x, const move_-`
`iterator< _IteratorR > &__y)`
- `template<typename _RealType >`
`bool operator== (const std::weibull_distribution< _RealType > &__d1, const`
`std::weibull_distribution< _RealType > &__d2)`
- `template<typename _StateT >`
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__-`
`rhs)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type op-`
`erator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT`
`> &__rhs)`
- `template<typename _Val >`
`bool operator== (const Rb_tree_iterator< _Val > &__x, const Rb_tree_-`
`const_iterator< _Val > &__y)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`_fun< __equal_to, typename _Dom1::value_type >::result_type > operator==`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_-`
`type >::result_type > operator== (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_-`
`_v)`
- `bool operator== (const error_condition &__lhs, const error_code &__rhs)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x,`
`const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`

- `template<class _T1, class _T2 >`
`bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _RealType >`
`bool operator== (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _RealType >`
`bool operator== (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _CharT, typename _Traits >`
`bool operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `bool operator== (const error_condition &__lhs, const error_condition &__rhs)`
- `template<typename _IntType >`
`bool operator== (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _IntType >`
`bool operator== (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool operator== (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _IntType >`
`bool operator== (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`

- `template<typename _Bi_iter >`
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `bool operator== (const error_code &__lhs, const error_condition &__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`bool operator== (const match_results< _Bi_iter, _Allocator > &__m1, const`
`match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_-`
`list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 >`
`&__b)`
- `template<typename _Tp >`
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter >`
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const &__-`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_-`
`ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _RealType >`
`bool operator== (const std::extreme_value_distribution< _RealType > &__d1,`
`const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator== (const tuple< _TElements...> &__t, const tuple< _-`
`UElements...> &__u)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator== (const __unordered_set< _Value, _Hash, _Pred, _Alloc, __-`
`cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred, _-`
`Alloc, __cache_hash_code > &__y)`
- `template<typename _RealType >`
`bool operator== (const std::normal_distribution< _RealType > &__d1, const`
`std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator== (const std::piecewise_linear_distribution< _RealType > &__-`
`d1, const std::piecewise_linear_distribution< _RealType > &__d2)`

- `template<typename _Bilter >`
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Iterator >`
`bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator== (const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Res, typename... _Args>`
`bool operator== (const function< _Res(_Args...)> &__f, nullptr_t)`
- `template<typename _Res, typename... _Args>`
`bool operator== (nullptr_t, const function< _Res(_Args...)> &__f)`
- `template<typename _T1, typename _T2 >`
`bool operator== (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >`
`bool operator== (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`bool operator== (const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > operator== (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _-`
`Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _Tp, typename _Dp >`
`bool operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `bool operator== (const error_code &__lhs, const error_code &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multi-`
`set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc`
`> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _-`
`Alloc > &__rhs)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _-`
`Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename _-`
`_fun< __greater, typename _Dom1::value_type >::result_type > operator>`
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename`
`__fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp >`
`&__v, const _Tp &__t)`
- `bool operator> (thread::id __x, thread::id __y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &_-`
`__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const move_iterator< _IteratorL > &__x, const move_-`
`iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const *__rhs)`

- `template<class _T1, class _T2 >`
`bool operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_`
`type > operator> (const typename _Dom::value_type &__t, const _Expr< _`
`Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type,`
`_Dom >, typename __fun< __greater, typename _Dom::value_type >::result_`
`type > operator> (const valarray< typename _Dom::value_type > &__v, const`
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename _Tp, typename _Seq >`
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`
`&__y)`
- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__`
`__y)`
- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs,`
`const sub_match< _Bi_iter > &__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __greater, typename _Dom::value_type`
`>::result_type > operator> (const _Expr< _Dom, typename _Dom::value_`
`type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Iterator >`
`bool operator> (const reverse_iterator< _Iterator > &__x, const reverse_`
`iterator< _Iterator > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _`
`fun< __greater, _Tp >::result_type > operator> (const _Tp &__t, const valar-`
`ray< _Tp > &__v)`

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _BiIter >`
`bool operator> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator> (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator> (const tuple< _TElements...> &__t, const tuple< _TElements...> &__u)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator> (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator> (const Deque_iterator< _Tp, _Ref, _Ptr > &__x, const Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`

- `template<typename _Tp >`
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp >`
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list<`
`_Tp, _Alloc > &__ly)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT *__rhs)`
- `template<typename... _TElements, typename... _UElements>`
`bool operator>= (const tuple< _TElements...> &__t, const tuple< _`
`UElements...> &__u)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > operator>= (const valar-`
`ray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _`
`Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_`
`iterator< _IteratorR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc`
`> &__y)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const &__`
`lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, const basic_string<`
`typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits,`
`_Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc >`
`&__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __greater_equal, _Tp >::result_type > operator>= (const valar-`
`ray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const *__`
`lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<typename _Tp, typename _Seq >`
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq >`
`&__y)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator>= (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr<`
`_Tp, _Lp > &__b)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`
`name _Alloc >`
`bool operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _-`
`Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`
`> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _-`
`Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _-`
`Alloc > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr<`
`_Up, _Ep > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_-`
`list< _Tp, _Alloc > &__ly)`
- `template<class _T1, class _T2 >`
`bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq >`
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const move_iterator< _IteratorL > &__x, const move_-`
`iterator< _IteratorR > &__y)`
- `template<typename _BiIter >`
`bool operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _-`
`BiIter > &__rhs)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __greater_equal, typename _Dom1::value_type >::result_type >`
`operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _-`
`Dom::value_type, _Dom >, typename __fun< __greater_equal, type-`
`name _Dom::value_type >::result_type > operator>= (const typename _-`
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__v)`

- `template<typename _Iterator >`
`bool operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `bool operator>= (thread::id __x, thread::id __y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>= (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Bi_iter >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator>= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _`

- Dom::value_type >::result_type > **operator**>= (const [valarray](#)< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Bi_iter >
bool **operator**>= (const [sub_match](#)< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
 - template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
 - template<typename _CharT, typename _Traits, typename _Alloc >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [basic_string](#)< _CharT, _Traits, _Alloc > &__str)
 - template<typename _CharT, typename _Traits >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, _Setiosflags __f)
 - template<typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [std::bernoulli_distribution](#) &__x)
 - template<typename _CharT, typename _Traits, typename _MoneyT >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)
 - template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [__gnu_cxx::__versa_string](#)< _CharT, _Traits, _Alloc, _Base > &__str)
 - template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator**>> ([std::basic_istream](#)< _CharT, _Traits > &__is, [shuffle_order_engine](#)< _RandomNumberEngine, __k > &__x)
 - template<typename _Tp, typename _CharT, class _Traits >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, [complex](#)< _Tp > &__x)
 - template<typename _Tp >
_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > **operator**>> (const [valarray](#)< _Tp > &__v, const _Tp &__t)
 - template<>
[basic_istream](#)< char > & **operator**>> ([basic_istream](#)< char > &__is, [basic_string](#)< char > &__str)
 - template<typename _CharT, typename _Traits >
[basic_istream](#)< _CharT, _Traits > & **operator**>> ([basic_istream](#)< _CharT, _Traits > &__is, _Setprecision __f)

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, _Resetiosflags __f)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, negative_binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, _Setfill< _CharT > __f)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`
`>::result_type > operator>> (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &__is, _Setw __f)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type`
`>::result_type > operator>> (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename`
`_Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p,`
`__r > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Tp >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _-`
`Traits > &&__is, _Tp &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > & __is, chi_squared_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _`
`Traits > & __is, _Setbase __f)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _`
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_`
`type >::result_type > operator>> (const _Expr< _Dom, typename _`
`Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _`
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_`
`type >::result_type > operator>> (const _Expr< _Dom, typename _`
`Dom::value_type > & __e, const valarray< typename _Dom::value_type > & _`
`__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __shift_right, typename _Dom1::value_type >::result_type >`
`operator>> (const _Expr< _Dom1, typename _Dom1::value_type > & __v,`
`const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > & __is, binomial_distribution< _IntType > & __x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT`
`, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _`
`CharT, _Traits > & __is, linear_congruential_engine< _UIntType, __a, __c, _`
`m > & __lcr)`

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, lognormal_distribution< _RealType > &__x)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`
`__fun< __shift_right, _Tp >::result_type > operator>> (const valarray< _Tp`
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __shift_right, _Tp >::result_type > operator>> (const _Tp &__`
`_t, const valarray< _Tp > &__v)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename`
`_Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r`
`> &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-`
`CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const valarray<`
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > operator^ (const _Tp &__t,`
`const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_-`
`type >::result_type > operator^ (const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `_Ios_Iostate operator^` (`_Ios_Iostate __a, _Ios_Iostate __b`)
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`
`name __fun< __bitwise_xor, _Tp >::result_type > operator^` (`const valarray<`
`_Tp > &__v, const _Tp &__t`)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type`
`>::result_type > operator^` (`const typename _Dom::value_type &__t, const _-`
`Expr< _Dom, typename _Dom::value_type > &__v`)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type`
`>::result_type > operator^` (`const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e`)
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __bitwise_xor, typename _Dom1::value_type >::result_type >`
`operator^` (`const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w`)
- `_Ios_Fmtflags operator^` (`_Ios_Fmtflags __a, _Ios_Fmtflags __b`)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _-`
`Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_-`
`type >::result_type > operator^` (`const _Expr< _Dom, typename _-`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__-`
`__v`)
- `_Ios_Openmode operator^` (`_Ios_Openmode __a, _Ios_Openmode __b`)
- `_Ios_Iostate & operator^=` (`_Ios_Iostate &__a, _Ios_Iostate __b`)
- `_Ios_Openmode & operator^=` (`_Ios_Openmode &__a, _Ios_Openmode __b`)
- `_Ios_Fmtflags & operator^=` (`_Ios_Fmtflags &__a, _Ios_Fmtflags __b`)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _-`
`Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_-`
`type >::result_type > operator|` (`const _Expr< _Dom, typename _-`
`Dom::value_type > &__v, const typename _Dom::value_type &__t`)
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, type-`
`name __fun< __bitwise_or, typename _Dom1::value_type >::result_type >`
`operator|` (`const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w`)
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_-`
`type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type`

- >::result_type > **operator**| (const [valarray](#)< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- _Ios_Iostate **operator**| (_Ios_Iostate __a, _Ios_Iostate __b)
- _Ios_Fmtflags **operator**| (_Ios_Fmtflags __a, _Ios_Fmtflags __b)
- template<class _Dom >
 _Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const _Expr< _Dom, typename _Dom::value_type > &__e, const [valarray](#)< typename _Dom::value_type > &__v)
- template<class _Dom >
 _Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp >
 _Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > **operator**| (const [valarray](#)< _Tp > &__v, const [valarray](#)< _Tp > &__w)
- template<typename _Tp >
 _Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > **operator**| (const [valarray](#)< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
 _Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > **operator**| (const _Tp &__t, const [valarray](#)< _Tp > &__v)
- _Ios_Openmode **operator**| (_Ios_Openmode __a, _Ios_Openmode __b)
- _Ios_Openmode & **operator**|= (_Ios_Openmode &__a, _Ios_Openmode __b)
- _Ios_Fmtflags & **operator**|= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)
- _Ios_Iostate & **operator**|= (_Ios_Iostate &__a, _Ios_Iostate __b)
- template<class _Dom >
 _Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > **operator**|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const [valarray](#)< typename _Dom::value_type > &__v)
- template<class _Dom >
 _Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > **operator**|| (const [valarray](#)< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
 _Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename

```

__fun< __logical_or, _Tp >::result_type > operator|| (const valarray< _Tp >
&__v, const _Tp &__t)
• template<typename _Tp >
  _Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename
  __fun< __logical_or, _Tp >::result_type > operator|| (const _Tp &__t, const
valarray< _Tp > &__v)
• template<class _Dom >
  _Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _-
  Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_-
  type >::result_type > operator|| (const _Expr< _Dom, typename _-
  Dom::value_type > &__v, const typename _Dom::value_type &__t)
• template<class _Dom >
  _Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_-
  type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type
  >::result_type > operator|| (const typename _Dom::value_type &__t, const _-
  Expr< _Dom, typename _Dom::value_type > &__v)
• template<class _Dom1, class _Dom2 >
  _Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, type-
  name __fun< __logical_or, typename _Dom1::value_type >::result_type >
operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const
  _Expr< _Dom2, typename _Dom2::value_type > &__w)
• template<typename _Tp >
  _Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename
  __fun< __logical_or, _Tp >::result_type > operator|| (const valarray< _Tp >
  &__v, const valarray< _Tp > &__w)
• _Ios_Fmtflags operator~ (_Ios_Fmtflags __a)
• _Ios_Iostate operator~ (_Ios_Iostate __a)
• _Ios_Openmode operator~ (_Ios_Openmode __a)
• template<typename _RandomAccessIterator >
  void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __-
  middle, _RandomAccessIterator __last)
• template<typename _RAIter, typename _Compare >
  void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)
• template<typename _RandomAccessIterator, typename _Compare >
  void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __-
  middle, _RandomAccessIterator __last, _Compare __comp)
• template<typename _RAIter >
  void partial_sort (_RAIter, _RAIter, _RAIter)
• template<typename _Iter, typename _RAIter >
  _RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)
• template<typename _Iter, typename _RAIter, typename _Compare >
  _RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)
• template<typename _InputIterator, typename _RandomAccessIterator >
  _RandomAccessIterator partial\_sort\_copy (_InputIterator __first, _InputIterator

```

-
- `__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`
`result_last)`
 - `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator`
`__last, _RandomAccessIterator __result_first, _RandomAccessIterator __-`
`result_last, _Compare __comp)`
 - `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _BinaryOperation __binary_op)`
 - `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result)`
 - `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _-`
`Predicate __pred)`
 - `template<typename _BIter, typename _Predicate >`
`_BIter partition (_BIter, _BIter, _Predicate)`
 - `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`
`pair< _OIter1, _OIter2 > partition_copy (_Iter, _Iter, _OIter1, _OIter2, _-`
`Predicate)`
 - `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, type-`
`name _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __-`
`first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __-`
`out_false, _Predicate __pred)`
 - `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`
 - `template<typename _FIter, typename _Predicate >`
`_FIter partition_point (_FIter, _FIter, _Predicate)`
 - `template<typename _Tp >`
`complex< _Tp > polar (const _Tp &, const _Tp &=0)`
 - `template<typename _RandomAccessIterator, typename _Compare >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_Compare __comp)`
 - `template<typename _RAIter, typename _Compare >`
`void pop_heap (_RAIter, _RAIter, _Compare)`
 - `template<typename _RAIter >`
`void pop_heap (_RAIter, _RAIter)`
 - `template<typename _RandomAccessIterator >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
 - `template<typename _Tp >`
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
-

- `template<typename _Tp, typename _Up>`
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_`
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`
`>::__type, _Up >::__type pow (_Tp __x, _Up __y)`
- `template<typename _Tp>`
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow`
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp>`
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow`
`(const valarray< _Tp > &__v, const _Tp &__t)`
- `template<class _Dom1, class _Dom2>`
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _`
`Dom1::value_type > pow (const _Expr< _Dom1, typename _Dom1::value_`
`type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<typename _Tp>`
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<class _Dom>`
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > pow (const valarray< typename _`
`Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<typename _Tp>`
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`
- `long double pow (long double __x, long double __y)`
- `template<typename _Tp>`
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow`
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom>`
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_`
`type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _`
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`
`__v)`
- `template<class _Dom>`
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_`
`type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom>`
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _`
`Dom >, typename _Dom::value_type > pow (const typename _Dom::value_`
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `float pow (float __x, float __y)`
- `template<typename _BidirectionalIterator>`
`_BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_`
`traits< _BidirectionalIterator >::difference_type __n=1)`

- `template<typename _BIter, typename _Compare >`
`bool prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BIter >`
`bool prev_permutation (_BIter, _BIter)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _Tp >`
`std::complex< _Tp > proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type proj (_Tp __x)`
- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__x)(_Arg1, _Arg2))`
- `template<typename _RAIter >`
`void push_heap (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`
`void push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _MoneyT >`
`_Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __intl=false)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _RandomAccessIterator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Generator >`
`void random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RAIter >`
`void random_shuffle (_RAIter, _RAIter)`
- `template<typename _Tp >`
`_Tp real (const complex< _Tp > &__z)`

- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__value)`
- `template<typename _Filter, typename _Tp >`
`_Filter remove (_Filter, _Filter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Filter, typename _Predicate >`
`_Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last,`
`_Predicate __pred)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__-`
`old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`
`_OIter replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _-`
`Tp >`
`_OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate _-`
`__pred, const _Tp &__new_value)`
- `_Resetiosflags resetiosflags (ios_base::fmtflags __mask)`
- `void rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `template<typename _Ex >`
`void rethrow_if_nested (const _Ex &__ex)`

- void [rethrow_if_nested](#) (const [nested_exception](#) &__ex)
- template<typename _Tp >
void [return_temporary_buffer](#) (_Tp *__p)
- template<typename _BIter >
void **reverse** (_BIter, _BIter)
- template<typename _BidirectionalIterator >
void [reverse](#) (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BIter, typename _OIter >
_OIter **reverse_copy** (_BIter, _BIter, _OIter)
- template<typename _BidirectionalIterator, typename _OutputIterator >
_OutputIterator [reverse_copy](#) (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)
- [ios_base](#) & [right](#) ([ios_base](#) &__base)
- template<typename _Filter >
void **rotate** (_Filter, _Filter, _Filter)
- template<typename _ForwardIterator >
void [rotate](#) (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _OutputIterator >
_OutputIterator [rotate_copy](#) (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)
- template<typename _Filter, typename _OIter >
_OIter **rotate_copy** (_Filter, _Filter, _Filter, _OIter)
- [ios_base](#) & [scientific](#) ([ios_base](#) &__base)
- template<typename _ForwardIterator1, typename _ForwardIterator2 >
_ForwardIterator1 [search](#) (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >
_Filter1 **search** (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >
_ForwardIterator1 [search](#) (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _Filter1, typename _Filter2 >
_Filter1 **search** (_Filter1, _Filter1, _Filter2, _Filter2)
- template<typename _Filter, typename _Size, typename _Tp >
_Filter **search_n** (_Filter, _Filter, _Size, const _Tp &)
- template<typename _ForwardIterator, typename _Integer, typename _Tp >
_ForwardIterator [search_n](#) (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >
_Filter **search_n** (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)

- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `new_handler set_new_handler (new_handler) throw ()`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `terminate_handler set_terminate (terminate_handler) throw ()`
- `unexpected_handler set_unexpected (unexpected_handler) throw ()`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `_Setbase setbase (int __base)`
- `template<typename _CharT >`
`_Setfill< _CharT > setfill (_CharT __c)`
- `_Setiosflags setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision setprecision (int __n)`
- `_Setw setw (int __n)`
- `ios_base & showbase (ios_base & __base)`
- `ios_base & showpoint (ios_base & __base)`
- `ios_base & showpos (ios_base & __base)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`
`UniformRandomNumberGenerator && __g)`
- `template<typename _RAIter, typename _UGenerator >`
`void shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > sin (const valarray< _Tp >`
`& __v)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type sin`
`(_Tp __x)`
- `long double sin (long double __x)`
- `float sin (float __x)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > sin`
`(const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`
`complex< _Tp > sin (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > sinh (const valarray< _Tp`
`> & __v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type > sinh`
`(const _Expr< _Dom, typename _Dom::value_type > & __e)`

- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`sinh (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > sinh (const complex< _Tp > &)`
- `long double sinh (long double __x)`
- `float sinh (float __x)`
- `ios_base & skipws (ios_base &__base)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _-`
`Compare __comp)`
- `template<typename _RAIter >`
`void sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_Compare __comp)`
- `template<typename _RAIter >`
`void sort_heap (_RAIter, _RAIter)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp`
`> &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt`
`(const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`
`sqrt (_Tp __x)`
- `float sqrt (float __x)`
- `template<typename _Tp >`
`complex< _Tp > sqrt (const complex< _Tp > &)`
- `long double sqrt (long double __x)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator _-`
`__last, _Predicate __pred)`
- `template<typename _BIter, typename _Predicate >`
`_BIter stable_partition (_BIter, _BIter, _Predicate)`

- `template<typename _RandomAccessIterator >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter, _RAIter)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp`
`> &__r)`
- `double stod (const string &__str, size_t *__idx=0)`
- `float stof (const string &__str, size_t *__idx=0)`
- `int stoi (const string &__str, size_t *__idx=0, int __base=10)`
- `long stol (const string &__str, size_t *__idx=0, int __base=10)`
- `long double stold (const string &__str, size_t *__idx=0)`
- `long long stoll (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long stoul (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long long stoull (const string &__str, size_t *__idx=0, int __base=10)`
- `char * strchr (char *__s, int __n)`
- `char * strpbrk (char *__s1, const char *__s2)`
- `char * strchr (char *__s, int __n)`
- `char * strstr (char *__s1, const char *__s2)`
- `template<typename _Res, typename... _Args>`
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__-`
`y)`
- `template<typename _Tp, typename _Sequence, typename _Compare >`
`void swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_-
queue< _Tp, _Sequence, _Compare > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _`
`CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _`
`Key, _Tp, _Compare, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x,`
`unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp >`
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`

- `template<typename _Tp, size_t _Nm>`
`void swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _T1, class _T2 >`
`void swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Allocator >`
`void swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results< _Bi_iter, _Allocator > &__rhs)`
- `template<typename _Tp >`
`void swap (_Tp &__a, _Tp &__b)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, typename _Seq >`
`void swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`void swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Dp >`
`void swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Res >`
`void swap (promise< _Res > &__x, promise< _Res > &__y)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_multiset< _Value, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Res, typename... _ArgTypes>`
`void swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b)`
- `template<typename _Mutex >`
`void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `void swap (thread &__x, thread &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename... _Elements>`
`void swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`
`void swap (__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__x, __unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

- `_GLIBCXX_CONST` const [error_category](#) & [system_category](#) () throw ()
- long double **`tan`** (long double __x)
- float **`tan`** (float __x)
- template<typename _Tp >
[complex](#)< _Tp > **`tan`** (const [complex](#)< _Tp > &)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **`tan`**
(_Tp __x)
- template<class _Dom >
_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > **`tan`**
(const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > **`tan`** (const [valarray](#)< _Tp
> &__v)
- template<typename _Tp >
__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type
`tanh` (_Tp __x)
- template<typename _Tp >
_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > **`tanh`** (const [valarray](#)< _Tp
> &__v)
- template<class _Dom >
_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >
`tanh` (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
[complex](#)< _Tp > **`tanh`** (const [complex](#)< _Tp > &)
- long double **`tanh`** (long double __x)
- float **`tanh`** (float __x)
- void [terminate](#) () __attribute__((__noreturn__)) throw ()
- const _Rb_tree_node_base *__root **`throw`** ()
- template<typename _Ex >
void [throw_with_nested](#) (_Ex __ex)
- template<typename... _Elements>
[tuple](#)< _Elements &...> **`tie`** (_Elements &...__args)
- [string to_string](#) (float __val)
- [string to_string](#) (unsigned long long __val)
- [string to_string](#) (int __val)
- [string to_string](#) (long long __val)
- [string to_string](#) (unsigned __val)
- [string to_string](#) (unsigned long __val)
- [string to_string](#) (double __val)
- [string to_string](#) (long double __val)
- [string to_string](#) (long __val)
- [wstring to_wstring](#) (unsigned long __val)
- [wstring to_wstring](#) (unsigned long long __val)

- `wstring to_wstring` (long long __val)
- `wstring to_wstring` (int __val)
- `wstring to_wstring` (unsigned __val)
- `wstring to_wstring` (double __val)
- `wstring to_wstring` (float __val)
- `wstring to_wstring` (long double __val)
- `wstring to_wstring` (long __val)
- `template<typename _CharT >`
`_CharT tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`_CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _BinaryOperation >`
`_OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_`
`op)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`
`int try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> tuple_cat (tuple< _TElements...>`
`&&__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> tuple_cat (const tuple< _TElements...>`
`&__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> tuple_cat (const tuple< _TElements...>`
`&__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>`
`tuple< _TElements..., _UElements...> tuple_cat (tuple< _TElements...>`
`&&__t, tuple< _UElements...> &&__u)`
- `bool uncaught_exception () __attribute__((__pure__)) throw ()`
- `void unexpected () __attribute__((__noreturn__))`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __-`
`last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _-`
`ForwardIterator __result)`

- `template<typename _ForwardIterator, typename _Tp >`
`void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _-`
`BinaryPredicate __binary_pred)`
- `template<typename _FIter, typename _BinaryPredicate >`
`_FIter unique (_FIter, _FIter, _BinaryPredicate)`
- `template<typename _FIter >`
`_FIter unique (_FIter, _FIter)`
- `template<typename _ForwardIterator >`
`_ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _IIter, typename _OIter, typename _BinaryPredicate >`
`_OIter unique_copy (_IIter, _IIter, _OIter, _BinaryPredicate)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result)`
- `template<typename _IIter, typename _OIter >`
`_OIter unique_copy (_IIter, _IIter, _OIter)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _BinaryPredicate __binary_pred)`
- `ios_base & unitbuf (ios_base &__base)`
- `template<typename _FIter, typename _Tp, typename _Compare >`
`_FIter upper_bound (_FIter, _FIter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __-`
`last, const _Tp &__val, _Compare __comp)`
- `template<typename _FIter, typename _Tp >`
`_FIter upper_bound (_FIter, _FIter, const _Tp &)`
- `ios_base & uppercase (ios_base &__base)`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &__loc)`
- `wchar_t * wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t * wcspbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t * wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > & _is)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, bitset< _Nb > &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const bitset< _Nb > &__x)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator- (const _Tp &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`complex< _Tp > operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool operator== (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`bool operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`bool operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`bool operator!= (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`reference_wrapper< _Tp > ref (_Tp &__t)`
- `template<typename _Tp >`
`reference_wrapper< const _Tp > cref (const _Tp &__t)`
- `template<typename _Tp >`
`reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t)`
- `template<typename _Tp >`
`reference_wrapper< const _Tp > cref (reference_wrapper< _Tp > __t)`

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`

- `template<typename _CharT, typename _Traits >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`

- `template<>`
`basic_istream< char > & operator>> (basic_istream< char > &__in, char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`
- `template<class _Traits >`
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char *__s)`

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<class _Traits >`
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits >`

```

bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _-
Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,
regex_constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >
bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex<
_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __-
flags=regex_constants::match_default)
• template<typename _Ch_type, typename _Allocator, typename _Rx_traits >
bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *,
_Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,
regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_-
type, typename _Rx_traits >
bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc >
&__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _-
Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_-
type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_-
constants::match_default)
• template<typename _Ch_type, class _Rx_traits >
bool regex_match (const _Ch_type *__s, const basic_regex< _Ch_-
type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_-
constants::match_default)
• template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _-
Rx_traits >
bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator
> &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_-
constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits
>
bool regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_-
iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,
regex_constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >
bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex<
_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __-
flags=regex_constants::match_default)
• template<typename _Ch_type, class _Allocator, class _Rx_traits >
bool regex_search (const _Ch_type *__s, match_results< const _Ch_type
*, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e,
regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Ch_type, typename _Rx_traits >
bool regex_search (const _Ch_type *__s, const basic_regex< _Ch_-
type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_-
constants::match_default)
• template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename
_Rx_traits >

```

- bool [regex_search](#) (const [basic_string](#)< _Ch_type, _Ch_traits, _String_allocator > &__s, const [basic_regex](#)< _Ch_type, _Rx_traits > &__e, [regex_constants::match_flag_type](#) __flags=[regex_constants::match_default](#))
- template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_type, typename _Rx_traits >
bool [regex_search](#) (const [basic_string](#)< _Ch_type, _Ch_traits, _Ch_alloc > &__s, [match_results](#)< typename [basic_string](#)< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Allocator > &__m, const [basic_regex](#)< _Ch_type, _Rx_traits > &__e, [regex_constants::match_flag_type](#) __f=[regex_constants::match_default](#))
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >
_Out_iter [regex_replace](#) (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const [basic_regex](#)< _Ch_type, _Rx_traits > &__e, const [basic_string](#)< _Ch_type > &__fmt, [regex_constants::match_flag_type](#) __flags=[regex_constants::match_default](#))
- template<typename _Rx_traits, typename _Ch_type >
[basic_string](#)< _Ch_type > [regex_replace](#) (const [basic_string](#)< _Ch_type > &__s, const [basic_regex](#)< _Ch_type, _Rx_traits > &__e, const [basic_string](#)< _Ch_type > &__fmt, [regex_constants::match_flag_type](#) __flags=[regex_constants::match_default](#))

Variables

- [enable_if](#)< (!is_member_pointer< _Functor >::value &&!is_function< _Functor >::value &&!is_function< typename remove_pointer< _Functor >::type >::value), typename result_of< _Functor(_Args...)>::type >::type [_invoke](#) (_Functor &__f, _Args &&...__args)
- static ios_base::Init [__ioinit](#)
- function< void()> [__once_function](#)
- [std::binder1st _GLIBCXX_DEPRECATED_ATTR](#)
- const [adopt_lock_t](#) [adopt_lock](#)
- static const [allocator_arg_t](#) [allocator_arg](#)
- const [defer_lock_t](#) [defer_lock](#)
- const [error_category](#) *const [future_category](#)
- const _Swallow_assign [ignore](#)
- [error_code](#) [make_error_code](#) (errc)
- [error_condition](#) [make_error_condition](#) (errc)
- const nothrow_t [nothrow](#)
- static const piecewise_construct_t [piecewise_construct](#)
- const [try_to_lock_t](#) [try_to_lock](#)

Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the *I/O forward declarations*

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the *HOWTO* linked to above.

- [istream cin](#)
- [ostream cout](#)
- [ostream cerr](#)
- [ostream clog](#)
- [wistream wcin](#)
- [wostream wcout](#)
- [wostream wcerr](#)
- [wostream wclog](#)

4.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

4.11.2 Typedef Documentation

4.11.2.1 `typedef void(* std::new_handler)()`

If you write your own error handler to be called by `new`, it must be of this type.

Definition at line 75 of file `new`.

4.11.2.2 `typedef long long std::streamoff`

Type used by `fpos`, [char_traits<char>](#), and [char_traits<wchar_t>](#).

In clauses 21.1.3.1 and 27.4.1 `streamoff` is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, `streamoff` was typedef `long`.

Definition at line 94 of file `postypes.h`.

4.11.2.3 `typedef fpos<mbstate_t> std::streampos`

File position for char streams.

Definition at line 228 of file `postypes.h`.

4.11.2.4 `typedef ptrdiff_t std::streamsize`

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file `postypes.h`.

4.11.2.5 `typedef fpos<mbstate_t> std::u16streampos`

File position for `char16_t` streams.

Definition at line 234 of file `postypes.h`.

4.11.2.6 `typedef fpos<mbstate_t> std::u32streampos`

File position for `char32_t` streams.

Definition at line 236 of file `postypes.h`.

4.11.2.7 `typedef fpos<mbstate_t> std::wstreampos`

File position for `wchar_t` streams.

Definition at line 230 of file `postypes.h`.

4.11.3 Enumeration Type Documentation

4.11.3.1 anonymous enum

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more. This controls some aspect of the sort routines.

Definition at line 2167 of file `stl_algo.h`.

4.11.3.2 `enum std::float_denorm_style`

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the `std::numeric_limits` class.

Enumerator:

denorm_indeterminate Indeterminate at compile time whether denormalized values are allowed.

denorm_absent The type does not allow denormalized values.

denorm_present The type allows denormalized values.

Definition at line 170 of file limits.

4.11.3.3 enum std::float_round_style

Describes the rounding style for floating-point types.

This is used in the [std::numeric_limits](#) class.

Enumerator:

round_indeterminate Self-explanatory.

round_toward_zero Self-explanatory.

round_to_nearest To the nearest representable value.

round_toward_infinity Self-explanatory.

round_toward_neg_infinity Self-explanatory.

Definition at line 155 of file limits.

4.11.4 Function Documentation

4.11.4.1 `template<typename _RandomAccessIterator > void
std::__final_insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2172 of file stl_algo.h.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

Referenced by `sort()`.

4.11.4.2 `template<typename _RandomAccessIterator , typename _Compare >
void std::__final_insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2187 of file stl_algo.h.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

4.11.4.3 `template<typename _InputIterator , typename _Tp > _InputIterator
std::__find (_InputIterator __first, _InputIterator __last, const _Tp
& __val, input_iterator_tag) [inline]`

This is an overload used by `find()` for the Input Iterator case.

Definition at line 128 of file `stl_algo.h`.

Referenced by `find()`.

4.11.4.4 `template<typename _RandomAccessIterator , typename _Tp >
_RandomAccessIterator std::__find (_RandomAccessIterator
__first, _RandomAccessIterator __last, const _Tp & __val,
random_access_iterator_tag)`

This is an overload used by `find()` for the RAI case.

Definition at line 150 of file `stl_algo.h`.

4.11.4.5 `template<typename _InputIterator , typename _Predicate >
_InputIterator std::__find_if (_InputIterator __first, _InputIterator
__last, _Predicate __pred, input_iterator_tag) [inline]`

This is an overload used by `find_if()` for the Input Iterator case.

Definition at line 139 of file `stl_algo.h`.

Referenced by `find_if()`.

4.11.4.6 `template<typename _RandomAccessIterator , typename _Predicate >
_RandomAccessIterator std::__find_if (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Predicate __pred,
random_access_iterator_tag)`

This is an overload used by `find_if()` for the RAI case.

Definition at line 198 of file `stl_algo.h`.

4.11.4.7 `template<typename _InputIterator , typename _Predicate >
_InputIterator std::__find_if_not (_InputIterator __first,
_InputIterator __last, _Predicate __pred, input_iterator_tag)
[inline]`

This is an overload used by `find_if_not()` for the Input Iterator case.

Definition at line 247 of file `stl_algo.h`.

Referenced by `find_if_not()`.

4.11.4.8 `template<typename _RandomAccessIterator, typename _Predicate >
_RandomAccessIterator std::__find_if_not (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Predicate __pred,
random_access_iterator_tag)`

This is an overload used by `find_if_not()` for the RAI case.

Definition at line 258 of file `stl_algo.h`.

4.11.4.9 `template<typename _EuclideanRingElement >
_EuclideanRingElement std::__gcd (_EuclideanRingElement __m,
_EuclideanRingElement __n)`

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1487 of file `stl_algo.h`.

4.11.4.10 `template<typename _RandomAccessIterator, typename _Compare
> void std::__heap_select (_RandomAccessIterator __first,
_RandomAccessIterator __middle, _RandomAccessIterator __last,
_Compare __comp)`

This is a helper function for the sort routines.

Definition at line 1908 of file `stl_algo.h`.

References `make_heap()`.

4.11.4.11 `template<typename _RandomAccessIterator > void
std::__heap_select (_RandomAccessIterator __first,
_RandomAccessIterator __middle, _RandomAccessIterator __last)`

This is a helper function for the sort routines.

Definition at line 1895 of file `stl_algo.h`.

References `make_heap()`.

Referenced by `partial_sort()`.

4.11.4.12 `template<typename _ForwardIterator, typename
_Predicate, typename _Distance > _ForwardIterator
std::__inplace_stable_partition (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred, _Distance __len)`

This is a helper function...

Definition at line 1772 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

4.11.4.13 `template<typename _RandomAccessIterator > void
std::__inplace_stable_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last)`

This is a helper function for the stable sorting routines.

Definition at line 3346 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

Referenced by `__inplace_stable_sort()`, and `stable_sort()`.

4.11.4.14 `template<typename _RandomAccessIterator, typename _Compare
> void std::__inplace_stable_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the stable sorting routines.

Definition at line 3365 of file `stl_algo.h`.

References `__inplace_stable_sort()`, `__insertion_sort()`, and `__merge_without_buffer()`.

4.11.4.15 `template<typename _RandomAccessIterator > void
std::__insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2095 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`, and `__inplace_stable_sort()`.

4.11.4.16 `template<typename _RandomAccessIterator, typename _Compare
> void std::__insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2118 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

4.11.4.17 `template<typename _RandomAccessIterator, typename _Size
> void std::__introsort_loop (_RandomAccessIterator __first,
_RandomAccessIterator __last, _Size __depth_limit)`

This is a helper function for the sort routine.

Definition at line 2267 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`.

Referenced by `__introsort_loop()`, and `sort()`.

4.11.4.18 `template<typename _RandomAccessIterator, typename
_Size, typename _Compare > void std::__introsort_loop (
_RandomAccessIterator __first, _RandomAccessIterator __last,
_Size __depth_limit, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2289 of file `stl_algo.h`.

References `__introsort_loop()`, and `__unguarded_partition_pivot()`.

4.11.4.19 `template<typename _Size > _Size std::__lg (_Size __n)
[inline]`

This is a helper function for the sort routines and for [random.tcc](#).

Definition at line 970 of file `stl_algobase.h`.

Referenced by `nth_element()`, `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`, and `sort()`.

4.11.4.20 `template<typename _BidirectionalIterator , typename
_Distance , typename _Pointer > void std::__merge_adaptive (`
`_BidirectionalIterator __first, _BidirectionalIterator __middle,`
`_BidirectionalIterator __last, _Distance __len1, _Distance __len2,`
`_Pointer __buffer, _Distance __buffer_size)`

This is a helper function for the merge routines.

Definition at line 2822 of file `stl_algo.h`.

References `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

Referenced by `__merge_adaptive()`, and `inplace_merge()`.

4.11.4.21 `template<typename _BidirectionalIterator , typename
_Distance , typename _Pointer , typename _Compare >`
`void std::__merge_adaptive (_BidirectionalIterator __first,`
`_BidirectionalIterator __middle, _BidirectionalIterator __last,`
`_Distance __len1, _Distance __len2, _Pointer __buffer, _Distance`
`__buffer_size, _Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2884 of file `stl_algo.h`.

References `__merge_adaptive()`, `__merge_backward()`, `__rotate_adaptive()`, `advance()`, `distance()`, `lower_bound()`, and `upper_bound()`.

4.11.4.22 `template<typename _BidirectionalIterator1 , typename
_BidirectionalIterator2 , typename _BidirectionalIterator3`
`> _BidirectionalIterator3 std::__merge_backward (`
`_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1,`
`_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2,`
`_BidirectionalIterator3 __result)`

This is a helper function for the merge routines.

Definition at line 2719 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

4.11.4.23 `template<typename _BidirectionalIterator1 , typename
_BidirectionalIterator2 , typename _BidirectionalIterator3
, typename _Compare > _BidirectionalIterator3
std::__merge_backward (_BidirectionalIterator1 __first1,
_BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,
_BidirectionalIterator2 __last2, _BidirectionalIterator3 __result,
_Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2754 of file `stl_algo.h`.

4.11.4.24 `template<typename _BidirectionalIterator , typename _Distance >
void std::__merge_without_buffer (_BidirectionalIterator __first,
_BidirectionalIterator __middle, _BidirectionalIterator __last,
_Distance __len1, _Distance __len2)`

This is a helper function for the merge routines.

Definition at line 2947 of file `stl_algo.h`.

References `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

Referenced by `__inplace_stable_sort()`, `__merge_without_buffer()`, and `inplace_merge()`.

4.11.4.25 `template<typename _BidirectionalIterator , typename _Distance
, typename _Compare > void std::__merge_without_buffer (
_BidirectionalIterator __first, _BidirectionalIterator __middle,
_BidirectionalIterator __last, _Distance __len1, _Distance __len2,
_Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2991 of file `stl_algo.h`.

References `__merge_without_buffer()`, `advance()`, `distance()`, `iter_swap()`, `lower_bound()`, `rotate()`, and `upper_bound()`.

4.11.4.26 `template<typename _Iterator > void std::__move_median_first (`
`_Iterator __a, _Iterator __b, _Iterator __c)`

Swaps the median value of `*__a`, `*__b` and `*__c` to `*__a`.

Definition at line 76 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

4.11.4.27 `template<typename _Iterator , typename _Compare > void
std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator
__c, _Compare __comp)`

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__a`.

Definition at line 100 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.28 `template<typename _BidirectionalIterator , typename _Predicate
> _BidirectionalIterator std::__partition (_BidirectionalIterator
__first, _BidirectionalIterator __last, _Predicate __pred,
bidirectional_iterator_tag)`

This is a helper function...

Definition at line 1742 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.29 `template<typename _ForwardIterator , typename _Predicate >
_ForwardIterator std::__partition (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred, forward_iterator_tag
)`

This is a helper function...

Definition at line 1717 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `partition()`.

4.11.4.30 `template<typename _RandomAccessIterator > void std::__reverse (
_RandomAccessIterator __first, _RandomAccessIterator __last,
random_access_iterator_tag)`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for random access iterators.

Definition at line 1407 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.31 `template<typename _BidirectionalIterator > void std::__reverse
(_BidirectionalIterator __first, _BidirectionalIterator __last,
bidirectional_iterator_tag)`

This is an uglified reverse(_BidirectionalIterator, _BidirectionalIterator) overloaded for bidirectional iterators.

Definition at line 1387 of file stl_algo.h.

References iter_swap().

Referenced by __rotate(), and reverse().

4.11.4.32 `template<typename _ForwardIterator > void std::__rotate
(_ForwardIterator __first, _ForwardIterator __middle,
_ForwardIterator __last, forward_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1501 of file stl_algo.h.

References iter_swap().

Referenced by __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object(), and rotate().

4.11.4.33 `template<typename _BidirectionalIterator > void std::__rotate (`
`_BidirectionalIterator __first, _BidirectionalIterator __middle,`
`_BidirectionalIterator __last, bidirectional_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1537 of file stl_algo.h.

References __reverse(), and iter_swap().

4.11.4.34 `template<typename _RandomAccessIterator > void std::__rotate (`
`_RandomAccessIterator __first, _RandomAccessIterator __middle,`
`_RandomAccessIterator __last, random_access_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1567 of file stl_algo.h.

References iter_swap(), swap(), and swap_ranges().

4.11.4.35 `template<typename _BidirectionalIterator1, typename
_BidirectionalIterator2, typename _Distance >
_BidirectionalIterator1 std::__rotate_adaptive (
_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle,
_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2,
_BidirectionalIterator2 __buffer, _Distance __buffer_size)`

This is a helper function for the merge routines.

Definition at line 2790 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `__merge_adaptive()`.

4.11.4.36 `template<typename _ForwardIterator, typename _Integer
, typename _Tp > _ForwardIterator std::__search_n (
_ForwardIterator __first, _ForwardIterator __last, _Integer
__count, const _Tp & __val, std::forward_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for forward iterators.

Definition at line 324 of file `stl_algo.h`.

Referenced by `search_n()`.

4.11.4.37 `template<typename _RandomAccessIter, typename _Integer
, typename _Tp > _RandomAccessIter std::__search_n (
_RandomAccessIter __first, _RandomAccessIter __last, _Integer
__count, const _Tp & __val, std::random_access_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&)` overloaded for random access iterators.

Definition at line 356 of file `stl_algo.h`.

4.11.4.38 `template<typename _ForwardIterator, typename _Integer ,
typename _Tp, typename _BinaryPredicate > _ForwardIterator
std::__search_n (_ForwardIterator __first, _ForwardIterator
__last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred, std::forward_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for forward iterators.

Definition at line 410 of file `stl_algo.h`.

4.11.4.39 `template<typename _RandomAccessIter , typename _Integer ,
typename _Tp , typename _BinaryPredicate > _RandomAccessIter
std::__search_n (_RandomAccessIter __first, _RandomAccessIter
__last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred, std::random_access_iterator_tag)`

This is an uglified `search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate)` overloaded for random access iterators.

Definition at line 449 of file `stl_algo.h`.

4.11.4.40 `template<typename _ForwardIterator , typename _Pointer ,
typename _Predicate , typename _Distance > _ForwardIterator
std::__stable_partition_adaptive (_ForwardIterator __first,
_ForwardIterator __last, _Predicate __pred, _Distance __len,
_Pointer __buffer, _Distance __buffer_size)`

This is a helper function...

Definition at line 1797 of file `stl_algo.h`.

References `advance()`, `distance()`, and `rotate()`.

Referenced by `stable_partition()`.

4.11.4.41 `template<typename _RandomAccessIterator > void
std::__unguarded_insertion_sort (_RandomAccessIterator __first,
_RandomAccessIterator __last) [inline]`

This is a helper function for the sort routine.

Definition at line 2140 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

4.11.4.42 `template<typename _RandomAccessIterator , typename _Compare
> void std::__unguarded_insertion_sort (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp)
[inline]`

This is a helper function for the sort routine.

Definition at line 2153 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

4.11.4.43 `template<typename _RandomAccessIterator > void
std::__unguarded_linear_insert (_RandomAccessIterator __last)`

This is a helper function for the sort routine.

Definition at line 2058 of file `stl_algo.h`.

Referenced by `__insertion_sort()`, and `__unguarded_insertion_sort()`.

4.11.4.44 `template<typename _RandomAccessIterator , typename _Compare
> void std::__unguarded_linear_insert (_RandomAccessIterator
__last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 2076 of file `stl_algo.h`.

4.11.4.45 `template<typename _RandomAccessIterator , typename
_Tp > _RandomAccessIterator std::__unguarded_partition (
_RandomAccessIterator __first, _RandomAccessIterator __last,
const _Tp & __pivot)`

This is a helper function...

Definition at line 2203 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

4.11.4.46 `template<typename _RandomAccessIterator , typename
_Tp , typename _Compare > _RandomAccessIterator
std::__unguarded_partition (_RandomAccessIterator __first,
_RandomAccessIterator __last, const _Tp & __pivot, _Compare
__comp)`

This is a helper function...

Definition at line 2223 of file `stl_algo.h`.

References `iter_swap()`.

4.11.4.47 `template<typename _RandomAccessIterator >
 _RandomAccessIterator std::__unguarded_partition_pivot (
 _RandomAccessIterator __first, _RandomAccessIterator __last)
 [inline]`

This is a helper function...

Definition at line 2244 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

4.11.4.48 `template<typename _RandomAccessIterator , typename _Compare
 > _RandomAccessIterator std::__unguarded_partition_pivot (
 _RandomAccessIterator __first, _RandomAccessIterator __last,
 _Compare __comp) [inline]`

This is a helper function...

Definition at line 2256 of file `stl_algo.h`.

References `__move_median_first()`, and `__unguarded_partition()`.

4.11.4.49 `template<typename _InputIterator , typename _ForwardIterator
 > _ForwardIterator std::__unique_copy (_InputIterator
 __first, _InputIterator __last, _ForwardIterator __result,
 input_iterator_tag , forward_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and forward iterator as result.

Definition at line 1285 of file `stl_algo.h`.

4.11.4.50 `template<typename _ForwardIterator , typename _OutputIterator ,
 typename _BinaryPredicate > _OutputIterator std::__unique_copy (
 _ForwardIterator __first, _ForwardIterator __last, _OutputIterator
 __result, _BinaryPredicate __binary_pred, forward_iterator_tag ,
 output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1306 of file `stl_algo.h`.

4.11.4.51 `template<typename _ForwardIterator, typename _OutputIterator
> _OutputIterator std::__unique_copy (_ForwardIterator
__first, _ForwardIterator __last, _OutputIterator __result,
forward_iterator_tag, output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for forward iterators and output iterator as result.

Definition at line 1239 of file `stl_algo.h`.

Referenced by `unique_copy()`.

4.11.4.52 `template<typename _InputIterator, typename _ForwardIterator,
typename _BinaryPredicate > _ForwardIterator std::__unique_copy
(_InputIterator __first, _InputIterator __last, _ForwardIterator
__result, _BinaryPredicate __binary_pred, input_iterator_tag,
forward_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1364 of file `stl_algo.h`.

4.11.4.53 `template<typename _InputIterator, typename _OutputIterator
> _OutputIterator std::__unique_copy (_InputIterator __first,
_InputIterator __last, _OutputIterator __result, input_iterator_tag
, output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator)` overloaded for input iterators and output iterator as result.

Definition at line 1262 of file `stl_algo.h`.

4.11.4.54 `template<typename _InputIterator, typename _OutputIterator,
typename _BinaryPredicate > _OutputIterator std::__unique_copy (_InputIterator
__first, _InputIterator __last, _OutputIterator
__result, _BinaryPredicate __binary_pred, input_iterator_tag,
output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1335 of file `stl_algo.h`.

4.11.4.55 `template<typename _T1, typename... _Args> void std::_Construct (_T1 * __p, _Args &&... __args) [inline]`

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 73 of file `stl_construct.h`.

4.11.4.56 `template<typename _Tp > void std::_Destroy (_Tp * __pointer) [inline]`

Destroy the object pointed to by a pointer type.

Definition at line 91 of file `stl_construct.h`.

Referenced by `std::deque< _Tp, _Alloc >::_M_fill_initialize()`, `std::deque< _Tp, _Alloc >::_M_range_initialize()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::vector< _Tp, _Alloc >::reserve()`, and `std::vector< std::sub_match< _Bi_iter >, _Allocator >::~~vector()`.

4.11.4.57 `template<typename _ForwardIterator > void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last) [inline]`

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 121 of file `stl_construct.h`.

4.11.4.58 `template<typename _InputIterator, typename _Tp > _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init) [inline]`

Accumulate values in a range.

Accumulates the values in the range `[first,last)` using `operator+()`. The initial value is *init*. The values are processed in order.

Parameters

first Start of range.

last End of range.

init Starting value to add other values to.

Returns

The final sum.

Definition at line 116 of file stl_numeric.h.

4.11.4.59 `template<typename _InputIterator , typename _Tp , typename
_BinaryOperation > _Tp std::accumulate (_InputIterator __first,
_InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
[inline]`

Accumulate values in a range with operation.

Accumulates the values in the range [first,last) using the function object *binary_op*. The initial value is *init*. The values are processed in order.

Parameters

first Start of range.

last End of range.

init Starting value to add other values to.

binary_op Function object to accumulate with.

Returns

The final sum.

Definition at line 142 of file stl_numeric.h.

4.11.4.60 `template<typename _Tp > _Tp* std::addressof (_Tp & __r)
[inline]`

declval, from type_traits.

Returns the actual address of the object or function referenced by r, even in the presence of an overloaded operator&.

Parameters

__r Reference to an object or function.

Returns

The actual address.

Definition at line 100 of file move.h.

4.11.4.61 `template<typename _InputIterator, typename _OutputIterator >
_OutputIterator std::adjacent_difference (_InputIterator __first,
_InputIterator __last, _OutputIterator __result)`

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using operator-() and writes the result to *result*.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 539. partial_sum and adjacent_difference should mention requirements

Definition at line 312 of file stl_numeric.h.

4.11.4.62 `template<typename _InputIterator, typename _OutputIterator
, typename _BinaryOperation > _OutputIterator
std::adjacent_difference (_InputIterator __first, _InputIterator
__last, _OutputIterator __result, _BinaryOperation __binary_op)`

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using the function object *binary_op* and writes the result to *result*.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 539. partial_sum and adjacent_difference should mention requirements

Definition at line 354 of file stl_numeric.h.

4.11.4.63 `template<typename _InputIterator , typename _Distance > void
std::advance (_InputIterator & __i, _Distance __n) [inline]`

A generalization of pointer arithmetic.

Parameters

- i* An input iterator.
- n* The *delta* by which to change *i*.

Returns

Nothing.

This increments *i* by *n*. For bidirectional and random access iterators, *n* may be negative, in which case *i* is decremented.

For random access iterators, this uses their + and - operations and are constant time. For other iterator classes they are linear time.

Definition at line 169 of file `std_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque< _Tp, _Alloc >::_M_range_initialize()`, `equal_range()`, `lower_bound()`, `partition_point()`, and `upper_bound()`.

4.11.4.64 `template<class _Container > auto std::begin (_Container & __cont
) [inline]`

Return an iterator pointing to the first element of the container.

Parameters

- cont* Container.

Definition at line 46 of file `range_access.h`.

4.11.4.65 `template<class _Container > auto std::begin (const _Container &
__cont) [inline]`

Return an iterator pointing to the first element of the const container.

Parameters

- cont* Container.

Definition at line 56 of file range_access.h.

4.11.4.66 `template<class _Tp> const _Tp* std::begin (initializer_list< _Tp > __ils) [inline]`

Return an iterator pointing to the first element of the initializer_list.

Parameters

il Initializer list.

Definition at line 86 of file initializer_list.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::crend()`, `std::list< _Tp, _Alloc >::crend()`, `std::basic_string< char >::crend()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::empty()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::front()`, `std::list< _Tp, _Alloc >::front()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::pop_front()`, `std::list< _Tp, _Alloc >::push_front()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::rend()`, `std::list< _Tp, _Alloc >::rend()`, `std::basic_string< char >::rend()`, and `std::list< _Tp, _Alloc >::size()`.

4.11.4.67 `template<class _Tp, size_t _Nm> _Tp* std::begin (_Tp(& __arr[_Nm]) [inline]`

Return an iterator pointing to the first element of the array.

Parameters

arr Array.

Definition at line 85 of file range_access.h.

4.11.4.68 `template<typename _Result, typename _Functor, typename... _ArgTypes> _Bindres_helper<_Result, _Functor, _ArgTypes...>::type std::bind (_Functor && __f, _ArgTypes &&... __args) [inline]`

Function template for `std::bind<R>`.

Definition at line 1457 of file functional.

4.11.4.69 ios_base& std::boolalpha (ios_base & __base) [inline]

Calls base.setf(ios_base::boolalpha).

Definition at line 794 of file ios_base.h.

References std::ios_base::setf().

Referenced by noboolalpha().

**4.11.4.70 template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::const_pointer_cast (const
__shared_ptr<_Tp1, _Lp> & __r) [inline]**

const_pointer_cast

Definition at line 946 of file shared_ptr_base.h.

**4.11.4.71 template<typename _Tp> reference_wrapper<const _Tp> std::cref
(const _Tp & __t) [inline]**

Denotes a const reference should be taken to a variable.

Definition at line 474 of file functional.

Referenced by cref().

**4.11.4.72 template<typename _Tp> reference_wrapper<const _Tp> std::cref
(reference_wrapper<_Tp> __t) [inline]**

Partial specialization.

Definition at line 486 of file functional.

References cref().

4.11.4.73 ios_base& std::dec (ios_base & __base) [inline]

Calls base.setf(ios_base::dec, ios_base::basefield).

Definition at line 932 of file ios_base.h.

References std::ios_base::setf().

Referenced by operator>>().

4.11.4.74 `template<typename _InputIterator > iterator_traits<_InputIterator>::difference_type std::distance (_InputIterator __first, _InputIterator __last) [inline]`

A generalization of pointer arithmetic.

Parameters

first An input iterator.

last An input iterator.

Returns

The distance between them.

Returns *n* such that *first* + *n* == *last*. This requires that *last* must be reachable from *first*. Note that *n* may be negative.

For random access iterators, this uses their + and - operations and are constant time. For other iterator classes they are linear time.

Definition at line 111 of file `std_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__stable_partition_adaptive()`, `std::deque<_Tp, _Alloc>::M_range_initialize()`, `equal_range()`, `inplace_merge()`, `is_heap_until()`, `std::sub_match<_Bi_iter>::length()`, `lower_bound()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `partition_point()`, `std::match_results<_FwdIterT, _Alloc>::position()`, `__gnu_cxx::random_sample_n()`, and `upper_bound()`.

4.11.4.75 `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (const __shared_ptr<_Tp1, _Lp> & __r) [inline]`

`dynamic_pointer_cast`

Definition at line 956 of file `shared_ptr_base.h`.

4.11.4.76 `template<class _Tp, size_t _Nm> _Tp* std::end (_Tp(&) __arr[_Nm]) [inline]`

Return an iterator pointing to one past the last element of the array.

Parameters

arr Array.

Definition at line 95 of file range_access.h.

4.11.4.77 `template<class _Container > auto std::end (const _Container & __cont) [inline]`

Return an iterator pointing to one past the last element of the const container.

Parameters

cont Container.

Definition at line 76 of file range_access.h.

4.11.4.78 `template<class _Tp > const _Tp* std::end (initializer_list< _Tp > __ils) [inline]`

Return an iterator pointing to one past the last element of the initializer_list.

Parameters

il Initializer list.

Definition at line 96 of file initializer_list.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::at()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::back()`, `std::list< _Tp, _Alloc >::back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::crbegin()`, `std::list< _Tp, _Alloc >::crbegin()`, `std::basic_string< char >::crbegin()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::map< _Key, _Tp, _Compare, _Alloc >::operator[]()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::push_back()`, `std::list< _Tp, _Alloc >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::rbegin()`, `std::list< _Tp, _Alloc >::rbegin()`, `std::basic_string< char >::rbegin()`, `std::vector< std::sub_match< _Bi_iter >, _Allocator >::resize()`, and `std::list< _Tp, _Alloc >::size()`.

4.11.4.79 `template<class _Container > auto std::end (_Container & __cont) [inline]`

Return an iterator pointing to one past the last element of the container.

Parameters

cont Container.

Definition at line 66 of file range_access.h.

4.11.4.80 `template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::endl (basic_ostream<
_CharT, _Traits > & __os) [inline]`

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple new-line is desired, leading to poor buffering performance. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this subject.

Definition at line 541 of file ostream.

References flush(), std::basic_ostream< _CharT, _Traits >::put(), and std::basic_ios< _CharT, _Traits >::widen().

4.11.4.81 `template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::ends (basic_ostream<
_CharT, _Traits > & __os) [inline]`

Write a null character into the output sequence.

Null character is CharT() by definition. For CharT of char, this correctly writes the ASCII NUL character string terminator.

Definition at line 552 of file ostream.

References std::basic_ostream< _CharT, _Traits >::put().

4.11.4.82 `ios_base& std::fixed (ios_base & __base) [inline]`

Calls base.setf(ios_base::fixed, ios_base::floatfield).

Definition at line 957 of file ios_base.h.

References std::ios_base::setf().

4.11.4.83 `template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::flush (basic_ostream<
_CharT, _Traits > & __os) [inline]`

Flushes the output stream.

This manipulator simply calls the stream's flush() member function.

Definition at line 562 of file ostream.

References std::basic_ostream< _CharT, _Traits >::flush().

Referenced by endl().

4.11.4.84 `template<typename _Tp > enable_if<!is_lvalue_reference<_Tp>::value, _Tp&&>::type std::forward (typename std::common_type< _Tp >::type & __t) [inline]`

forward (as per N2835) /// Forward lvalues as rvalues.

Definition at line 58 of file move.h.

4.11.4.85 `template<typename _Tp > enable_if<!is_lvalue_reference<_Tp>::value, _Tp&&>::type std::forward (typename std::common_type< _Tp >::type && __t) [inline]`

Forward rvalues as rvalues.

Definition at line 64 of file move.h.

4.11.4.86 `template<typename _MoneyT > _Get_money<_MoneyT> std::get_money (_MoneyT & __mon, bool __intl = false) [inline]`

Extended manipulator for extracting money.

Parameters

mon Either long double or a specialization of `basic_string`.

intl A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts *mon*.

Definition at line 256 of file iomanip.

4.11.4.87 `template<typename _Tp > pair<_Tp*, ptrdiff_t> std::get_temporary_buffer (ptrdiff_t __len)`

Allocates a temporary buffer.

Parameters

len The number of objects of type *Tp*.

Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of `sizeof(Tp)`)*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the `nothrow` exception guarantee.

Definition at line 84 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`.

4.11.4.88 `template<typename _CharT, typename _Traits, typename _Alloc >
basic_istream<_CharT, _Traits> & std::getline (basic_istream<
_CharT, _Traits> & __is, basic_string<_CharT, _Traits, _Alloc >
& __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

is Input stream.

str Buffer to store into.

delim Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *is* into *str* until *delim* is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased. If *delim* was encountered, it is extracted but not stored into *str*.

Definition at line 1068 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::erase()`, `std::basic_string<_CharT, _Traits, _Alloc >::max_size()`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, and `std::basic_ios<_CharT, _Traits >::setstate()`.

4.11.4.89 `template<typename _CharT, typename _Traits, typename _Alloc >
basic_istream<_CharT, _Traits> & std::getline (basic_istream<
_CharT, _Traits> & __is, basic_string<_CharT, _Traits, _Alloc >
& __str) [inline]`

Read a line from stream into a string.

Parameters

is Input stream.
str Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *is* into *str* until ‘

’ is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *str*. Any previous contents of *str* are erased. If end of line was encountered, it is extracted but not stored into *str*.

Definition at line 2682 of file `basic_string.h`.

References `std::basic_ios<_CharT, _Traits>::widen()`.

4.11.4.90 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

__is Input stream.
__str Buffer to store into.
__delim Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until *__delim* is found, the end of the stream is encountered, or *str.max_size()* is reached. If *is.width()* is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If *delim* was encountered, it is extracted but not stored into *__str*.

Definition at line 622 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::max_size()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.11.4.91 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream<_CharT, _Traits>& std::getline (basic_istream<_CharT, _Traits> & __is, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Read a line from stream into a string.

Parameters

`__is` Input stream.

`__str` Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `is` into `__str` until ' '

' is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If end of line was encountered, it is extracted but not stored into `__str`.

Definition at line 2447 of file `vstring.h`.

References `std::basic_ios<_CharT, _Traits>::widen()`.

4.11.4.92 `template<typename _Facet> bool std::has_facet (const locale & __loc) throw ()`

Test for the presence of a facet.

`has_facet` tests the `locale` argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Parameters

Facet The facet type to test the presence of.

locale The locale to test.

Returns

true if `locale` contains a facet of type `Facet`, else false.

Definition at line 91 of file `locale_classes.tcc`.

4.11.4.93 `ios_base& std::hex (ios_base & __base) [inline]`

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 940 of file `ios_base.h`.

References `std::ios_base::setf()`.

4.11.4.94 `template<typename _InputIterator1, typename _InputIterator2, typename _Tp> _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init) [inline]`

Compute inner product of two ranges.

Starting with an initial value of *init*, multiplies successive elements from the two ranges and adds each product into the accumulated value using `operator+()`. The values in the ranges are processed in order.

Parameters

first1 Start of range 1.

last1 End of range 1.

first2 Start of range 2.

init Starting value to add other values to.

Returns

The final inner product.

Definition at line 170 of file `stl_numeric.h`.

4.11.4.95 `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2> _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2) [inline]`

Compute inner product of two ranges.

Starting with an initial value of *init*, applies *binary_op2* to successive elements from the two ranges and accumulates each result into the accumulated value using *binary_op1*. The values in the ranges are processed in order.

Parameters

first1 Start of range 1.

last1 End of range 1.

first2 Start of range 2.

init Starting value to add other values to.

binary_op1 Function object to accumulate with.

binary_op2 Function object to apply to pairs of input values.

Returns

The final inner product.

Definition at line 202 of file stl_numeric.h.

4.11.4.96 ios_base& std::internal (ios_base & __base) [inline]

Calls base.setf(ios_base::internal, ios_base::adjustfield).

Definition at line 907 of file ios_base.h.

References __gnu_debug::__base().

4.11.4.97 template<typename _ForwardIterator , typename _Tp > void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)

Create a range of sequentially increasing values.

For each element in the range [first,last) assigns value and increments value as if by ++value.

Parameters

first Start of range.

last End of range.

value Starting value.

Returns

Nothing.

Definition at line 81 of file stl_numeric.h.

4.11.4.98 template<typename _CharT > bool std::isalnum (_CharT __c, const locale & __loc) [inline]

Convenience interface to ctype.is(ctype_base::alnum, __c).

4.11.4.99 `template<typename _CharT > bool std::isalpha (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

4.11.4.100 `template<typename _CharT > bool std::isctrl (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

4.11.4.101 `template<typename _CharT > bool std::isdigit (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

4.11.4.102 `template<typename _CharT > bool std::isgraph (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

4.11.4.103 `template<typename _CharT > bool std::islower (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

4.11.4.104 `template<typename _CharT > bool std::isprint (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::print, __c)`.

4.11.4.105 `template<typename _CharT > bool std::ispunct (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

4.11.4.106 `template<typename _CharT > bool std::isspace (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::space, __c)`.

4.11.4.107 `template<typename _CharT > bool std::isupper (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

4.11.4.108 `template<typename _CharT > bool std::isxdigit (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

4.11.4.109 `ios_base& std::left (ios_base & __base) [inline]`

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 915 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `operator<<()`.

4.11.4.110 `template<class _T1 , class _T2 > pair<typename
__decay_and_strip<_T1>::__type, typename
__decay_and_strip<_T2>::__type> std::make_pair (_T1 && __x,
_T2 && __y) [inline]`

A convenience wrapper for creating a pair from two objects.

Parameters

x The first object.

y The second object.

Returns

A newly-constructed `pair<>` object of the appropriate type.

The standard requires that the objects be passed by reference-to-const, but LWG issue #181 says they should be passed by const value. We follow the LWG by default.

Definition at line 260 of file `stl_pair.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__parallel_merge_advance()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_parallel::__find_first_of_selector<_FIterator>::__M_sequential_algorithm()`, `__gnu_parallel::__adjacent_find_selector::__M_sequential_algorithm()`, `__gnu_parallel::__find_if_selector::__M_sequential_`

algorithm(), minmax_element(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::parallel_multiway_merge(), and __gnu_parallel::parallel_sort_mwms_pu().

4.11.4.111 ios_base& std::noboolalpha (ios_base & __base) [inline]

Calls base.unsetf(ios_base::boolalpha).

Definition at line 802 of file ios_base.h.

References boolalpha(), and std::ios_base::unsetf().

4.11.4.112 ios_base& std::noshowbase (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showbase).

Definition at line 818 of file ios_base.h.

References showbase(), and std::ios_base::unsetf().

4.11.4.113 ios_base& std::noshowpoint (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showpoint).

Definition at line 834 of file ios_base.h.

References showpoint(), and std::ios_base::unsetf().

4.11.4.114 ios_base& std::noshowpos (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showpos).

Definition at line 850 of file ios_base.h.

References showpos(), and std::ios_base::unsetf().

4.11.4.115 ios_base& std::noskipws (ios_base & __base) [inline]

Calls base.unsetf(ios_base::skipws).

Definition at line 866 of file ios_base.h.

References skipws(), and std::ios_base::unsetf().

4.11.4.116 ios_base& std::nounitbuf (ios_base & __base) [inline]

Calls base.unsetf(ios_base::unitbuf).

Definition at line 898 of file ios_base.h.

References `unitbuf()`, and `std::ios_base::unsetf()`.

4.11.4.117 `ios_base& std::nouppercase (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::uppercase)`.

Definition at line 882 of file ios_base.h.

References `std::ios_base::unsetf()`, and `uppercase()`.

4.11.4.118 `ios_base& std::oct (ios_base & __base) [inline]`

Calls `base.setf(ios_base::oct, ios_base::basefield)`.

Definition at line 948 of file ios_base.h.

References `std::ios_base::setf()`.

4.11.4.119 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test difference of two strings.

Parameters

lhs First string.

rhs Second string.

Returns

True if `lhs.compare(rhs) != 0`. False otherwise.

Definition at line 2421 of file basic_string.h.

4.11.4.120 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator!= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test difference of C string and string.

Parameters

lhs C string.

rhs String.

Returns

True if *rhs.compare(lhs) != 0*. False otherwise.

Definition at line 2433 of file `basic_string.h`.

4.11.4.121 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator!= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test difference of string and C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs.compare(rhs) != 0*. False otherwise.

Definition at line 2445 of file `basic_string.h`.

4.11.4.122 `template<typename _Tp , typename _Alloc > bool std::operator!= (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]`

Based on `operator==`.

Definition at line 1941 of file `stl_deque.h`.

4.11.4.123 `template<typename _Res , typename... _Args> bool std::operator!= (const function< _Res(_Args...)> & __f, nullptr_t) [inline]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`false` if the wrapper has no target, `true` otherwise

This function will not throw an exception.

Definition at line 2233 of file `functional`.

4.11.4.124 `template<typename _Res , typename... _Args> bool std::operator!=
(nullptr_t, const function< _Res(_Args...)> & __f) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2239 of file functional.

4.11.4.125 `template<typename _Tp > bool std::operator!= (const
_Fwd_list_iterator< _Tp > & __x, const _Fwd_list_const_iterator<
_Tp > & __y) [inline]`

Forward list iterator inequality comparison.

Definition at line 266 of file forward_list.h.

4.11.4.126 `template<typename _Tp , typename _Alloc > bool std::operator!= (
const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y)
[inline]`

Based on operator==.

Definition at line 1587 of file stl_list.h.

4.11.4.127 `template<typename _Tp , typename _Alloc > bool std::operator!= (
const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly) [inline]`

Based on operator==.

Definition at line 1260 of file forward_list.h.

4.11.4.128 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator!= (const map< _Key, _Tp,
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Based on operator==.

Definition at line 861 of file stl_map.h.

4.11.4.129 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator!= (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on `operator==`.

Definition at line 792 of file `stl_multimap.h`.

4.11.4.130 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator!= (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns `!(x == y)`.

Definition at line 687 of file `stl_multiset.h`.

4.11.4.131 `template<class _T1 , class _T2 > bool std::operator!= (const pair<
_T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Uses `operator==` to find the result.

Definition at line 207 of file `stl_pair.h`.

4.11.4.132 `template<typename _Tp , typename _Seq > bool std::operator!= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Based on `operator==`.

Definition at line 284 of file `stl_queue.h`.

4.11.4.133 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator!= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns `!(x == y)`.

Definition at line 700 of file `stl_set.h`.

4.11.4.134 `template<typename _Tp, typename _Seq > bool std::operator!= (
 const stack<_Tp, _Seq> & __x, const stack<_Tp, _Seq> & __y
) [inline]`

Based on operator==.

Definition at line 259 of file stl_stack.h.

4.11.4.135 `template<typename _Tp, typename _Alloc > bool std::operator!= (
 const vector<_Tp, _Alloc> & __x, const vector<_Tp, _Alloc> &
 __y) [inline]`

Based on operator==.

Definition at line 1295 of file stl_vector.h.

4.11.4.136 `template<size_t _Nb> bitset<_Nb> std::operator& (const bitset<
 _Nb> & __x, const bitset<_Nb> & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

x A bitset.

y A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1365 of file bitset.

4.11.4.137 `template<typename _CharT, typename _Traits, typename _Alloc
 > basic_string<_CharT, _Traits, _Alloc> std::operator+ (
 const basic_string<_CharT, _Traits, _Alloc> & __lhs, const
 basic_string<_CharT, _Traits, _Alloc> & __rhs)`

Concatenate two strings.

Parameters

lhs First string.

rhs Last string.

Returns

New string with value of *lhs* followed by *rhs*.

Definition at line 2304 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::append().

4.11.4.138 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > std::operator+ (const
_CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs)`

Concatenate C string and string.

Parameters

lhs First string.

rhs Last string.

Returns

New string with value of *lhs* followed by *rhs*.

Definition at line 692 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

4.11.4.139 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT
__lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`

Concatenate character and string.

Parameters

lhs First string.

rhs Last string.

Returns

New string with *lhs* followed by *rhs*.

Definition at line 708 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

4.11.4.140 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> std::operator+ (const
basic_string<_CharT, _Traits, _Alloc > & __lhs, const _CharT *
__rhs) [inline]`

Concatenate string and C string.

Parameters

lhs First string.

rhs Last string.

Returns

New string with *lhs* followed by *rhs*.

Definition at line 2341 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc >::append()`.

4.11.4.141 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> std::operator+ (const
basic_string<_CharT, _Traits, _Alloc > & __lhs, _CharT __rhs)
[inline]`

Concatenate string and character.

Parameters

lhs First string.

rhs Last string.

Returns

New string with *lhs* followed by *rhs*.

Definition at line 2357 of file `basic_string.h`.

4.11.4.142 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator< (const basic_string<_CharT, _Traits, _Alloc
> & __lhs, const basic_string<_CharT, _Traits, _Alloc > & __rhs
) [inline]`

Test if string precedes string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2458 of file basic_string.h.

```
4.11.4.143  template<typename _CharT , typename _Traits , typename _Alloc  
            > bool std::operator< ( const basic_string< _CharT, _Traits, _Alloc  
            > & __lhs, const _CharT * __rhs ) [inline]
```

Test if string precedes C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2470 of file basic_string.h.

```
4.11.4.144  template<typename _CharT , typename _Traits , typename _Alloc  
            > bool std::operator< ( const _CharT * __lhs, const basic_string<  
            _CharT, _Traits, _Alloc > & __rhs ) [inline]
```

Test if C string precedes string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* precedes *rhs*. False otherwise.

Definition at line 2482 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.145 `template<typename _Tp, typename _Alloc> bool std::operator< (const deque< _Tp, _Alloc> & __x, const deque< _Tp, _Alloc> & __y) [inline]`

Deque ordering relation.

Parameters

x A deque.

y A deque of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1933 of file `stl_deque.h`.

References `lexicographical_compare()`.

4.11.4.146 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc> & __x, const map< _Key, _Tp, _Compare, _Alloc> & __y) [inline]`

Map ordering relation.

Parameters

x A map.

y A map of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 854 of file `stl_map.h`.

4.11.4.147 `template<typename _Tp, typename _Alloc > bool std::operator< (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]`

Forward list ordering relation.

Parameters

lx A forward_list.

ly A forward_list of the same type as *lx*.

Returns

True iff *lx* is lexicographically less than *ly*.

This is a total ordering relation. It is linear in the size of the forward lists. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 1252 of file forward_list.h.

References lexicographical_compare().

4.11.4.148 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Multimap ordering relation.

Parameters

x A multimap.

y A multimap of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 785 of file stl_multimap.h.

4.11.4.149 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator< (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Multiset ordering relation.

Parameters

- x* A multiset.
- y* A multiset of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 680 of file `stl_multiset.h`.

4.11.4.150 `template<class _T1 , class _T2 > bool std::operator< (const pair<
_T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

<<http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>>

Definition at line 200 of file `stl_pair.h`.

4.11.4.151 `template<typename _Tp , typename _Seq > bool std::operator< (
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &
__y) [inline]`

Queue ordering relation.

Parameters

- x* A queue.
- y* A queue of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 278 of file `stl_queue.h`.

```
4.11.4.152  template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator< ( const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y ) [inline]
```

Set ordering relation.

Parameters

x A set.

y A set of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 693 of file `stl_set.h`.

```
4.11.4.153  template<typename _Tp , typename _Seq > bool std::operator< (
const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
) [inline]
```

Stack ordering relation.

Parameters

x A stack.

y A stack of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 253 of file stl_stack.h.

```
4.11.4.154  template<typename _Tp , typename _Alloc > bool std::operator< (  
            const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y )  
            [inline]
```

List ordering relation.

Parameters

x A list.

y A list of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 1580 of file stl_list.h.

References lexicographical_compare().

```
4.11.4.155  template<typename _Tp , typename _Alloc > bool std::operator< (  
            const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > &  
            __y ) [inline]
```

Vector ordering relation.

Parameters

x A vector.

y A vector of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 1288 of file stl_vector.h.

References lexicographical_compare().

4.11.4.156 `template<typename _CharT , typename _Traits , typename
_Alloc , template< typename, typename, typename > class
_Base> basic_ostream<_CharT, _Traits>& std::operator<<
(basic_ostream< _CharT, _Traits > & __os, const
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &
__str) [inline]`

Write string to a stream.

Parameters

`__os` Output stream.

`__str` String to write out.

Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2401 of file `vstring.h`.

4.11.4.157 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<(basic_ostream< char, _Traits > & __out, const
char * __s) [inline]`

String inserters.

Parameters

`out` An output stream.

`s` A character string.

Returns

`out`

Precondition

`s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at `s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 508 of file `ostream`.

4.11.4.158 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<< (basic_ostream< char, _Traits > & __out, char
__c) [inline]`

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 460 of file `ostream`.

4.11.4.159 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<< (basic_ostream< char, _Traits > & __out, const
signed char * __s) [inline]`

String inserters.

Parameters

out An output stream.

s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 521 of file `ostream`.

4.11.4.160 `template<class _Traits > basic_ostream<char, _Traits>&
std::operator<<(basic_ostream< char, _Traits > & __out, const
unsigned char * __s) [inline]`

String inserters.

Parameters

out An output stream.

s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 526 of file ostream.

4.11.4.161 `template<class _CharT , class _Traits , size_t _Nb>
std::basic_ostream<_CharT, _Traits>& std::operator<<(
std::basic_ostream< _CharT, _Traits > & __os, const bitset< _Nb
> & __x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1470 of file bitset.

References `std::__ctype_abstract_base< _CharT >::widen()`.

4.11.4.162 `template<typename _CharT , typename _Traits , typename
_Alloc > basic_ostream<_CharT, _Traits>& std::operator<<(
basic_ostream< _CharT, _Traits > & __os, const basic_string<
_CharT, _Traits, _Alloc > & __str) [inline]`

Write string to a stream.

Parameters

os Output stream.
str String to write out.

Returns

Reference to the output stream.

Output characters of *str* into *os* following the same rules as for writing a C string.

Definition at line 2641 of file `basic_string.h`.

4.11.4.163 `template<class _Traits > basic_ostream<char, _Traits>&
 std::operator<< (basic_ostream< char, _Traits > & __out,
 unsigned char __c) [inline]`

Character inserters.

Parameters

out An output stream.
c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 471 of file `ostream`.

4.11.4.164 `template<typename _CharT , typename _Traits >
 basic_ostream<_CharT, _Traits>& std::operator<< (
 basic_ostream< _CharT, _Traits > & __out, _CharT __c)
 [inline]`

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). *out.width(0)* is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 449 of file `ostream`.

```
4.11.4.165  template<typename _CharT , typename _Traits >
              basic_ostream<_CharT, _Traits>& std::operator<< (
              basic_ostream< _CharT, _Traits > & __out, char __c )
              [inline]
```

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). *out.width(0)* is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 454 of file `ostream`.

```
4.11.4.166  template<typename _CharT , typename _Traits , typename
              _Tp > basic_ostream<_CharT, _Traits>& std::operator<< (
              basic_ostream< _CharT, _Traits > && __os, const _Tp & __x )
              [inline]
```

Generic inserter for rvalue stream.

Parameters

- os* An input stream.
- x* A reference to the object being inserted.

Returns

os

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

Definition at line 579 of file ostream.

```
4.11.4.167  template<typename _CharT , typename _Traits > basic_ostream<
              _CharT, _Traits > & std::operator<< ( basic_ostream< _CharT,
              _Traits > & __out, const char * __s )
```

String inserters.

Parameters

- out* An output stream.
- s* A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 321 of file ostream.tcc.

References `std::ios_base::badbit`.

```
4.11.4.168  template<typename _CharT , typename _Traits >
              basic_ostream<_CharT, _Traits>& std::operator<< (
              basic_ostream< _CharT, _Traits > & __out, const _CharT * __s )
              [inline]
```

String inserters.

Parameters

out An output stream.

s A character string.

Returns

out

Precondition

s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts `traits::length(s)` characters starting at *s*, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

Definition at line 491 of file ostream.

4.11.4.169 `template<class _Traits> basic_ostream<char, _Traits>&
std::operator<<(basic_ostream< char, _Traits> & __out, signed
char __c) [inline]`

Character inserters.

Parameters

out An output stream.

c A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in [std::basic_ostream](#). After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `out.width(0)` is then called.

If *c* is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 466 of file ostream.

4.11.4.170 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator<= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs) [inline]`

Test if string doesn't follow string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2532 of file basic_string.h.

4.11.4.171 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator<= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test if string doesn't follow C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2544 of file basic_string.h.

4.11.4.172 `template<typename _CharT , typename _Traits , typename
_Alloc > bool std::operator<= (const _CharT * __lhs, const
basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if C string doesn't follow string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* doesn't follow *rhs*. False otherwise.

Definition at line 2556 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.173 `template<typename _Tp, typename _Alloc> bool std::operator<=`
`(const deque<_Tp, _Alloc> & __x, const deque<_Tp, _Alloc>`
`& __y) [inline]`

Based on `operator<`.

Definition at line 1955 of file `stl_deque.h`.

4.11.4.174 `template<typename _Tp, typename _Alloc> bool std::operator<=`
`(const list<_Tp, _Alloc> & __x, const list<_Tp, _Alloc> & __y`
`) [inline]`

Based on `operator<`.

Definition at line 1599 of file `stl_list.h`.

4.11.4.175 `template<typename _Key, typename _Tp, typename _Compare,`
`typename _Alloc> bool std::operator<= (const map<_Key, _Tp,`
`_Compare, _Alloc> & __x, const map<_Key, _Tp, _Compare,`
`_Alloc> & __y) [inline]`

Based on `operator<`.

Definition at line 875 of file `stl_map.h`.

4.11.4.176 `template<typename _Tp, typename _Alloc> bool std::operator<=`
`(const forward_list<_Tp, _Alloc> & __lx, const forward_list<`
`_Tp, _Alloc> & __ly) [inline]`

Based on `operator<`.

Definition at line 1281 of file `forward_list.h`.

4.11.4.177 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator<= (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on operator<.

Definition at line 806 of file stl_multimap.h.

4.11.4.178 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator<= (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns !(y < x).

Definition at line 701 of file stl_multiset.h.

4.11.4.179 `template<typename _Tp , typename _Seq > bool std::operator<= (
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &
__y) [inline]`

Based on operator<.

Definition at line 296 of file stl_queue.h.

4.11.4.180 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator<= (const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns !(y < x).

Definition at line 714 of file stl_set.h.

4.11.4.181 `template<typename _Tp , typename _Seq > bool std::operator<= (
const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
) [inline]`

Based on operator<.

Definition at line 271 of file stl_stack.h.

4.11.4.182 `template<class _T1, class _T2 > bool std::operator<= (const
pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)
[inline]`

Uses `operator<` to find the result.

Definition at line 219 of file `stl_pair.h`.

4.11.4.183 `template<typename _Tp, typename _Alloc > bool std::operator<=
(const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc >
& __y) [inline]`

Based on `operator<`.

Definition at line 1307 of file `stl_vector.h`.

4.11.4.184 `template<typename _CharT, typename _Traits, typename _Alloc
> bool std::operator==(const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > &
__rhs) [inline]`

Test equivalence of two strings.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs.compare(rhs) == 0*. False otherwise.

Definition at line 2375 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.185 `template<typename _Tp, typename _Alloc > bool std::operator==(
const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > &
__y) [inline]`

Deque equality comparison.

Parameters

x A deque.

y A deque of the same type as *x*.

Returns

True iff the size and elements of the dequeues are equal.

This is an equivalence relation. It is linear in the size of the dequeues. Dequeues are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1915 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, `equal()`, and `std::deque< _Tp, _Alloc >::size()`.

4.11.4.186 `template<typename _Res , typename... _Args> bool
std::operator==(const function< _Res(_Args...)> & __f, nullptr_t
) [inline]`

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`true` if the wrapper has no target, `false` otherwise

This function will not throw an exception.

Definition at line 2215 of file `functional`.

4.11.4.187 `template<typename _Tp , typename _Alloc > bool std::operator==(
const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly)`

Forward list equality comparison.

Parameters

lx A `forward_list`

ly A `forward_list` of the same type as *lx*.

Returns

True iff the size and elements of the forward lists are equal.

This is an equivalence relation. It is linear in the size of the forward lists. Dequeues are considered equivalent if corresponding elements compare equal.

Definition at line 374 of file `forward_list.tcc`.

References `std::forward_list< _Tp, _Alloc >::cbegin()`, and `std::forward_list< _Tp, _Alloc >::cend()`.

4.11.4.188 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator==(const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Multiset equality comparison.

Parameters

x A multiset.

y A multiset of the same type as *x*.

Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 663 of file `std_multiset.h`.

4.11.4.189 `template<class _T1 , class _T2 > bool std::operator==(const pair<
_T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Two pairs of the same type are equal iff their members are equal.

Definition at line 194 of file `std_pair.h`.

References `std::pair< _T1, _T2 >::first`, and `std::pair< _T1, _T2 >::second`.

4.11.4.190 `template<typename _Tp , typename _Seq > bool std::operator==(
const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
) [inline]`

Stack equality comparison.

Parameters

x A stack.

y A stack of the same type as *x*.

Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 235 of file `stl_stack.h`.

4.11.4.191 `template<typename _Tp, typename _Seq > bool std::operator==(const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Queue equality comparison.

Parameters

x A queue.

y A queue of the same type as *x*.

Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 260 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

4.11.4.192 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator==(const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test equivalence of C string and string.

Parameters

lhs C string.

rhs String.

Returns

True if *rhs.compare(lhs) == 0*. False otherwise.

Definition at line 2396 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.193 `template<typename _StateT > bool std::operator==(const fpos< _StateT > & __lhs, const fpos< _StateT > & __rhs) [inline]`

Test if equivalent to another position.

Definition at line 216 of file postypes.h.

4.11.4.194 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator==(const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test equivalence of string and C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs.compare(rhs) == 0*. False otherwise.

Definition at line 2408 of file basic_string.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.195 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator==(const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Set equality comparison.

Parameters

x A set.

y A set of the same type as *x*.

Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 676 of file stl_set.h.

4.11.4.196 `template<typename _Tp, typename _Alloc> bool std::operator==(const vector<_Tp, _Alloc> & __x, const vector<_Tp, _Alloc> & __y) [inline]`

Vector equality comparison.

Parameters

- x* A vector.
- y* A vector of the same type as *x*.

Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1271 of file `std_vector.h`.

References `std::vector<_Tp, _Alloc>::begin()`, `std::vector<_Tp, _Alloc>::end()`, `equal()`, and `std::vector<_Tp, _Alloc>::size()`.

4.11.4.197 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator==(const multimap<_Key, _Tp, _Compare, _Alloc> & __x, const multimap<_Key, _Tp, _Compare, _Alloc> & __y) [inline]`

Multimap equality comparison.

Parameters

- x* A multimap.
- y* A multimap of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 768 of file `std_multimap.h`.

4.11.4.198 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator==(const map< _Key, _Tp,
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Map equality comparison.

Parameters

- x* A map.
- y* A map of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 837 of file `stl_map.h`.

4.11.4.199 `template<typename _Res , typename... _Args> bool
std::operator==(nullptr_t, const function< _Res(_Args...)> &
__f) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2221 of file `functional`.

4.11.4.200 `template<typename _Tp > bool std::operator==(const
_Fwd_list_iterator< _Tp > & __x, const _Fwd_list_const_iterator<
_Tp > & __y) [inline]`

Forward list iterator equality comparison.

Definition at line 257 of file `forward_list.h`.

4.11.4.201 `template<typename _Tp , typename _Alloc > bool std::operator==(
const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y)
[inline]`

List equality comparison.

Parameters

- x* A list.

y A list of the same type as *x*.

Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1551 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

4.11.4.202 `template<typename _Tp, typename _Alloc> bool std::operator> (const vector<_Tp, _Alloc> & __x, const vector<_Tp, _Alloc> & __y) [inline]`

Based on `operator<`.

Definition at line 1301 of file `stl_vector.h`.

4.11.4.203 `template<typename _Tp, typename _Alloc> bool std::operator> (const list<_Tp, _Alloc> & __x, const list<_Tp, _Alloc> & __y) [inline]`

Based on `operator<`.

Definition at line 1593 of file `stl_list.h`.

4.11.4.204 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator> (const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string follows string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2519 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

4.11.4.205 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator> (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns $y < x$.

Definition at line 694 of file `stl_multiset.h`.

4.11.4.206 `template<typename _Tp , typename _Seq > bool std::operator> (
const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > &
__y) [inline]`

Based on `operator<`.

Definition at line 290 of file `stl_queue.h`.

4.11.4.207 `template<typename _Tp , typename _Alloc > bool std::operator> (
const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly) [inline]`

Based on `operator<`.

Definition at line 1267 of file `forward_list.h`.

4.11.4.208 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator> (const set< _Key, _Compare, _Alloc > &
__x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns $y < x$.

Definition at line 707 of file `stl_set.h`.

4.11.4.209 `template<typename _Tp , typename _Seq > bool std::operator> (
const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y
) [inline]`

Based on `operator<`.

Definition at line 265 of file `stl_stack.h`.

4.11.4.210 `template<typename _Tp, typename _Alloc > bool std::operator> (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]`

Based on operator<.

Definition at line 1948 of file stl_deque.h.

4.11.4.211 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT* __rhs) [inline]`

Test if string follows C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2507 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.212 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if string follows string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* follows *rhs*. False otherwise.

Definition at line 2495 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.213 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator> (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 799 of file `stl_multimap.h`.

4.11.4.214 `template<class _T1 , class _T2 > bool std::operator> (const pair<
_T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]`

Uses `operator<` to find the result.

Definition at line 213 of file `stl_pair.h`.

4.11.4.215 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator> (const map< _Key, _Tp,
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 868 of file `stl_map.h`.

4.11.4.216 `template<typename _Tp , typename _Alloc > bool std::operator>=
(const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y
) [inline]`

Based on `operator<`.

Definition at line 1605 of file `stl_list.h`.

4.11.4.217 `template<typename _Tp , typename _Alloc > bool std::operator>=
(const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc >
& __y) [inline]`

Based on `operator<`.

Definition at line 1962 of file `stl_deque.h`.

4.11.4.218 `template<typename _CharT , typename _Traits , typename _Alloc
> bool std::operator>= (const basic_string< _CharT, _Traits,
_Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test if string doesn't precede C string.

Parameters

lhs String.

rhs C string.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2581 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`.

4.11.4.219 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator>= (const multimap< _Key,
_Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp,
_Compare, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 813 of file `stl_multimap.h`.

4.11.4.220 `template<class _T1 , class _T2 > bool std::operator>= (const
pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)
[inline]`

Uses `operator<` to find the result.

Definition at line 225 of file `stl_pair.h`.

4.11.4.221 `template<typename _Tp , typename _Alloc > bool std::operator>=
(const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc >
& __y) [inline]`

Based on `operator<`.

Definition at line 1313 of file `stl_vector.h`.

4.11.4.222 `template<typename _Tp , typename _Seq > bool std::operator>= (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y) [inline]`

Based on operator<.

Definition at line 277 of file stl_stack.h.

4.11.4.223 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator>= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns !(x < y).

Definition at line 721 of file stl_set.h.

4.11.4.224 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if string doesn't precede string.

Parameters

lhs First string.

rhs Second string.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2569 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.225 `template<typename _Tp , typename _Seq > bool std::operator>= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Based on operator<.

Definition at line 302 of file stl_queue.h.

4.11.4.226 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > bool std::operator>= (const map< _Key, _Tp,
_Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

Based on operator<.

Definition at line 882 of file stl_map.h.

4.11.4.227 `template<typename _Key , typename _Compare , typename _Alloc
> bool std::operator>= (const multiset< _Key, _Compare, _Alloc
> & __x, const multiset< _Key, _Compare, _Alloc > & __y)
[inline]`

Returns !(x < y).

Definition at line 708 of file stl_multiset.h.

4.11.4.228 `template<typename _CharT , typename _Traits , typename
_Alloc > bool std::operator>= (const _CharT * __lhs, const
basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if C string doesn't precede string.

Parameters

lhs C string.

rhs String.

Returns

True if *lhs* doesn't precede *rhs*. False otherwise.

Definition at line 2593 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

4.11.4.229 `template<typename _Tp , typename _Alloc > bool std::operator>=
(const forward_list< _Tp, _Alloc > & __lx, const forward_list<
_Tp, _Alloc > & __ly) [inline]`

Based on operator<.

Definition at line 1274 of file forward_list.h.

4.11.4.230 `template<class _Traits > basic_istream<char, _Traits>&
std::operator>> (basic_istream< char, _Traits > & __in, signed
char & __c) [inline]`

Character extractors.

Parameters

- in* An input stream.
- c* A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 709 of file istream.

4.11.4.231 `template<typename _CharT , typename _Traits , typename
_Tp > basic_istream<_CharT, _Traits>& std::operator>> (
basic_istream< _CharT, _Traits > && __is, _Tp & __x)
[inline]`

Generic extractor for rvalue stream.

Parameters

- is* An input stream.
- x* A reference to the extraction target.

Returns

is

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

Definition at line 847 of file istream.

4.11.4.232 `template<class _CharT , class _Traits , size_t _Nb>
std::basic_istream<_CharT, _Traits>& std::operator>> (
std::basic_istream< _CharT, _Traits > & __is, bitset<_Nb > &
__x)`

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1402 of file bitset.

References `std::basic_string<_CharT, _Traits, _Alloc >::empty()`, `std::basic_string<_CharT, _Traits, _Alloc >::push_back()`, `std::basic_ios<_CharT, _Traits >::rdbuf()`, `std::basic_string<_CharT, _Traits, _Alloc >::reserve()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_ios<_CharT, _Traits >::widen()`.

4.11.4.233 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits> & std::operator>> (basic_istream<_CharT, _Traits> & __in, _CharT * __s)`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is *the number of elements of the largest array of* *
- *char_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 935 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::getloc()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::ios_base::width()`.

4.11.4.234 `template<> basic_istream<char>& std::operator>> (basic_istream<char> & __in, char * __s)`

Character string extractors.

Parameters

in An input stream.

s A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of *
- *char_type* that can store a terminating `eos`.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

```

4.11.4.235  template<typename _CharT , typename _Traits , typename
              _Alloc , template< typename, typename, typename
              > class _Base> basic_istream< _CharT, _Traits > &
              std::operator>> ( basic_istream< _CharT, _Traits > & __is,
              __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &
              __str )

```

Read stream into a string.

Parameters

`__is` Input stream.
`__str` Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 547 of file `vstring.tcc`.

```

4.11.4.236  template<typename _CharT , typename _Traits , typename _Alloc
              > basic_istream< _CharT, _Traits > & std::operator>> (
              basic_istream< _CharT, _Traits > & __is, basic_string< _CharT,
              _Traits, _Alloc > & __str )

```

Read stream into a string.

Parameters

`is` Input stream.
`str` Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `is` into `str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `str`. Any previous contents of `str` are erased.

Definition at line 996 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::erase()`, `std::ios_base::getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::max_size()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::ios_base::width()`.

4.11.4.237 `template<class _Traits > basic_istream<char, _Traits>&
std::operator>> (basic_istream< char, _Traits > & __in,
unsigned char & __c) [inline]`

Character extractors.

Parameters

- in* An input stream.
- c* A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 704 of file istream.

4.11.4.238 `template<class _Traits > basic_istream<char, _Traits>&
std::operator>> (basic_istream< char, _Traits > & __in,
unsigned char * __s) [inline]`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of *
- *char_type* that can store a terminating *eos*.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 751 of file `istream`.

4.11.4.239 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::operator>> (basic_istream<_CharT, _Traits> & __in, _CharT & __c)`

Character extractors.

Parameters

in An input stream.

c A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *c*. Otherwise, sets failbit in the input stream.

Definition at line 903 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

4.11.4.240 `template<class _Traits> basic_istream<char, _Traits> & std::operator>> (basic_istream<char, _Traits> & __in, signed char * __s) [inline]`

Character string extractors.

Parameters

- in* An input stream.
- s* A pointer to a character array.

Returns

in

Behaves like one of the formatted arithmetic extractors described in [std::basic_istream](#). After constructing a sentry object with good status, this function extracts up to *n* characters and stores them into the array starting at *s*. *n* is defined as:

- if `width()` is greater than zero, *n* is `width()` otherwise
- *n* is the number of elements of the largest array of *
- *char_type* that can store a terminating `eos`.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 756 of file `istream`.

4.11.4.241 `template<size_t _Nb> bitset<_Nb> std::operator^ (const bitset<_Nb> & __x, const bitset<_Nb> & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

- x* A bitset.
- y* A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1383 of file `bitset`.

4.11.4.242 `template<size_t _Nb> bitset<_Nb> std::operator| (const bitset<_Nb> & __x, const bitset<_Nb> & __y) [inline]`

Global bitwise operations on bitsets.

Parameters

x A bitset.

y A bitset of the same size as *x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1374 of file `bitset`.

4.11.4.243 `template<typename _InputIterator , typename _OutputIterator
> _OutputIterator std::partial_sum (_InputIterator __first,
_InputIterator __last, _OutputIterator __result)`

Return list of partial sums.

Accumulates the values in the range `[first,last)` using `operator+()`. As each successive input value is added into the total, that partial sum is written to *result*. Therefore, the first value in *result* is the first value of the input, the second value in *result* is the sum of the first and second input values, and so on.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to *result*.

Definition at line 233 of file `stl_numeric.h`.

4.11.4.244 `template<typename _InputIterator, typename _OutputIterator,
typename _BinaryOperation > _OutputIterator std::partial_sum (
_InputIterator __first, _InputIterator __last, _OutputIterator
__result, _BinaryOperation __binary_op)`

Return list of partial sums.

Accumulates the values in the range [first,last) using operator+(). As each successive input value is added into the total, that partial sum is written to *result*. Therefore, the first value in result is the first value of the input, the second value in result is the sum of the first and second input values, and so on.

Parameters

first Start of input range.

last End of input range.

result Output to write sums to.

Returns

Iterator pointing just beyond the values written to result.

Definition at line 273 of file `stl_numeric.h`.

4.11.4.245 `template<typename _MoneyT > _Put_money<_MoneyT>
std::put_money (const _MoneyT & __mon, bool __intl = false)
[inline]`

Extended manipulator for inserting money.

Parameters

mon Either long double or a specialization of `basic_string`.

intl A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts *mon*.

Definition at line 292 of file `iomanip`.

4.11.4.246 `template<typename _Tp > reference_wrapper<_Tp> std::ref (
_Tp & __t) [inline]`

Denotes a reference should be taken to a variable.

Definition at line 468 of file `functional`.

Referenced by `ref()`.

4.11.4.247 `template<typename _Tp > reference_wrapper<_Tp> std::ref (`
`reference_wrapper<_Tp > __t) [inline]`

Partial specialization.

Definition at line 480 of file functional.

References `ref()`.

4.11.4.248 `template<typename _InputIterator , typename _OutputIterator`
`, typename _Tp > _OutputIterator std::replace_copy (`
`_InputIterator __first, _InputIterator __last, _OutputIterator`
`__result, const _Tp & __old_value, const _Tp & __new_value)`

Copy a sequence, replacing each element of one value with another value.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

old_value The value to be replaced.

new_value The replacement value.

Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range `[first,last)` to the output range `[result,result+(last-first))` replacing elements equal to `old_value` with `new_value`.

Definition at line 3741 of file `stl_algo.h`.

4.11.4.249 `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
`[inline]`

Manipulator for `setf`.

Parameters

mask A format flags mask.

Sent to a stream object, this manipulator resets the specified flags, via `stream.setf(0,mask)`.

Definition at line 63 of file `iomanip`.

4.11.4.250 `template<typename _Tp > void std::return_temporary_buffer (`
`_Tp * __p) [inline]`

The companion to [get_temporary_buffer\(\)](#).

Parameters

p A buffer previously allocated by `get_temporary_buffer`.

Returns

None.

Frees the memory pointed to by *p*.

Definition at line 111 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer()`.

4.11.4.251 `ios_base& std::right (ios_base & __base) [inline]`

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 923 of file `ios_base.h`.

References `std::ios_base::setf()`.

4.11.4.252 `ios_base& std::scientific (ios_base & __base) [inline]`

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

Definition at line 965 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `operator<<()`.

4.11.4.253 `new_handler std::set_new_handler (new_handler) throw ()`

Takes a replacement handler as the argument, returns the previous handler.

4.11.4.254 `_Setbase std::setbase (int __base) [inline]`

Manipulator for `setf`.

Parameters

base A numeric base.

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when *base* is 8, 10, or 16, accordingly, and to 0 if *base* is any other value.

Definition at line 124 of file `iomanip`.

4.11.4.255 `template<typename _CharT> _Setfill<_CharT> std::setfill (`
`_CharT __c) [inline]`

Manipulator for `fill`.

Parameters

c The new fill character.

Sent to a stream object, this manipulator calls `fill(c)` for that object.

Definition at line 162 of file `iomanip`.

4.11.4.256 `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
`[inline]`

Manipulator for `setf`.

Parameters

mask A format flags mask.

Sent to a stream object, this manipulator sets the format flags to *mask*.

Definition at line 93 of file `iomanip`.

4.11.4.257 `_Setprecision std::setprecision (int __n) [inline]`

Manipulator for `precision`.

Parameters

n The new precision.

Sent to a stream object, this manipulator calls `precision(n)` for that object.

Definition at line 192 of file `iomanip`.

4.11.4.258 `_Setw std::setw (int __n) [inline]`

Manipulator for `width`.

Parameters

n The new width.

Sent to a stream object, this manipulator calls `width(n)` for that object.

Definition at line 222 of file `iomanip`.

4.11.4.259 `ios_base& std::showbase (ios_base & __base) [inline]`

Calls `base.setf(ios_base::showbase)`.

Definition at line 810 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noshowbase()`.

4.11.4.260 `ios_base& std::showpoint (ios_base & __base) [inline]`

Calls `base.setf(ios_base::showpoint)`.

Definition at line 826 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noshowpoint()`.

4.11.4.261 `ios_base& std::showpos (ios_base & __base) [inline]`

Calls `base.setf(ios_base::showpos)`.

Definition at line 842 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noshowpos()`.

4.11.4.262 `ios_base& std::skipws (ios_base & __base) [inline]`

Calls `base.setf(ios_base::skipws)`.

Definition at line 858 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `noskipws()`, and operator `>>()`.

4.11.4.263 `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
 __shared_ptr<_Tp, _Lp> std::static_pointer_cast (const
 __shared_ptr<_Tp1, _Lp> & __r) [inline]`

`static_pointer_cast`

Definition at line 936 of file `shared_ptr_base.h`.

4.11.4.264 `template<typename _Key, typename _Compare, typename _Alloc
 > void std::swap (multiset< _Key, _Compare, _Alloc> & __x,
 multiset< _Key, _Compare, _Alloc> & __y) [inline]`

See [std::multiset::swap\(\)](#).

Definition at line 715 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc>::swap()`.

4.11.4.265 `template<typename _CharT, typename _Traits, typename _Alloc>
 void std::swap (basic_string< _CharT, _Traits, _Alloc> & __lhs,
 basic_string< _CharT, _Traits, _Alloc> & __rhs) [inline]`

Swap contents of two strings.

Parameters

lhs First string.

rhs Second string.

Exchanges the contents of *lhs* and *rhs* in constant time.

Definition at line 2606 of file `basic_string.h`.

References `std::basic_string< _CharT, _Traits, _Alloc>::swap()`.

4.11.4.266 `template<typename _Res, typename... _Args> void std::swap (
 function< _Res(_Args...)> & __x, function< _Res(_Args...)> &
 __y) [inline]`

Swap the targets of two polymorphic function object wrappers.

This function will not throw an exception.

Definition at line 2251 of file `functional`.

Referenced by `__gnu_parallel::LoserTree< false, _Tp, _Compare>::__delete_min_insert()`, `__gnu_parallel::LoserTree< __stable, _Tp, _Compare>::__delete_min_insert()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_`

partition(), `__gnu_parallel::__qsb_divide()`, `__gnu_parallel::__qsb_local_sort_` -
`with_helping()`, `__rotate()`, `std::regex_traits< _Ch_type >::imbue()`, `std::vector<`
`std::sub_match< _Bi_iter >, _Allocator >::swap()`, and `std::function< _Res(_`
`ArgTypes...)>::swap()`.

4.11.4.267 `template<typename _Tp , typename _Alloc > void std::swap (`
`forward_list< _Tp, _Alloc > & __lx, forward_list< _Tp, _Alloc >`
`& __ly) [inline]`

See [std::forward_list::swap\(\)](#).

Definition at line 1288 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::swap()`.

4.11.4.268 `template<typename _Tp , typename _Alloc > void std::swap (list<`
`_Tp, _Alloc > & __x, list< _Tp, _Alloc > & __y) [inline]`

See [std::list::swap\(\)](#).

Definition at line 1611 of file `stl_list.h`.

References `std::list< _Tp, _Alloc >::swap()`.

4.11.4.269 `template<typename _Tp , typename _Alloc > void std::swap (`
`vector< _Tp, _Alloc > & __x, vector< _Tp, _Alloc > & __y)`
`[inline]`

See [std::vector::swap\(\)](#).

Definition at line 1319 of file `stl_vector.h`.

References `std::vector< _Tp, _Alloc >::swap()`.

4.11.4.270 `template<typename _Tp , typename _Alloc > void std::swap (`
`deque< _Tp, _Alloc > & __x, deque< _Tp, _Alloc > & __y)`
`[inline]`

See [std::deque::swap\(\)](#).

Definition at line 1969 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::swap()`.

4.11.4.271 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > void std::swap (multimap< _Key, _Tp,
_Compare, _Alloc > & __x, multimap< _Key, _Tp, _Compare,
_Alloc > & __y) [inline]`

See [std::multimap::swap\(\)](#).

Definition at line 820 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::swap()`.

4.11.4.272 `template<typename _Key , typename _Compare , typename _Alloc
> void std::swap (set< _Key, _Compare, _Alloc > & __x, set<
_Key, _Compare, _Alloc > & __y) [inline]`

See [std::set::swap\(\)](#).

Definition at line 728 of file `stl_set.h`.

References `std::set< _Key, _Compare, _Alloc >::swap()`.

4.11.4.273 `template<typename _Key , typename _Tp , typename _Compare ,
typename _Alloc > void std::swap (map< _Key, _Tp, _Compare,
_Alloc > & __x, map< _Key, _Tp, _Compare, _Alloc > & __y)
[inline]`

See [std::map::swap\(\)](#).

Definition at line 889 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::swap()`.

4.11.4.274 `template<class _T1 , class _T2 > void std::swap (pair< _T1, _T2 >
& __x, pair< _T1, _T2 > & __y) [inline]`

See `std::pair::swap()`.

Definition at line 234 of file `stl_pair.h`.

4.11.4.275 `template<typename _CharT > _CharT std::tolower (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to `ctype::tolower(__c)`.

Referenced by `std::regex_traits< _Ch_type >::translate_nocase()`.

4.11.4.276 `template<typename _CharT > _CharT std::toupper (_CharT __c,
const locale & __loc) [inline]`

Convenience interface to ctype.toupper(__c).

4.11.4.277 `template<typename _InputIterator , typename _ForwardIterator >
_ForwardIterator std::uninitialized_copy (_InputIterator __first,
_InputIterator __last, _ForwardIterator __result) [inline]`

Copies the range [first,last) into result.

Parameters

first An input iterator.

last An input iterator.

result An output iterator.

Returns

result + (first - last)

Like copy(), but does not require an initialized output range.

Definition at line 107 of file stl_uninitialized.h.

Referenced by __gnu_parallel::parallel_sort_mwms_pu().

4.11.4.278 `template<typename _InputIterator , typename _Size , typename
_ForwardIterator > _ForwardIterator std::uninitialized_copy_n (
_InputIterator __first, _Size __n, _ForwardIterator __result)
[inline]`

Copies the range [first,first+n) into result.

Parameters

first An input iterator.

n The number of elements to copy.

result An output iterator.

Returns

result + n

Like [copy_n\(\)](#), but does not require an initialized output range.

Definition at line 629 of file stl_uninitialized.h.

References [__iterator_category\(\)](#).

4.11.4.279 `template<typename _ForwardIterator , typename _Tp > void
std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator
__last, const _Tp & __x) [inline]`

Copies the value *x* into the range [*first*,*last*).

Parameters

first An input iterator.

last An input iterator.

x The source value.

Returns

Nothing.

Like `fill()`, but does not require an initialized output range.

Definition at line 164 of file `stl_uninitialized.h`.

4.11.4.280 `template<typename _ForwardIterator , typename _Size , typename
_Tp > void std::uninitialized_fill_n (_ForwardIterator __first,
_Size __n, const _Tp & __x) [inline]`

Copies the value *x* into the range [*first*,*first*+*n*).

Parameters

first An input iterator.

n The number of copies to make.

x The source value.

Returns

Nothing.

Like `fill_n()`, but does not require an initialized output range.

Definition at line 218 of file `stl_uninitialized.h`.

4.11.4.281 `ios_base& std::unitbuf (ios_base & __base) [inline]`

Calls `base.setf(ios_base::unitbuf)`.

Definition at line 890 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `nounitbuf()`.

4.11.4.282 `ios_base& std::uppercase (ios_base & __base) [inline]`

Calls `base.setf(ios_base::uppercase)`.

Definition at line 874 of file `ios_base.h`.

References `std::ios_base::setf()`.

Referenced by `nouppercase()`.

4.11.4.283 `template<typename _Facet > const _Facet & std::use_facet (const locale & __loc)`

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

Parameters

Facet The facet type to access.

locale The locale to use.

Returns

Reference to facet of type `Facet`.

Exceptions

`std::bad_cast` if locale doesn't contain a facet of type `Facet`.

Definition at line 105 of file `locale_classes.tcc`.

Referenced by `std::regex_traits< _Ch_type >::isctype()`, and `std::regex_traits< _Ch_type >::transform()`.

4.11.4.284 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > & __is)`

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass    mc;

std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling operator>> on cin and your object. Note that the same effect can be achieved by creating a [std::basic_istream::sentry](#) inside your definition of operator>>.

Definition at line 996 of file istream.tcc.

References [std::ios_base::eofbit](#), [std::ios_base::getloc\(\)](#), [std::basic_ios< _CharT, _Traits >::rdbuf\(\)](#), and [std::basic_ios< _CharT, _Traits >::setstate\(\)](#).

4.11.5 Variable Documentation

4.11.5.1 `enable_if<(is_pointer< _Functor >::value &&is_function< typename remove_pointer< _Functor >::type >::value), typename result_of< _Functor(_Args...)>::type >::type std::__invoke [inline]`

Invoke a function object, which may be either a member pointer or a function object. The first parameter will tell which.

Definition at line 238 of file functional.

4.11.5.2 `ostream std::cerr`

Linked to standard error (unbuffered).

4.11.5.3 `istream std::cin`

Linked to standard input.

4.11.5.4 `ostream std::clog`

Linked to standard error (buffered).

4.11.5.5 `ostream std::cout`

Linked to standard output.

4.11.5.6 `wostream std::wcerr`

Linked to standard error (unbuffered).

4.11.5.7 wistream std::wcin

Linked to standard input.

4.11.5.8 wostream std::wlog

Linked to standard error (buffered).

4.11.5.9 wostream std::wcout

Linked to standard output.

4.12 std::__debug Namespace Reference

GNU debug code, replaces standard behavior with debug behavior.

Classes

- class [bitset](#)
Class [std::bitset](#) with additional safety/checking/debug instrumentation.
- class [deque](#)
Class [std::deque](#) with safety/checking/debug instrumentation.
- class [list](#)
Class [std::list](#) with safety/checking/debug instrumentation.
- class [map](#)
Class [std::map](#) with safety/checking/debug instrumentation.
- class [multimap](#)
Class [std::multimap](#) with safety/checking/debug instrumentation.
- class [multiset](#)
Class [std::multiset](#) with safety/checking/debug instrumentation.
- class [set](#)
Class [std::set](#) with safety/checking/debug instrumentation.
- class [unordered_map](#)

Class [`std::unordered_map`](#) with safety/checking/debug instrumentation.

- class [`unordered_multimap`](#)

Class [`std::unordered_multimap`](#) with safety/checking/debug instrumentation.

- class [`unordered_multiset`](#)

Class [`std::unordered_multiset`](#) with safety/checking/debug instrumentation.

- class [`unordered_set`](#)

Class [`std::unordered_set`](#) with safety/checking/debug instrumentation.

- class [`vector`](#)

Class [`std::vector`](#) with safety/checking/debug instrumentation.

Functions

- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`
`&__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _-`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc`
`>`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc`
`> &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &_-`
`_y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`
`const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc >`
`&__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb >`
`&__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _`
`Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const`
`set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`
`> &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`
`_Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator==(const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _CharT, typename _Traits, size_t _Nb>
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>
bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>
bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >
void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >
void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >
void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >
void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >
void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Tp, typename _Alloc >
void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >
void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >
void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

4.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior. Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_XXX` macros are merely wrappers around the `__glibcxx_check_XXX` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

4.13 std::__detail Namespace Reference

Implementation details not part of the namespace `std` interface.

Functions

- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator __transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Value, bool __cache>`
`bool operator!= (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value, bool __cache>`
`bool operator!= (const _Hashtable_iterator_base< _Value, __cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value, bool __cache>`
`bool operator== (const _Node_iterator_base< _Value, __cache > &__x, const _Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value, bool __cache>`
`bool operator== (const _Hashtable_iterator_base< _Value, __cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`

Variables

- `const unsigned long __prime_list []`

4.13.1 Detailed Description

Implementation details not part of the namespace std interface.

4.14 std::__parallel Namespace Reference

GNU parallel code, replaces standard behavior with parallel behavior.

Classes

- struct [_CRandNumber](#)
Functor wrapper for std::rand().

Functions

- template<typename _Iter, typename _Tp, typename _IteratorTag >
_Tp **accumulate_switch** (_Iter __begin, _Iter __end, _Tp __init, _-
IteratorTag)
- template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >
_Tp **accumulate_switch** (_Iter __begin, _Iter __end, _Tp __init, _-
BinaryOperation __binary_op, _IteratorTag)
- template<typename _RAIter, typename _Tp, typename _BinaryOperation >
_Tp **accumulate_switch** (_RAIter __begin, _RAIter __end, _Tp __init, _-
BinaryOperation __binary_op, [random_access_iterator_tag](#), [__gnu_parallel::_-
Parallelism](#) __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _Iter, typename _Tp, typename _Tag >
_Tp **accumulate_switch** (_Iter, _Iter, _Tp, _Tag)
- template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >
_Tp **accumulate_switch** (_Iter, _Iter, _Tp, _BinaryOper, _Tag)
- template<typename _RAIter, typename _Tp, typename _BinaryOper >
_Tp **accumulate_switch** (_RAIter, _RAIter, _Tp, _BinaryOper, [random_-
access_iterator_tag](#), [__gnu_parallel::_Parallelism](#) __parallelism=__gnu_-
parallel::parallel_unbalanced)
- template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename
_Tag2 >
_OIter **adjacent_difference_switch** (_Iter, _Iter, _OIter, _BinaryOper, _-
Tag1, _Tag2)
- template<typename _Iter, typename _OIter, typename _BinaryOper >
_OIter **adjacent_difference_switch** (_Iter, _Iter, _OIter, _BinaryOper,
[random_access_iterator_tag](#), [random_access_iterator_tag](#), [__gnu_parallel::_-
Parallelism](#) __parallelism=__gnu_parallel::parallel_unbalanced)

- `template<typename _Iter , typename _OutputIterator , typename _BinaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >`
`_OutputIterator __adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter , typename _OutputIterator , typename _BinaryOperation >`
`_OutputIterator __adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter >`
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _FIterator , typename _IteratorTag >`
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _FIterator , typename _BinaryPredicate , typename _IteratorTag >`
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _RAIter , typename _BinaryPredicate >`
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _FIter , typename _IterTag >`
`_FIter __adjacent_find_switch (_FIter, _FIter, _IterTag)`
- `template<typename _FIter , typename _BiPredicate , typename _IterTag >`
`_FIter __adjacent_find_switch (_FIter, _FIter, _BiPredicate, _IterTag)`
- `template<typename _RAIter , typename _BiPredicate >`
`_RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAIter , typename _Predicate >`
`iterator_traits< _RAIter >::difference_type __count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter , typename _Predicate , typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type __count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter , typename _Predicate , typename _IterTag >`
`iterator_traits< _Iter >::difference_type __count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter , typename _Tp >`
`iterator_traits< _RAIter >::difference_type __count_switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter , typename _Tp , typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type __count_switch (_Iter __begin, _Iter __end, const _Tp &__value, _IteratorTag)`

- `template<typename _Iter, typename _Tp, typename _IterTag >`
`iterator_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const`
`_Tp &, _IterTag)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2`
`>`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __`
`begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _`
`IteratorTag >`
`_RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator`
`__begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_`
`iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _`
`IteratorTag1, typename _IteratorTag2 >`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator _`
`__begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _`
`IteratorTag2)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate,`
`random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, type-`
`name _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _`
`IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`_Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _`
`IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter __find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred,`
`random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Iter __find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _`
`IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter __find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val,`
`random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`_Iter __find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Function >`
`_Function __for_each_switch (_RAIter __begin, _RAIter __end, _Function`

- `__f, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >
_Function __for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _Iter, typename _Function, typename _IterTag >
_Function __for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >
_OutputIterator __generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >
_RAIter __generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >
_OIter __generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag >
void __generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >
void __generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIter, typename _Generator, typename _IterTag >
void __generate_switch (_FIter, _FIter, _Generator, _IterTag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >
_Tp __inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2 >
_Tp __inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >
_Tp __inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >
_Tp __inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2,`

```

_Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2,
_IteratorTag1, _IteratorTag2)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1,
typename _IteratorTag2 >
bool __lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1,
_Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _-
IteratorTag2)
• template<typename _RAIter1, typename _RAIter2, typename _Predicate >
bool __lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 _-
end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_-
access\_iterator\_tag, random\_access\_iterator\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, type-
name _IterTag2 >
bool __lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _-
Predicate, _IterTag1, _IterTag2)
• template<typename _Filter, typename _Compare, typename _IterTag >
_Filter __max_element_switch (_Filter, _Filter, _Compare, _IterTag)
• template<typename _FIterator, typename _Compare, typename _IteratorTag >
_FIterator __max_element_switch (_FIterator __begin, _FIterator __end, _-
Compare __comp, _IteratorTag)
• template<typename _RAIter, typename _Compare >
_RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _-
Compare __comp, random\_access\_iterator\_tag, \_\_gnu\_parallel::Parallelism _-
_parallelism_tag=\_\_gnu\_parallel::parallel\_balanced)
• template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _-
IterTag1, _IterTag2, _IterTag3)
• template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >
_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-
Compare, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_-
access\_iterator\_tag)
• template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare,
typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2
__begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, _-
IteratorTag1, _IteratorTag2, _IteratorTag3)
• template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >
_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 _-
begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, random\_-
access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
• template<typename _Filter, typename _Compare, typename _IterTag >
_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)
• template<typename _FIterator, typename _Compare, typename _IteratorTag >
_FIterator __min_element_switch (_FIterator __begin, _FIterator __end, _-
Compare __comp, _IteratorTag)

```

- `template<typename _RAIter, typename _Compare >`
`_RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, random_access_iterator_tag, __gnu_parallel::Parallelism _-`
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1,`
`typename _IteratorTag2 >`
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _-`
`RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_`
`iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, type-`
`name _IterTag2 >`
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _-`
`Predicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _-`
`OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_`
`tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename`
`_IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _-`
`OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _-`
`IteratorTag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename`
`_Tag2 >`
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _-`
`Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random_`
`access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`
`_FIterator __partition_switch (_FIterator __begin, _FIterator __end, _Predicate`
`__pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate _-`
`pred, random_access_iterator_tag)`
- `template<typename _FIter, typename _Predicate, typename _IterTag >`
`_FIter __partition_switch (_FIter, _FIter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate _-`
`pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::`
`Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`

- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`
`void __replace_if_switch (_FIterator __begin, _FIterator __end, _Predicate __-`
`pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag >`
`void __replace_if_switch (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _FIter, typename _Tp, typename _IterTag >`
`void __replace_switch (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`
`void __replace_switch (_FIterator __begin, _FIterator __end, const _Tp &__-`
`old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp &__-`
`_old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=`
`__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _-`
`BiPredicate, random_access_iterator_tag)`
- `template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate, typename`
`_IterTag >`
`_FIter __search_n_switch (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate,`
`_IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __-`
`count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_-`
`iterator_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate,`
`typename _IteratorTag >`
`_FIterator __search_n_switch (_FIterator __begin, _FIterator __end, _Integer`
`__count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1,`
`typename _IterTag2 >`
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _-`
`IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2`
`__begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_-`
`iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _-`
`IteratorTag2 >`
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _-`
`FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 _-`
`__begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_-`
`tag, random_access_iterator_tag)`

- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _`
`IteratorTag1, _IteratorTag2)`
- `template<typename _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2 >`
`_FIter1 __search_switch (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _`
`IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _`
`BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator,`
`typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_difference_switch (_IIter1 __begin1, _IIter1 __end1, _`
`IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _`
`IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _`
`Predicate >`
`_Output_RAIter __set_difference_switch (_RAIter1 __begin1, _RAIter1 _`
`__end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _`
`Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,
random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename`
`_IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter __set_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _`
`Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator,`
`typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_intersection_switch (_IIter1 __begin1, _IIter1 __end1,`
`_IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred,`
`_IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _`
`Predicate >`
`_Output_RAIter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 _`
`__end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _`
`Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,
random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename`
`_IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter __set_intersection_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _`
`Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _`
`Predicate >`
`_Output_RAIter __set_symmetric_difference_switch (_RAIter1 __begin1, _`
`RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter`

- __result, _Predicate __pred, [random_access_iterator_tag](#), [random_access_iterator_tag](#))
- template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **__set_symmetric_difference_switch** (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
 - template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter **__set_symmetric_difference_switch** (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
 - template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter **__set_union_switch** (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
 - template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **__set_union_switch** (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
 - template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >
_Output_RAIter **__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, [random_access_iterator_tag](#), [random_access_iterator_tag](#), [random_access_iterator_tag](#))
 - template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >
_OIter **__transform1_switch** (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)
 - template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >
_RAOIter **__transform1_switch** (_RAIter, _RAIter, _RAOIter, _UnaryOperation, [random_access_iterator_tag](#), [random_access_iterator_tag](#), [__gnu_parallel::Parallelism](#) __parallelism=[__gnu_parallel::parallel_balanced](#))
 - template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >
_RAIter2 **__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, [random_access_iterator_tag](#), [random_access_iterator_tag](#), [__gnu_parallel::Parallelism](#) __parallelism_tag=[__gnu_parallel::parallel_balanced](#))
 - template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >
_RAIter2 **__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)
 - template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation >

- `_RAIter3 __transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism=gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3 >
_OIter __transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >
_RAIter3 __transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag=gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >
_OutputIterator __transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >
_OIter __unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >
_RandomAccess_OIter __unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >
_OutputIterator __unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >
RandomAccessOutputIterator __unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >
_Tp __accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >
_Tp __accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >
_Tp __accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`

- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::_Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp accumulate (_Iter __begin, _Iter __end, _Tp __init)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::_Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::_Parallelism)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _-`
`OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_-`
`Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _-`
`OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _FIterator >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, __gnu_-`
`parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _-`
`BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _-`
`BinaryPredicate __pred)`
- `template<typename _FIter >`
`_FIter adjacent_find (_FIter, _FIter)`
- `template<typename _FIter >`
`_FIter adjacent_find (_FIter, _FIter, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _BiPredicate >`
`_FIter adjacent_find (_FIter, _FIter, _BiPredicate)`
- `template<typename _FIter, typename _BiPredicate >`
`_FIter adjacent_find (_FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_-`
`tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`
`const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`
`const _Tp &__value, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end,`
`const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`
`_Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`
`_Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end,`
`_Predicate __pred)`

- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIterator >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIter >`
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter, _BiPredicate)`
- `template<typename _Iter, typename _FIter >`
`_Iter find_first_of (_Iter, _Iter, _FIter, _FIter)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _Function >`
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iter, typename _Function >`
`_Function for_each (_Iter, _Iter, _Function)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _FIter, typename _Generator >`
`void generate (_FIter, _FIter, _Generator)`
- `template<typename _FIter, typename _Generator >`
`void generate (_FIter, _FIter, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _Generator >`
`void generate (_FIter, _FIter, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`

- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init,`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`
`name _BinaryFunction2 >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __`
`__init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __-`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, type-`
`name BinaryFunction2 >`
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, Binary-`
`Function2, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`
`name _BinaryFunction2 >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __`
`__init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __-`
`gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-`
`name _BinaryFunction2 >`
`_Tp inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init,`
`_BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`
`begin2, _Iter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`
`begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`
`begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __`
`begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter, __gnu_parallel::Parallelism)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`

- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter)`

- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter, __gnu_parallel::Parallelism __-`
`parallelism_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`
`comp)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, __gnu-`
`parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`
`comp, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, __gnu-`
`parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __-`
`comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`
`__begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`
`__begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`
`__begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2`
`__begin2, _Predicate __pred)`
- `template<typename _RAIter >`
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu-`
`parallel::sequential_tag)`

- template<typename _RAIter, typename _Compare >
void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, [__gnu_parallel::sequential_tag](#))
- template<typename _RAIter, typename _Compare >
void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter, typename _Compare >
void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, [__gnu_parallel::sequential_tag](#))
- template<typename _RAIter >
void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, [__gnu_parallel::sequential_tag](#))
- template<typename _RAIter, typename _Compare >
void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _Iter, typename _OIter, typename _BinaryOper >
_OIter **partial_sum** (_Iter, _Iter, _OIter, _BinaryOper)
- template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator **partial_sum** (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator **partial_sum** (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, [__gnu_parallel::sequential_tag](#))
- template<typename _Iter, typename _OutputIterator >
_OutputIterator **partial_sum** (_Iter __begin, _Iter __end, _OutputIterator __result, [__gnu_parallel::sequential_tag](#))
- template<typename _Iter, typename _OIter >
_OIter **partial_sum** (_Iter, _Iter, _OIter __result)
- template<typename _Iter, typename _OIter >
_OIter **partial_sum** (_Iter, _Iter, _OIter, [__gnu_parallel::sequential_tag](#))
- template<typename _Iter, typename _OIter, typename _BinaryOper >
_OIter **partial_sum** (_Iter, _Iter, _OIter, _BinaryOper, [__gnu_parallel::sequential_tag](#))
- template<typename _Iter, typename _OutputIterator >
_OutputIterator **partial_sum** (_Iter __begin, _Iter __end, _OutputIterator __result)
- template<typename _Filter, typename _Predicate >
_Filter **partition** (_Filter, _Filter, _Predicate)
- template<typename _Filter, typename _Predicate >
_Filter **partition** (_Filter, _Filter, _Predicate, [__gnu_parallel::sequential_tag](#))

- `template<typename _FIterator, typename _Predicate >`
`_FIterator partition (_FIterator __begin, _FIterator __end, _Predicate __pred,`
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Predicate >`
`_FIterator partition (_FIterator __begin, _FIterator __end, _Predicate __pred)`
- `template<typename _RAIter >`
`void random_shuffle (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Tp >`
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIter, typename _Tp >`
`void replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _FIter, typename _Tp >`
`void replace (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_Parallelism)`
- `template<typename _FIterator, typename _Tp >`
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`
`void replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _Tp >`
`void replace (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_Parallelism)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`
`void replace_if (_FIter, _FIter, _Predicate, const _Tp &)`

- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const`
`_Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`
`begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2`
`__begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`
`begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __-`
`begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer _-`
`__count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`

- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count,`
`const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count,`
`const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count,`
`const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __-`
`begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __-`
`begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __-`
`begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __-`
`begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __-`
`begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __-`
`begin2, _IIter2 __end2, _OutputIterator __out)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate > _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate > _OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_intersection (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator > _OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator > _OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate > _OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate > _OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter > _OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Iter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`

- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter >`
`_OIter unique_copy (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`

- `template<typename _Iter, typename _OIter, typename _Predicate >
_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >
_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >
_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator
__out, _Predicate __pred, __gnu_parallel::sequential_tag)`

4.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

4.15 `std::__profile` Namespace Reference

GNU profile code, replaces standard behavior with profile behavior.

Classes

- class [bitset](#)
Class [std::bitset](#) wrapper with performance instrumentation.
- class [deque](#)
Class [std::deque](#) wrapper with performance instrumentation.
- class [list](#)
List wrapper with performance instrumentation.
- class [map](#)
Class [std::map](#) wrapper with performance instrumentation.
- class [multimap](#)
Class [std::multimap](#) wrapper with performance instrumentation.
- class [multiset](#)
Class [std::multiset](#) wrapper with performance instrumentation.
- class [set](#)
Class [std::set](#) wrapper with performance instrumentation.
- class [unordered_map](#)

Class [std::unordered_map](#) wrapper with performance instrumentation.

- class [unordered_multimap](#)

Class [std::unordered_multimap](#) wrapper with performance instrumentation.

- class [unordered_multiset](#)

[Unordered_multiset](#) wrapper with performance instrumentation.

- class [unordered_set](#)

[Unordered_set](#) wrapper with performance instrumentation.

Functions

- `template<typename _Tp, typename _Alloc >
bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >
bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >
bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >
bool operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >
bool operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >
bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`

- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >`
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<typename _Iterator , typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence > operator+ (typename __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _IteratorL , typename _IteratorR , typename _Sequence >`
`__iterator_tracker< _IteratorL, _Sequence >::difference_type operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator , typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence >::difference_type operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp , typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp , typename _Alloc >`
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_ostream< _CharT, _Traits > &operator<< (std::basic_ostream< _`
`CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`
`> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp,`
`_Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp,`
`_Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs,`
`const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const`
`set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const`
`multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator== (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator== (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _-`
`Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _-`
`Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set<`
`_Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator> (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator> (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__-`
`lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc`
`> &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs,`
`const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs,`
`const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const`
`set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp,`
`_Alloc > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`

- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
void **swap** ([unordered_set](#)< _Value, _Hash, _Pred, _Alloc > &__x, [unordered_set](#)< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
void **swap** (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)

4.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

4.16 std::chrono Namespace Reference

ISO C++ 0x entities sub namespace for time and date.

Classes

- struct [duration](#)
duration
- struct [duration_values](#)
duration_values
- struct [system_clock](#)
system_clock
- struct [time_point](#)
time_point
- struct [treat_as_floating_point](#)
treat_as_floating_point

Typedefs

- typedef [system_clock](#) **high_resolution_clock**
- typedef [duration](#)< int, [ratio](#)< 3600 > > **hours**
- typedef [duration](#)< int64_t, micro > **microseconds**
- typedef [duration](#)< int64_t, milli > **milliseconds**
- typedef [duration](#)< int, [ratio](#)< 60 > > **minutes**
- typedef [system_clock](#) **monotonic_clock**

- typedef [duration](#)< int64_t, nano > [nanoseconds](#)
- typedef [duration](#)< int64_t > [seconds](#)

Functions

- template<typename _ToDuration, typename _Rep, typename _Period >
[enable_if](#)< __is_duration< _ToDuration >::value, _ToDuration >::type
[duration_cast](#) (const [duration](#)< _Rep, _Period > &__d)
- template<typename _Clock, typename _Duration1, typename _Duration2 >
bool **operator!=** (const [time_point](#)< _Clock, _Duration1 > &__lhs, const
[time_point](#)< _Clock, _Duration2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
bool **operator!=** (const [duration](#)< _Rep1, _Period1 > &__lhs, const [duration](#)<
_Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
common_type< [duration](#)< _Rep1, _Period1 >, [duration](#)< _Rep2, _Period2 >
>::type **operator%** (const [duration](#)< _Rep1, _Period1 > &__lhs, const [dura-](#)
[tion](#)< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period, typename _Rep2 >
[duration](#)< typename __common_rep_type< _Rep1, typename [enable_if](#)<!_
is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period > **operator%**
(const [duration](#)< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period, typename _Rep2 >
[duration](#)< typename __common_rep_type< _Rep1, _Rep2 >::type, _Period >
operator* (const [duration](#)< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period, typename _Rep2 >
[duration](#)< typename __common_rep_type< _Rep2, _Rep1 >::type, _Period >
operator* (const _Rep1 &__s, const [duration](#)< _Rep2, _Period > &__d)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >
common_type< [duration](#)< _Rep1, _Period1 >, [duration](#)< _Rep2, _Period2 >
>::type **operator+** (const [duration](#)< _Rep1, _Period1 > &__lhs, const [dura-](#)
[tion](#)< _Rep2, _Period2 > &__rhs)
- template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >
[time_point](#)< _Clock, typename common_type< _Duration1, [duration](#)< _Rep2,
_Period2 > >::type > **operator+** (const [time_point](#)< _Clock, _Duration1 >
&__lhs, const [duration](#)< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Clock, typename _Duration2 >
[time_point](#)< _Clock, typename common_type< [duration](#)< _Rep1, _Period1 >,
_Duration2 >::type > **operator+** (const [duration](#)< _Rep1, _Period1 > &__lhs,
const [time_point](#)< _Clock, _Duration2 > &__rhs)
- template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >
[time_point](#)< _Clock, typename common_type< _Duration1, [duration](#)< _Rep2,
_Period2 > >::type > **operator-** (const [time_point](#)< _Clock, _Duration1 > &_
__lhs, const [duration](#)< _Rep2, _Period2 > &__rhs)

- `template<typename _Clock, typename _Duration1, typename _Duration2 >`
`common_type< _Duration1, _Duration2 >::type operator- (const time_point<`
`_Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__`
`rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 >`
`>::type operator- (const duration< _Rep1, _Period1 > &__lhs, const dura-`
`tion< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`duration< typename __common_rep_type< _Rep1, typename enable_if<!__`
`is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period > operator/`
`(const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`common_type< _Rep1, _Rep2 >::type operator/ (const duration< _Rep1, _`
`Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`
`bool operator< (const time_point< _Clock, _Duration1 > &__lhs, const time_-`
`point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`bool operator< (const duration< _Rep1, _Period1 > &__lhs, const duration<`
`_Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`bool operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration<`
`_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`
`bool operator<= (const time_point< _Clock, _Duration1 > &__lhs, const`
`time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`
`bool operator== (const time_point< _Clock, _Duration1 > &__lhs, const`
`time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration<`
`_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`
`bool operator> (const time_point< _Clock, _Duration1 > &__lhs, const time_-`
`point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`bool operator> (const duration< _Rep1, _Period1 > &__lhs, const duration<`
`_Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`
`bool operator>= (const time_point< _Clock, _Duration1 > &__lhs, const`
`time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration<`
`_Rep2, _Period2 > &__rhs)`

- `template<typename _ToDuration, typename _Clock, typename _Duration >
enable_if< __is_duration< _ToDuration >::value, time_point< _Clock, _-
ToDuration > >::type time_point_cast (const time_point< _Clock, _Duration
> &__t)`

4.16.1 Detailed Description

ISO C++ 0x entities sub namespace for time and date.

4.16.2 Typedef Documentation

4.16.2.1 `typedef duration<int, ratio<3600> > std::chrono::hours`

hours

Definition at line 488 of file chrono.

4.16.2.2 `typedef duration<int64_t, micro> std::chrono::microseconds`

microseconds

Definition at line 476 of file chrono.

4.16.2.3 `typedef duration<int64_t, milli> std::chrono::milliseconds`

milliseconds

Definition at line 479 of file chrono.

4.16.2.4 `typedef duration<int, ratio< 60> > std::chrono::minutes`

minutes

Definition at line 485 of file chrono.

4.16.2.5 `typedef duration<int64_t, nano> std::chrono::nanoseconds`

nanoseconds

Definition at line 473 of file chrono.

4.16.2.6 typedef duration<int64_t> std::chrono::seconds

seconds

Definition at line 482 of file chrono.

4.16.3 Function Documentation

4.16.3.1 `template<typename _ToDuration , typename _Rep , typename
_Period > enable_if<__is_duration<_ToDuration>::value,
_ToDuration>::type std::chrono::duration_cast (const duration<
_Rep, _Period > & __d) [inline]`

duration_cast

Definition at line 155 of file chrono.

Referenced by std::this_thread::sleep_for(), and time_point_cast().

4.16.3.2 `template<typename _ToDuration , typename _Clock , typename
_Duration > enable_if<__is_duration<_ToDuration>::value, time_
point<_Clock, _ToDuration> >::type std::chrono::time_point_cast (
const time_point< _Clock, _Duration > & __t) [inline]`

time_point_cast

Definition at line 550 of file chrono.

References duration_cast().

4.17 std::decimal Namespace Reference

ISO/IEC TR 24733 Decimal floating-point arithmetic.

Classes

- class [decimal128](#)
3.2.4 Class [decimal128](#).
- class [decimal32](#)
3.2.2 Class [decimal32](#).
- class [decimal64](#)
3.2.3 Class [decimal64](#).

Functions

- double **decimal128_to_double** ([decimal128](#) __d)
- float **decimal128_to_float** ([decimal128](#) __d)
- long double **decimal128_to_long_double** ([decimal128](#) __d)
- long long **decimal128_to_long_long** ([decimal128](#) __d)
- double **decimal32_to_double** ([decimal32](#) __d)
- float **decimal32_to_float** ([decimal32](#) __d)
- long double **decimal32_to_long_double** ([decimal32](#) __d)
- long long [decimal32_to_long_long](#) ([decimal32](#) __d)
- double **decimal64_to_double** ([decimal64](#) __d)
- float **decimal64_to_float** ([decimal64](#) __d)
- long double **decimal64_to_long_double** ([decimal64](#) __d)
- long long **decimal64_to_long_long** ([decimal64](#) __d)
- double **decimal_to_double** ([decimal32](#) __d)
- double **decimal_to_double** ([decimal64](#) __d)
- double **decimal_to_double** ([decimal128](#) __d)
- float **decimal_to_float** ([decimal32](#) __d)
- float **decimal_to_float** ([decimal64](#) __d)
- float **decimal_to_float** ([decimal128](#) __d)
- long double **decimal_to_long_double** ([decimal32](#) __d)
- long double **decimal_to_long_double** ([decimal64](#) __d)
- long double **decimal_to_long_double** ([decimal128](#) __d)
- long long **decimal_to_long_long** ([decimal32](#) __d)
- long long **decimal_to_long_long** ([decimal64](#) __d)
- long long **decimal_to_long_long** ([decimal128](#) __d)
- static [decimal128](#) **make_decimal128** (long long __coeff, int __exp)
- static [decimal128](#) **make_decimal128** (unsigned long long __coeff, int __exp)
- static [decimal32](#) **make_decimal32** (long long __coeff, int __exp)
- static [decimal32](#) **make_decimal32** (unsigned long long __coeff, int __exp)
- static [decimal64](#) **make_decimal64** (unsigned long long __coeff, int __exp)
- static [decimal64](#) **make_decimal64** (long long __coeff, int __exp)
- bool **operator!=** ([decimal32](#) __lhs, [decimal32](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, [decimal64](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, [decimal128](#) __rhs)
- bool **operator!=** ([decimal32](#) __lhs, int __rhs)
- bool **operator!=** ([decimal32](#) __lhs, unsigned int __rhs)
- bool **operator!=** ([decimal32](#) __lhs, long __rhs)
- bool **operator!=** ([decimal32](#) __lhs, unsigned long __rhs)
- bool **operator!=** ([decimal32](#) __lhs, long long __rhs)

- bool **operator!=** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal32 __rhs)
- bool **operator!=** (long __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal32 __rhs)
- bool **operator!=** (long long __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal64 __lhs, int __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal64 __lhs, long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal64 __lhs, long long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator!=** (long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator!=** (long long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal128 __lhs, int __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal128 __lhs, long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal128 __lhs, long long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal128 __rhs)
- bool **operator!=** (long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator!=** (long long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator*** (decimal32 __lhs, int __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long long __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long long __rhs)

- [decimal32 operator*](#) (int __lhs, [decimal32](#) __rhs)
- [decimal32 operator*](#) (unsigned int __lhs, [decimal32](#) __rhs)
- [decimal32 operator*](#) (long __lhs, [decimal32](#) __rhs)
- [decimal32 operator*](#) (unsigned long __lhs, [decimal32](#) __rhs)
- [decimal32 operator*](#) (long long __lhs, [decimal32](#) __rhs)
- [decimal32 operator*](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- [decimal64 operator*](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator*](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- [decimal64 operator*](#) ([decimal64](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator*](#) ([decimal64](#) __lhs, int __rhs)
- [decimal64 operator*](#) ([decimal64](#) __lhs, unsigned int __rhs)
- [decimal64 operator*](#) ([decimal64](#) __lhs, long __rhs)
- [decimal64 operator*](#) ([decimal64](#) __lhs, unsigned long __rhs)
- [decimal64 operator*](#) ([decimal64](#) __lhs, long long __rhs)
- [decimal64 operator*](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- [decimal64 operator*](#) (int __lhs, [decimal64](#) __rhs)
- [decimal64 operator*](#) (unsigned int __lhs, [decimal64](#) __rhs)
- [decimal64 operator*](#) (long __lhs, [decimal64](#) __rhs)
- [decimal64 operator*](#) (unsigned long __lhs, [decimal64](#) __rhs)
- [decimal64 operator*](#) (long long __lhs, [decimal64](#) __rhs)
- [decimal64 operator*](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- [decimal128 operator*](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, int __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, unsigned int __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, long __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, long long __rhs)
- [decimal128 operator*](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- [decimal128 operator*](#) (int __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (long __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (long long __lhs, [decimal128](#) __rhs)
- [decimal128 operator*](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- [decimal32 operator*](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- [decimal64 operator+](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) ([decimal64](#) __rhs)
- [decimal128 operator+](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)

- [decimal128 operator+](#) ([decimal128](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, int __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, unsigned int __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, long __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, long long __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- [decimal128 operator+](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- [decimal128 operator+](#) (int __lhs, [decimal128](#) __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, int __rhs)
- [decimal128 operator+](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) (long __lhs, [decimal128](#) __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, unsigned int __rhs)
- [decimal128 operator+](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [decimal128 operator+](#) (long long __lhs, [decimal128](#) __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, long __rhs)
- [decimal128 operator+](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, unsigned long __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, long long __rhs)
- [decimal32 operator+](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- [decimal32 operator+](#) (int __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (unsigned int __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (long __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (unsigned long __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (long long __lhs, [decimal32](#) __rhs)
- [decimal32 operator+](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- [decimal64 operator+](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, int __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, unsigned int __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, long __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, unsigned long __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, long long __rhs)
- [decimal64 operator+](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- [decimal64 operator+](#) (int __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) (unsigned int __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) (long __lhs, [decimal64](#) __rhs)
- [decimal64 operator+](#) (unsigned long __lhs, [decimal64](#) __rhs)
- [decimal32 operator+](#) ([decimal32](#) __rhs)

- [decimal64 operator+](#) (long long __lhs, [decimal64](#) __rhs)
- [decimal32 operator-](#) ([decimal32](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, int __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned int __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, long long __rhs)
- [decimal32 operator-](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- [decimal32 operator-](#) (int __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned int __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (long long __lhs, [decimal32](#) __rhs)
- [decimal32 operator-](#) (unsigned long long __lhs, [decimal32](#) __rhs)
- [decimal64 operator-](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, int __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned int __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, long long __rhs)
- [decimal64 operator-](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- [decimal64 operator-](#) (int __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned int __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (long long __lhs, [decimal64](#) __rhs)
- [decimal64 operator-](#) (unsigned long long __lhs, [decimal64](#) __rhs)
- [decimal128 operator-](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal64](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal64](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, int __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, unsigned int __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, long __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, long long __rhs)
- [decimal128 operator-](#) ([decimal128](#) __lhs, unsigned long long __rhs)

- **decimal128 operator-** (int __lhs, [decimal128](#) __rhs)
- **decimal128 operator-** (unsigned int __lhs, [decimal128](#) __rhs)
- **decimal128 operator-** (long __lhs, [decimal128](#) __rhs)
- **decimal128 operator-** (unsigned long __lhs, [decimal128](#) __rhs)
- **decimal128 operator-** (long long __lhs, [decimal128](#) __rhs)
- **decimal128 operator-** (unsigned long long __lhs, [decimal128](#) __rhs)
- **decimal32 operator/** ([decimal32](#) __lhs, [decimal32](#) __rhs)
- **decimal32 operator/** ([decimal32](#) __lhs, int __rhs)
- **decimal32 operator/** ([decimal32](#) __lhs, unsigned int __rhs)
- **decimal32 operator/** ([decimal32](#) __lhs, long __rhs)
- **decimal32 operator/** ([decimal32](#) __lhs, unsigned long __rhs)
- **decimal32 operator/** ([decimal32](#) __lhs, long long __rhs)
- **decimal32 operator/** ([decimal32](#) __lhs, unsigned long long __rhs)
- **decimal32 operator/** (int __lhs, [decimal32](#) __rhs)
- **decimal32 operator/** (unsigned int __lhs, [decimal32](#) __rhs)
- **decimal32 operator/** (long __lhs, [decimal32](#) __rhs)
- **decimal32 operator/** (unsigned long __lhs, [decimal32](#) __rhs)
- **decimal128 operator/** (long long __lhs, [decimal128](#) __rhs)
- **decimal32 operator/** (long long __lhs, [decimal32](#) __rhs)
- **decimal32 operator/** (unsigned long long __lhs, [decimal32](#) __rhs)
- **decimal64 operator/** ([decimal32](#) __lhs, [decimal64](#) __rhs)
- **decimal64 operator/** ([decimal64](#) __lhs, [decimal32](#) __rhs)
- **decimal64 operator/** ([decimal64](#) __lhs, [decimal64](#) __rhs)
- **decimal64 operator/** ([decimal64](#) __lhs, int __rhs)
- **decimal64 operator/** ([decimal64](#) __lhs, unsigned int __rhs)
- **decimal128 operator/** ([decimal128](#) __lhs, long __rhs)
- **decimal64 operator/** ([decimal64](#) __lhs, long __rhs)
- **decimal64 operator/** ([decimal64](#) __lhs, unsigned long __rhs)
- **decimal64 operator/** ([decimal64](#) __lhs, long long __rhs)
- **decimal128 operator/** ([decimal128](#) __lhs, [decimal64](#) __rhs)
- **decimal64 operator/** ([decimal64](#) __lhs, unsigned long long __rhs)
- **decimal64 operator/** (int __lhs, [decimal64](#) __rhs)
- **decimal64 operator/** (unsigned int __lhs, [decimal64](#) __rhs)
- **decimal64 operator/** (long __lhs, [decimal64](#) __rhs)
- **decimal64 operator/** (unsigned long __lhs, [decimal64](#) __rhs)
- **decimal64 operator/** (long long __lhs, [decimal64](#) __rhs)
- **decimal64 operator/** (unsigned long long __lhs, [decimal64](#) __rhs)
- **decimal128 operator/** ([decimal32](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator/** ([decimal64](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator/** ([decimal128](#) __lhs, [decimal32](#) __rhs)
- **decimal128 operator/** ([decimal128](#) __lhs, [decimal128](#) __rhs)
- **decimal128 operator/** ([decimal128](#) __lhs, int __rhs)
- **decimal128 operator/** ([decimal128](#) __lhs, unsigned int __rhs)

- [decimal128 operator/](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, long long __rhs)
- [decimal128 operator/](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- [decimal128 operator/](#) (int __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (long __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [decimal128 operator/](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- [bool operator<](#) ([decimal128](#) __lhs, [decimal32](#) __rhs)
- [bool operator<](#) ([decimal64](#) __lhs, [decimal32](#) __rhs)
- [bool operator<](#) (int __lhs, [decimal32](#) __rhs)
- [bool operator<](#) (unsigned long __lhs, [decimal32](#) __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, unsigned long long __rhs)
- [bool operator<](#) (unsigned int __lhs, [decimal32](#) __rhs)
- [bool operator<](#) (long __lhs, [decimal32](#) __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, unsigned int __rhs)
- [bool operator<](#) (unsigned int __lhs, [decimal128](#) __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, long __rhs)
- [bool operator<](#) ([decimal64](#) __lhs, unsigned long long __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, long long __rhs)
- [bool operator<](#) (unsigned long long __lhs, [decimal128](#) __rhs)
- [bool operator<](#) (long __lhs, [decimal128](#) __rhs)
- [bool operator<](#) (unsigned long __lhs, [decimal64](#) __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, unsigned long __rhs)
- [bool operator<](#) (unsigned long __lhs, [decimal128](#) __rhs)
- [bool operator<](#) (long long __lhs, [decimal128](#) __rhs)
- [bool operator<](#) ([decimal64](#) __lhs, unsigned int __rhs)
- [bool operator<](#) ([decimal64](#) __lhs, int __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, [decimal128](#) __rhs)
- [bool operator<](#) ([decimal128](#) __lhs, [decimal128](#) __rhs)
- [bool operator<](#) ([decimal128](#) __lhs, long __rhs)
- [bool operator<](#) ([decimal128](#) __lhs, unsigned int __rhs)
- [bool operator<](#) ([decimal128](#) __lhs, int __rhs)
- [bool operator<](#) ([decimal128](#) __lhs, long long __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, int __rhs)
- [bool operator<](#) ([decimal128](#) __lhs, unsigned long long __rhs)
- [bool operator<](#) (int __lhs, [decimal128](#) __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, [decimal64](#) __rhs)
- [bool operator<](#) ([decimal128](#) __lhs, unsigned long __rhs)
- [bool operator<](#) ([decimal32](#) __lhs, [decimal32](#) __rhs)
- [bool operator<](#) (int __lhs, [decimal64](#) __rhs)
- [bool operator<](#) (long long __lhs, [decimal32](#) __rhs)
- [bool operator<](#) (unsigned long long __lhs, [decimal32](#) __rhs)

- bool **operator**< (unsigned long long __lhs, [decimal64](#) __rhs)
- bool **operator**< ([decimal64](#) __lhs, [decimal128](#) __rhs)
- bool **operator**< (unsigned int __lhs, [decimal64](#) __rhs)
- bool **operator**< (long long __lhs, [decimal64](#) __rhs)
- bool **operator**< (long __lhs, [decimal64](#) __rhs)
- bool **operator**< ([decimal64](#) __lhs, long __rhs)
- bool **operator**< ([decimal64](#) __lhs, unsigned long __rhs)
- bool **operator**< ([decimal128](#) __lhs, [decimal64](#) __rhs)
- bool **operator**< ([decimal64](#) __lhs, long long __rhs)
- bool **operator**< ([decimal64](#) __lhs, [decimal64](#) __rhs)
- bool **operator**== (int __lhs, [decimal128](#) __rhs)
- bool **operator**== (unsigned int __lhs, [decimal128](#) __rhs)
- bool **operator**== (long __lhs, [decimal128](#) __rhs)
- bool **operator**== (unsigned long __lhs, [decimal128](#) __rhs)
- bool **operator**== (long long __lhs, [decimal128](#) __rhs)
- bool **operator**== (unsigned long long __lhs, [decimal128](#) __rhs)
- bool **operator**== ([decimal128](#) __lhs, unsigned long long __rhs)
- bool **operator**== ([decimal32](#) __lhs, unsigned long __rhs)
- bool **operator**== ([decimal32](#) __lhs, [decimal128](#) __rhs)
- bool **operator**== ([decimal128](#) __lhs, long long __rhs)
- bool **operator**== ([decimal128](#) __lhs, long __rhs)
- bool **operator**== ([decimal32](#) __lhs, long __rhs)
- bool **operator**== ([decimal32](#) __lhs, unsigned int __rhs)
- bool **operator**== ([decimal64](#) __lhs, int __rhs)
- bool **operator**== ([decimal128](#) __lhs, unsigned int __rhs)
- bool **operator**== (long __lhs, [decimal64](#) __rhs)
- bool **operator**== ([decimal128](#) __lhs, unsigned long __rhs)
- bool **operator**== ([decimal64](#) __lhs, long long __rhs)
- bool **operator**== ([decimal64](#) __lhs, unsigned int __rhs)
- bool **operator**== ([decimal64](#) __lhs, [decimal128](#) __rhs)
- bool **operator**== ([decimal64](#) __lhs, [decimal64](#) __rhs)
- bool **operator**== (long long __lhs, [decimal32](#) __rhs)
- bool **operator**== (unsigned long __lhs, [decimal32](#) __rhs)
- bool **operator**== ([decimal128](#) __lhs, [decimal128](#) __rhs)
- bool **operator**== (long long __lhs, [decimal64](#) __rhs)
- bool **operator**== ([decimal32](#) __lhs, [decimal32](#) __rhs)
- bool **operator**== ([decimal32](#) __lhs, [decimal64](#) __rhs)
- bool **operator**== (unsigned long long __lhs, [decimal64](#) __rhs)
- bool **operator**== ([decimal32](#) __lhs, int __rhs)
- bool **operator**== ([decimal128](#) __lhs, [decimal32](#) __rhs)
- bool **operator**== ([decimal32](#) __lhs, long long __rhs)
- bool **operator**== ([decimal32](#) __lhs, unsigned long long __rhs)
- bool **operator**== (int __lhs, [decimal32](#) __rhs)

- bool **operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **operator==** (long __lhs, decimal32 __rhs)
- bool **operator==** (decimal64 __lhs, long __rhs)
- bool **operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator==** (int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, int __rhs)
- bool **operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator>** (decimal32 __lhs, decimal64 __rhs)
- bool **operator>** (decimal64 __lhs, decimal32 __rhs)
- bool **operator>** (decimal64 __lhs, decimal128 __rhs)
- bool **operator>** (long __lhs, decimal32 __rhs)
- bool **operator>** (unsigned long __lhs, decimal32 __rhs)
- bool **operator>** (decimal128 __lhs, int __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long __rhs)
- bool **operator>** (decimal64 __lhs, unsigned int __rhs)
- bool **operator>** (unsigned int __lhs, decimal32 __rhs)
- bool **operator>** (int __lhs, decimal64 __rhs)
- bool **operator>** (decimal32 __lhs, decimal128 __rhs)
- bool **operator>** (decimal32 __lhs, long long __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator>** (decimal32 __lhs, int __rhs)
- bool **operator>** (decimal32 __lhs, decimal32 __rhs)
- bool **operator>** (long long __lhs, decimal64 __rhs)
- bool **operator>** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator>** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator>** (decimal64 __lhs, decimal64 __rhs)
- bool **operator>** (long __lhs, decimal128 __rhs)
- bool **operator>** (int __lhs, decimal128 __rhs)
- bool **operator>** (decimal32 __lhs, unsigned int __rhs)
- bool **operator>** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator>** (long long __lhs, decimal32 __rhs)
- bool **operator>** (decimal128 __lhs, unsigned int __rhs)
- bool **operator>** (unsigned int __lhs, decimal128 __rhs)
- bool **operator>** (decimal128 __lhs, decimal128 __rhs)
- bool **operator>** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator>** (long long __lhs, decimal128 __rhs)
- bool **operator>** (decimal64 __lhs, unsigned long __rhs)
- bool **operator>** (decimal128 __lhs, long __rhs)

- bool **operator**> (decimal64 __lhs, long __rhs)
- bool **operator**> (decimal128 __lhs, decimal32 __rhs)
- bool **operator**> (decimal128 __lhs, unsigned long __rhs)
- bool **operator**> (decimal64 __lhs, int __rhs)
- bool **operator**> (decimal128 __lhs, long long __rhs)
- bool **operator**> (decimal128 __lhs, decimal64 __rhs)
- bool **operator**> (unsigned long __lhs, decimal128 __rhs)
- bool **operator**> (decimal64 __lhs, long long __rhs)
- bool **operator**> (unsigned int __lhs, decimal64 __rhs)
- bool **operator**> (unsigned long __lhs, decimal64 __rhs)
- bool **operator**> (long __lhs, decimal64 __rhs)
- bool **operator**> (decimal32 __lhs, long __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**> (int __lhs, decimal32 __rhs)
- bool **operator**>= (unsigned long __lhs, decimal32 __rhs)
- bool **operator**>= (decimal64 __lhs, int __rhs)
- bool **operator**>= (decimal128 __lhs, int __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned long long __rhs)
- bool **operator**>= (decimal32 __lhs, long __rhs)
- bool **operator**>= (decimal32 __lhs, decimal64 __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned long long __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned long __rhs)
- bool **operator**>= (decimal128 __lhs, long __rhs)
- bool **operator**>= (decimal32 __lhs, long long __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned int __rhs)
- bool **operator**>= (long long __lhs, decimal32 __rhs)
- bool **operator**>= (decimal128 __lhs, decimal128 __rhs)
- bool **operator**>= (decimal64 __lhs, decimal64 __rhs)
- bool **operator**>= (decimal32 __lhs, int __rhs)
- bool **operator**>= (decimal64 __lhs, long long __rhs)
- bool **operator**>= (unsigned int __lhs, decimal32 __rhs)
- bool **operator**>= (long long __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned int __rhs)
- bool **operator**>= (unsigned long long __lhs, decimal128 __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned long __rhs)
- bool **operator**>= (decimal64 __lhs, decimal32 __rhs)
- bool **operator**>= (int __lhs, decimal128 __rhs)
- bool **operator**>= (decimal32 __lhs, decimal32 __rhs)
- bool **operator**>= (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned int __rhs)
- bool **operator**>= (decimal128 __lhs, decimal32 __rhs)
- bool **operator**>= (unsigned long __lhs, decimal64 __rhs)

- bool **operator**>= (unsigned int __lhs, decimal64 __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned long __rhs)
- bool **operator**>= (long __lhs, decimal128 __rhs)
- bool **operator**>= (int __lhs, decimal64 __rhs)
- bool **operator**>= (decimal32 __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, long long __rhs)
- bool **operator**>= (int __lhs, decimal32 __rhs)
- bool **operator**>= (decimal64 __lhs, decimal128 __rhs)
- bool **operator**>= (long __lhs, decimal64 __rhs)
- bool **operator**>= (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**>= (long long __lhs, decimal64 __rhs)
- bool **operator**>= (unsigned int __lhs, decimal128 __rhs)
- bool **operator**>= (long __lhs, decimal32 __rhs)
- bool **operator**>= (decimal64 __lhs, long __rhs)
- bool **operator**>= (unsigned long __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, decimal64 __rhs)

4.17.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

4.17.2 Function Documentation

4.17.2.1 long long std::decimal::decimal32_to_long_long (decimal32 __d)

Non-conforming extension: Conversion to integral type.

4.18 std::placeholders Namespace Reference

ISO C++ 0x entities sub namespace for functional.

Define a large number of placeholders. There is no way to simplify this with variadic templates, because we're introducing unique names for each.

4.18.1 Detailed Description

ISO C++ 0x entities sub namespace for functional.

Define a large number of placeholders. There is no way to simplify this with variadic templates, because we're introducing unique names for each.

4.19 `std::regex_constants` Namespace Reference

ISO C++-0x entities sub namespace for regex.

5.1 Regular Expression Syntax Options

- enum `__syntax_option` {
`_S_icode`, `_S_nosubs`, `_S_optimize`, `_S_collate`,
`_S_ECMAScript`, `_S_basic`, `_S_extended`, `_S_awk`,
`_S_grep`, `_S_egrep`, `_S_syntax_last` }
- typedef unsigned int `syntax_option_type`
- static const `syntax_option_type` `icase`
- static const `syntax_option_type` `nosubs`
- static const `syntax_option_type` `optimize`
- static const `syntax_option_type` `collate`
- static const `syntax_option_type` `ECMAScript`
- static const `syntax_option_type` `basic`
- static const `syntax_option_type` `extended`
- static const `syntax_option_type` `awk`
- static const `syntax_option_type` `grep`
- static const `syntax_option_type` `egrep`

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `__match_flag` {
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- typedef `std::bitset<_S_match_flag_last>` `match_flag_type`
- static const `match_flag_type` `match_default`
- static const `match_flag_type` `match_not_bol`
- static const `match_flag_type` `match_not_eol`
- static const `match_flag_type` `match_not_bow`
- static const `match_flag_type` `match_not_eow`
- static const `match_flag_type` `match_any`
- static const `match_flag_type` `match_not_null`

- static const [match_flag_type](#) [match_continuous](#)
- static const [match_flag_type](#) [match_prev_avail](#)
- static const [match_flag_type](#) [format_default](#)
- static const [match_flag_type](#) [format_sed](#)
- static const [match_flag_type](#) [format_no_copy](#)
- static const [match_flag_type](#) [format_first_only](#)

5.3 Error Types

- enum [error_type](#) {
 [_S_error_collate](#), [_S_error_ctype](#), [_S_error_escape](#), [_S_error_backref](#),
 [_S_error_brack](#), [_S_error_paren](#), [_S_error_brace](#), [_S_error_badbrace](#),
 [_S_error_range](#), [_S_error_space](#), [_S_error_badrepeat](#), [_S_error_](#)-
 [complexity](#),
 [_S_error_stack](#), [_S_error_last](#) }
 • static const [error_type](#) [error_collate](#) ([_S_error_collate](#))
 • static const [error_type](#) [error_ctype](#) ([_S_error_ctype](#))
 • static const [error_type](#) [error_escape](#) ([_S_error_escape](#))
 • static const [error_type](#) [error_backref](#) ([_S_error_backref](#))
 • static const [error_type](#) [error_brack](#) ([_S_error_brack](#))
 • static const [error_type](#) [error_paren](#) ([_S_error_paren](#))
 • static const [error_type](#) [error_brace](#) ([_S_error_brace](#))
 • static const [error_type](#) [error_badbrace](#) ([_S_error_badbrace](#))
 • static const [error_type](#) [error_range](#) ([_S_error_range](#))
 • static const [error_type](#) [error_space](#) ([_S_error_space](#))
 • static const [error_type](#) [error_badrepeat](#) ([_S_error_badrepeat](#))
 • static const [error_type](#) [error_complexity](#) ([_S_error_complexity](#))
 • static const [error_type](#) [error_stack](#) ([_S_error_stack](#))

4.19.1 Detailed Description

ISO C++-0x entities sub namespace for regex.

4.19.2 Typedef Documentation

4.19.2.1 `typedef std::bitset<_S_match_flag_last> std::regex_constants::match_flag_type`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 190 of file `regex_constants.h`.

4.19.2.2 `typedef unsigned int std::regex_constants::syntax_option_type`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 72 of file `regex_constants.h`.

4.19.3 Enumeration Type Documentation

4.19.3.1 `enum std::regex_constants::__match_flag`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 167 of file `regex_constants.h`.

4.19.3.2 `enum std::regex_constants::__syntax_option`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 46 of file `regex_constants.h`.

4.19.3.3 `enum std::regex_constants::error_type`

The expression contained an invalid collating element name.

Definition at line 43 of file `regex_error.h`.

4.19.4 Function Documentation

4.19.4.1 static const error_type std::regex_constants::error_backref (
 _S_error_backref) [static]

The expression contained an invalid back reference.

4.19.4.2 static const error_type std::regex_constants::error_badbrace (
 _S_error_badbrace) [static]

The expression contained an invalid range in a { } expression.

4.19.4.3 static const error_type std::regex_constants::error_badrepeat (
 _S_error_badrepeat) [static]

One of **?+{* was not preceded by a valid regular expression.

4.19.4.4 static const error_type std::regex_constants::error_brace (
 _S_error_brace) [static]

The expression contained mismatched { and }

4.19.4.5 static const error_type std::regex_constants::error_brack (
 _S_error_brack) [static]

The expression contained mismatched [and].

4.19.4.6 static const error_type std::regex_constants::error_collate (
 _S_error_collate) [static]

The expression contained an invalid collating element name.

4.19.4.7 static const error_type std::regex_constants::error_complexity (
 _S_error_complexity) [static]

The complexity of an attempted match against a regular expression exceeded a pre-set level.

4.19.4.8 `static const error_type std::regex_constants::error_ctype (_S_error_ctype) [static]`

The expression contained an invalid character class name.

4.19.4.9 `static const error_type std::regex_constants::error_escape (_S_error_escape) [static]`

The expression contained an invalid escaped character, or a trailing escape.

4.19.4.10 `static const error_type std::regex_constants::error_paren (_S_error_paren) [static]`

The expression contained mismatched (and).

4.19.4.11 `static const error_type std::regex_constants::error_range (_S_error_range) [static]`

The expression contained an invalid character range, such as [b-a] in most encodings.

4.19.4.12 `static const error_type std::regex_constants::error_space (_S_error_space) [static]`

There was insufficient memory to convert the expression into a finite state machine.

4.19.4.13 `static const error_type std::regex_constants::error_stack (_S_error_stack) [static]`

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

4.19.5 Variable Documentation

4.19.5.1 `const syntax_option_type std::regex_constants::awk [static]`

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `awk` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\'`, `\`, and `\ddd` (where `ddd` is one, two, or three octal digits).

Definition at line 136 of file `regex_constants.h`.

4.19.5.2 const syntax_option_type std::regex_constants::basic [static]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology -- Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 118 of file regex_constants.h.

4.19.5.3 const syntax_option_type std::regex_constants::collate [static]

Specifies that character ranges of the form [a-b] should be locale sensitive.

Definition at line 99 of file regex_constants.h.

4.19.5.4 const syntax_option_type std::regex_constants::ECMAScript [static]

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

Definition at line 109 of file regex_constants.h.

4.19.5.5 const syntax_option_type std::regex_constants::egrep [static]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep when given the -E option in IEEE Std 1003.1-2001. This option is identical to syntax_option_type extended, except that newlines are treated as whitespace.

Definition at line 152 of file regex_constants.h.

4.19.5.6 const syntax_option_type std::regex_constants::extended [static]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

Definition at line 126 of file regex_constants.h.

4.19.5.7 `const match_flag_type std::regex_constants::format_default` [static]

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript replace function in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 String.prototype.replace. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`)
- `$&` The matched substring.
- `$'` The portion of *string* that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of *string* that follows the matched substring. This would be `match_results::suffix()`.
- `$n` The *n*th capture, where *n* is in [1,9] and `$n` is not followed by a decimal digit. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on [01, 99]. If `nn <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `nn > match_results::size()`, the result is implementation-defined.

Definition at line 272 of file `regex_constants.h`.

4.19.5.8 `const match_flag_type std::regex_constants::format_first_only` [static]

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 293 of file `regex_constants.h`.

4.19.5.9 `const match_flag_type std::regex_constants::format_no_copy` [static]

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

Definition at line 287 of file `regex_constants.h`.

4.19.5.10 const match_flag_type std::regex_constants::format_sed [static]

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX sed utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology -- Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 280 of file regex_constants.h.

4.19.5.11 const syntax_option_type std::regex_constants::grep [static]

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep in IEEE Std 1003.1-2001. This option is identical to syntax_option_type basic, except that newlines are treated as whitespace.

Definition at line 144 of file regex_constants.h.

4.19.5.12 const syntax_option_type std::regex_constants::icase [static]

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

Definition at line 78 of file regex_constants.h.

4.19.5.13 const match_flag_type std::regex_constants::match_any [static]

If more than one match is possible then any match is an acceptable result.

Definition at line 227 of file regex_constants.h.

4.19.5.14 const match_flag_type std::regex_constants::match_continuous [static]

The expression only matches a sub-sequence that begins at first .

Definition at line 237 of file regex_constants.h.

4.19.5.15 const match_flag_type std::regex_constants::match_default [static]

The default matching rules.

Definition at line 195 of file regex_constants.h.

4.19.5.16 `const match_flag_type std::regex_constants::match_not_bol`
`[static]`

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

Definition at line 202 of file regex_constants.h.

4.19.5.17 `const match_flag_type std::regex_constants::match_not_bow`
`[static]`

The expression \b is not matched against the sub-sequence [first,first).

Definition at line 215 of file regex_constants.h.

4.19.5.18 `const match_flag_type std::regex_constants::match_not_eol`
`[static]`

The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character (\$) in the regular expression shall not match [last, last).

Definition at line 209 of file regex_constants.h.

4.19.5.19 `const match_flag_type std::regex_constants::match_not_eow`
`[static]`

The expression \b should not be matched against the sub-sequence [last,last).

Definition at line 221 of file regex_constants.h.

4.19.5.20 `const match_flag_type std::regex_constants::match_not_null`
`[static]`

The expression does not match an empty sequence.

Definition at line 232 of file regex_constants.h.

4.19.5.21 `const match_flag_type std::regex_constants::match_prev_avail`
`[static]`

--first is a valid iterator position. When this flag is set then the flags match_not_bol and match_not_bow are ignored by the regular expression algorithms 7.11 and iterators 7.12.

Definition at line 244 of file regex_constants.h.

4.19.5.22 const syntax_option_type std::regex_constants::nosubs [static]

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied [match_results](#) structure.

Definition at line 85 of file regex_constants.h.

4.19.5.23 const syntax_option_type std::regex_constants::optimize [static]

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

Definition at line 93 of file regex_constants.h.

4.20 std::rel_ops Namespace Reference

The generated relational operators are sequestered here.

Functions

- template<class _Tp >
bool [operator!=](#) (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
bool [operator<=](#) (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
bool [operator>](#) (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
bool [operator>=](#) (const _Tp &__x, const _Tp &__y)

4.20.1 Detailed Description

The generated relational operators are sequestered here.

4.20.2 Function Documentation

4.20.2.1 `template<class _Tp > bool std::rel_ops::operator!= (const _Tp & __x, const _Tp & __y) [inline]`

Defines != for arbitrary types, in terms of ==.

Parameters

x A thing.

y Another thing.

Returns

$x \neq y$

This function uses == to determine its result.

Definition at line 85 of file stl_relops.h.

4.20.2.2 `template<class _Tp > bool std::rel_ops::operator<= (const _Tp & __x, const _Tp & __y) [inline]`

Defines <= for arbitrary types, in terms of <.

Parameters

x A thing.

y Another thing.

Returns

$x \leq y$

This function uses < to determine its result.

Definition at line 111 of file stl_relops.h.

4.20.2.3 `template<class _Tp > bool std::rel_ops::operator> (const _Tp & __x, const _Tp & __y) [inline]`

Defines > for arbitrary types, in terms of <.

Parameters

x A thing.

`y` Another thing.

Returns

`x > y`

This function uses `<` to determine its result.

Definition at line 98 of file `std_relops.h`.

4.20.2.4 `template<class _Tp> bool std::rel_ops::operator>= (const _Tp & __x, const _Tp & __y) [inline]`

Defines `>=` for arbitrary types, in terms of `<`.

Parameters

`x` A thing.

`y` Another thing.

Returns

`x >= y`

This function uses `<` to determine its result.

Definition at line 124 of file `std_relops.h`.

4.21 `std::this_thread` Namespace Reference

ISO C++ 0x entities sub namespace for thread. 30.2.2 Namespace [this_thread](#).

Functions

- [thread::id](#) `get_id ()`
- `template<typename _Rep, typename _Period>`
void [sleep_for](#) (const [chrono::duration](#)< _Rep, _Period > &__rtime)
- `template<typename _Clock, typename _Duration>`
void [sleep_until](#) (const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- void [yield](#) ()

4.21.1 Detailed Description

ISO C++ 0x entities sub namespace for thread. 30.2.2 Namespace [this_thread](#).

4.21.2 Function Documentation

4.21.2.1 `thread::id` `std::this_thread::get_id ()` `[inline]`

`get_id`

Definition at line 247 of file `thread`.

4.21.2.2 `template<typename _Rep , typename _Period > void` `std::this_thread::sleep_for (const chrono::duration< _Rep, _Period` `> & __ptime)` `[inline]`

`sleep_for`

Definition at line 266 of file `thread`.

References `std::chrono::duration_cast()`.

Referenced by `sleep_until()`.

4.21.2.3 `template<typename _Clock , typename _Duration > void` `std::this_thread::sleep_until (const chrono::time_point< _Clock,` `_Duration > & __ptime)` `[inline]`

`sleep_until`

Definition at line 260 of file `thread`.

References `sleep_for()`.

4.21.2.4 `void std::this_thread::yield ()` `[inline]`

`yield`

Definition at line 252 of file `thread`.

4.22 `std::tr1` Namespace Reference

ISO C++ TR1 entities toplevel namespace is [std::tr1](#).

Namespaces

- namespace [__detail](#)

Classes

- struct [__is_member_pointer_helper](#)
is_member_pointer
- struct [add_const](#)
add_const
- struct [add_cv](#)
add_cv
- struct [add_pointer](#)
add_pointer
- struct [add_volatile](#)
add_volatile
- struct [alignment_of](#)
alignment_of
- struct [array](#)
A standard container for storing a fixed size sequence of elements.
- class [bad_weak_ptr](#)
Exception possibly thrown by [shared_ptr](#).
- struct [extent](#)
extent
- struct [has_virtual_destructor](#)
has_virtual_destructor
- struct [integral_constant](#)
integral_constant
- struct [is_abstract](#)
is_abstract
- struct [is_arithmetic](#)
is_arithmetic
- struct [is_array](#)
is_array

- struct [is_class](#)
is_class
- struct [is_compound](#)
is_compound
- struct [is_const](#)
is_const
- struct [is_empty](#)
is_empty
- struct [is_enum](#)
is_enum
- struct [is_floating_point](#)
is_floating_point
- struct [is_function](#)
is_function
- struct [is_fundamental](#)
is_fundamental
- struct [is_integral](#)
is_integral
- struct [is_member_function_pointer](#)
is_member_function_pointer
- struct [is_member_object_pointer](#)
is_member_object_pointer
- struct [is_object](#)
is_object
- struct [is_pointer](#)
is_pointer
- struct [is_polymorphic](#)
is_polymorphic

- struct [is_same](#)
is_same
- struct [is_scalar](#)
is_scalar
- struct [is_union](#)
is_union
- struct [is_void](#)
is_void
- struct [is_volatile](#)
is_volatile
- struct [rank](#)
rank
- struct [remove_all_extents](#)
remove_all_extents
- struct [remove_const](#)
remove_const
- struct [remove_cv](#)
remove_cv
- struct [remove_extent](#)
remove_extent
- struct [remove_pointer](#)
remove_pointer
- struct [remove_volatile](#)
remove_volatile

Typedefs

- typedef [integral_constant](#)< bool, false > [false_type](#)
- typedef [integral_constant](#)< bool, true > [true_type](#)

Functions

- `template<typename _Tp >`
`std::complex< _Tp > __complex_acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > __complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > __complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > __complex_asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > __complex_atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > __complex_atanh (const std::complex< _Tp > &__z)`
- `void __throw_bad_weak_ptr ()`
- `template<typename _Tp >`
`std::complex< _Tp > acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type arg (_Tp __x)`
- `template<typename _Tp >`
`std::complex< _Tp > asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tp >`
`std::complex< _Tp > atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< _Tp > atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tpx, typename _Tpy >`
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type beta (_Tpx __x, _Tpy __y)`
- `float betaf (float __x, float __y)`

- long double **betal** (long double __x, long double __y)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **comp_ellint_1** (_Tp __k)
- float **comp_ellint_1f** (float __k)
- long double **comp_ellint_1l** (long double __k)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **comp_ellint_2** (_Tp __k)
- float **comp_ellint_2f** (float __k)
- long double **comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn >
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type **comp_ellint_3** (_Tp __k, _Tpn __nu)
- float **comp_ellint_3f** (float __k, float __nu)
- long double **comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp >
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type **conf_hyperg** (_Tpa __a, _Tpc __c, _Tp __x)
- float **conf_hypergf** (float __a, float __c, float __x)
- long double **conf_hypergl** (long double __a, long double __c, long double __x)
- template<typename _Tp >
std::complex< _Tp > **conj** (const **std::complex**< _Tp > &__z)
- template<typename _Tp >
std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **conj** (_Tp __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **cyl_bessel_i** (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_if** (float __nu, float __x)
- long double **cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **cyl_bessel_j** (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_jf** (float __nu, float __x)
- long double **cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **cyl_bessel_k** (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_kf** (float __nu, float __x)
- long double **cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **cyl_neumann** (_Tpnu __nu, _Tp __x)
- float **cyl_neumannf** (float __nu, float __x)
- long double **cyl_neumannl** (long double __nu, long double __x)

- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 (_Tp __k, _Tpp __-`
`phi)`
- `float ellint_1f (float __k, float __phi)`
- `long double ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 (_Tp __k, _Tpp __-`
`phi)`
- `float ellint_2f (float __k, float __phi)`
- `long double ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp >`
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint_3 (_Tp __k, _Tpn`
`__nu, _Tpp __phi)`
- `float ellint_3f (float __k, float __nu, float __phi)`
- `long double ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)`
- `float expintf (float __x)`
- `long double expintl (long double __x)`
- `template<typename _Tp >`
`_Tp fabs (const std::complex< _Tp > &__z)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`
`_Tp & get (array< _Tp, _Nm > &__arr)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`
`const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & get (const`
`std::pair< _Tp1, _Tp2 > &__in)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >`
`tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & get (std::pair< _Tp1,`
`_Tp2 > &__in)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`
`const _Tp & get (const array< _Tp, _Nm > &__arr)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)`
- `float hermitef (unsigned int __n, float __x)`
- `long double hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __-`
`a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float hypergf (float __a, float __b, float __c, float __x)`
- `long double hypergl (long double __a, long double __b, long double __c, long`
`double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type imag (_Tp)`

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type` `laguerre` (unsigned int __n, _Tp __x)
- `float` `laguerref` (unsigned int __n, float __x)
- `long double` `laguerrel` (unsigned int __n, long double __x)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type` `legendre` (unsigned int __n, _Tp __x)
- `float` `legendref` (unsigned int __n, float __x)
- `long double` `legendrel` (unsigned int __n, long double __x)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type` `norm` (_Tp __x)
- `template<typename _Tp, std::size_t _Nm>`
`bool` `operator!=` (const `array`< _Tp, _Nm > &__one, const `array`< _Tp, _Nm > &__two)
- `template<typename _Tp, std::size_t _Nm>`
`bool` `operator`< (const `array`< _Tp, _Nm > &__a, const `array`< _Tp, _Nm > &__b)
- `template<typename _Tp, std::size_t _Nm>`
`bool` `operator`<= (const `array`< _Tp, _Nm > &__one, const `array`< _Tp, _Nm > &__two)
- `template<typename _Tp, std::size_t _Nm>`
`bool` `operator`== (const `array`< _Tp, _Nm > &__one, const `array`< _Tp, _Nm > &__two)
- `template<typename _Tp, std::size_t _Nm>`
`bool` `operator`> (const `array`< _Tp, _Nm > &__one, const `array`< _Tp, _Nm > &__two)
- `template<typename _Tp, std::size_t _Nm>`
`bool` `operator`>= (const `array`< _Tp, _Nm > &__one, const `array`< _Tp, _Nm > &__two)
- `template<typename _Tp, typename _Up >`
`std::complex`< typename `__gnu_cxx::__promote_2`< _Tp, _Up >::__type > `polar` (const _Tp &__rho, const _Up &__theta)
- `template<typename _Tp, typename _Up >`
`std::complex`< typename `__gnu_cxx::__promote_2`< _Tp, _Up >::__type > `pow` (const _Tp &__x, const `std::complex`< _Up > &__y)
- `template<typename _Tp, typename _Up >`
`__gnu_cxx::__promote_2`< _Tp, _Up >::__type `pow` (_Tp __x, _Up __y)
- `long double` `pow` (long double __x, long double __y)
- `double` `pow` (double __x, double __y)
- `template<typename _Tp >`
`std::complex`< _Tp > `pow` (const `std::complex`< _Tp > &__x, const _Tp &__y)
- `template<typename _Tp >`
`std::complex`< _Tp > `pow` (const _Tp &__x, const `std::complex`< _Tp > &__y)

- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp >`
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const`
`std::complex< _Tp > &__y)`
- `float pow (float __x, float __y)`
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`
`pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type real (_Tp __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)`
- `float riemann_zetaf (float __x)`
- `long double riemann_zetal (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)`
- `float sph_besself (unsigned int __n, float __x)`
- `long double sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, un-`
`signed int __m, _Tp __theta)`
- `float sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double sph_legendrel (unsigned int __l, unsigned int __m, long double`
`__theta)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp`
`__x)`
- `float sph_neumannf (unsigned int __n, float __x)`
- `long double sph_neumannl (unsigned int __n, long double __x)`
- `template<typename _Tp, std::size_t _Nm>`
`void swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two)`

4.22.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is [std::tr1](#).

4.23 std::tr1::__detail Namespace Reference

Implementation details not part of the namespace [std::tr1](#) interface.

4.23.1 Detailed Description

Implementation details not part of the namespace [std::tr1](#) interface.

Chapter 5

Class Documentation

5.1 `__atomic0::atomic_address` Struct Reference

29.4.2, address types

Public Member Functions

- **atomic_address** (void *__v)
- **atomic_address** (const [atomic_address](#) &)
- bool **compare_exchange_strong** (void *&__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_strong** (void *&__v1, void *__v2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_weak** (void *&__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_weak** (void *&__v1, void *__v2, memory_order __m=memory_order_seq_cst)
- void * **exchange** (void *__v, memory_order __m=memory_order_seq_cst)
- void * **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst)
- void * **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst)
- bool **is_lock_free** () const
- void * **load** (memory_order __m=memory_order_seq_cst) const
- **operator void *** () const
- void * **operator+=** (ptrdiff_t __d)
- void * **operator-=** (ptrdiff_t __d)
- [atomic_address](#) & **operator=** (const [atomic_address](#) &) volatile
- void * **operator=** (void *__v)
- void **store** (void *__v, memory_order __m=memory_order_seq_cst)

5.1.1 Detailed Description

29.4.2, address types

Definition at line 103 of file atomic_0.h.

The documentation for this struct was generated from the following file:

- [atomic_0.h](#)

5.2 __atomic0::atomic_bool Struct Reference

[atomic_bool](#)

Public Member Functions

- **atomic_bool** (bool __i)
- **atomic_bool** (const [atomic_bool](#) &)
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst)
- bool **is_lock_free** () const
- bool **load** (memory_order __m=memory_order_seq_cst) const
- **operator bool** () const
- [atomic_bool](#) & **operator=** (const [atomic_bool](#) &) volatile
- bool **operator=** (bool __i)
- void **store** (bool __i, memory_order __m=memory_order_seq_cst)

5.2.1 Detailed Description

[atomic_bool](#)

Definition at line 391 of file atomic_0.h.

The documentation for this struct was generated from the following file:

- [atomic_0.h](#)

5.3 `__atomic0::atomic_flag` Struct Reference

[atomic_flag](#)

Inherits `__atomic_flag_base`.

Public Member Functions

- **atomic_flag** (bool __i)
- **atomic_flag** (const [atomic_flag](#) &)
- void **clear** (memory_order __m=memory_order_seq_cst)
- [atomic_flag](#) & **operator=** (const [atomic_flag](#) &) volatile
- bool **test_and_set** (memory_order __m=memory_order_seq_cst)

5.3.1 Detailed Description

[atomic_flag](#)

Definition at line 85 of file `atomic_0.h`.

The documentation for this struct was generated from the following file:

- [atomic_0.h](#)

5.4 `__atomic2::atomic_address` Struct Reference

29.4.2, address types

Public Member Functions

- **atomic_address** (void *__v)
- **atomic_address** (const [atomic_address](#) &)
- bool **compare_exchange_strong** (void *&__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_strong** (void *&__v1, void *__v2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_weak** (void *&__v1, void *__v2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_weak** (void *&__v1, void *__v2, memory_order __m=memory_order_seq_cst)
- void * **exchange** (void *__v, memory_order __m=memory_order_seq_cst)
- void * **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst)

- void * **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst)
- bool **is_lock_free** () const
- void * **load** (memory_order __m=memory_order_seq_cst) const
- **operator void** * () const
- void * **operator+=** (ptrdiff_t __d)
- void * **operator-=** (ptrdiff_t __d)
- [atomic_address](#) & **operator=** (const [atomic_address](#) &) volatile
- void * **operator=** (void *__v)
- void **store** (void *__v, memory_order __m=memory_order_seq_cst)

5.4.1 Detailed Description

29.4.2, address types

Definition at line 81 of file atomic_2.h.

The documentation for this struct was generated from the following file:

- [atomic_2.h](#)

5.5 __atomic2::atomic_bool Struct Reference

[atomic_bool](#)

Public Member Functions

- **atomic_bool** (bool __i)
- **atomic_bool** (const [atomic_bool](#) &)
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst)
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2)
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst)
- bool **is_lock_free** () const
- bool **load** (memory_order __m=memory_order_seq_cst) const
- **operator bool** () const
- [atomic_bool](#) & **operator=** (const [atomic_bool](#) &) volatile
- bool **operator=** (bool __i)
- void **store** (bool __i, memory_order __m=memory_order_seq_cst)

5.5.1 Detailed Description

[atomic_bool](#)

Definition at line 391 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic_2.h](#)

5.6 `__atomic2::atomic_flag` Struct Reference

[atomic_flag](#)

Inherits `__atomic_flag_base`.

Public Member Functions

- `atomic_flag` (`bool __i`)
- `atomic_flag` (`const atomic_flag &`)
- `atomic_flag` & `operator=` (`const atomic_flag &`) volatile

5.6.1 Detailed Description

[atomic_flag](#)

Definition at line 47 of file `atomic_2.h`.

The documentation for this struct was generated from the following file:

- [atomic_2.h](#)

5.7 `__cxxabiv1::__forced_unwind` Class Reference

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

5.7.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

Definition at line 47 of file cxxabi-forced.h.

The documentation for this class was generated from the following file:

- [cxxabi-forced.h](#)

5.8 `__gnu_cxx::__common_pool_policy< _PoolTp, _Thread >` Struct Template Reference

Policy for shared `__pool` objects.

Inherits `__common_pool_base< _PoolTp, _Thread >`.

5.8.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread> struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >
```

Policy for shared `__pool` objects.

Definition at line 456 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

5.9 `__gnu_cxx::detail::__mini_vector< _Tp >` Class Template Reference

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

Public Types

- typedef const `_Tp` & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef pointer **iterator**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef size_t **size_type**
- typedef `_Tp` **value_type**

5.10 `__gnu_cxx::__detail::_Bitmap_counter< _Tp >` Class Template Reference

Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator __pos) throw ()
- void **insert** (iterator __pos, const_reference __x)
- reference **operator[]** (const size_type __pos) const throw ()
- void **pop_back** () throw ()
- void **push_back** (const_reference __x)
- size_type **size** () const throw ()

5.9.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::__mini_vector< _Tp >`

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`. It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses `operator new()` to get memory, and `operator delete()` to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

Definition at line 70 of file `bitmap_allocator.h`.

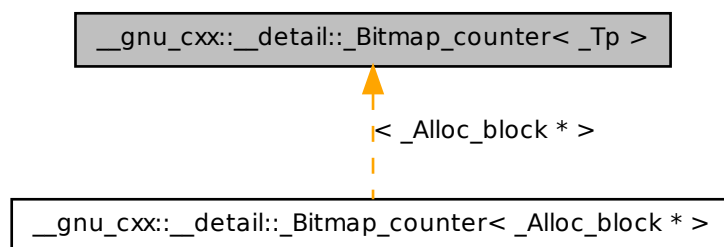
The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

5.10 `__gnu_cxx::__detail::_Bitmap_counter< _Tp >` Class Template Reference

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

Inheritance diagram for `__gnu_cxx::__detail::_Bitmap_counter<_Tp>`:



Public Member Functions

- `_Bitmap_counter` ([_BPVector](#) &Rvbp, long __index=-1)
- `pointer _M_base` () const throw ()
- `bool _M_finished` () const throw ()
- `size_t * _M_get` () const throw ()
- `_Index_type _M_offset` () const throw ()
- `void _M_reset` (long __index=-1) throw ()
- `void _M_set_internal_bitmap` (size_t *__new_internal_marker) throw ()
- `_Index_type _M_where` () const throw ()
- `_Bitmap_counter` & `operator++` () throw ()

5.10.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__detail::_Bitmap_counter<_Tp>`

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

Definition at line 399 of file `bitmap_allocator.h`.

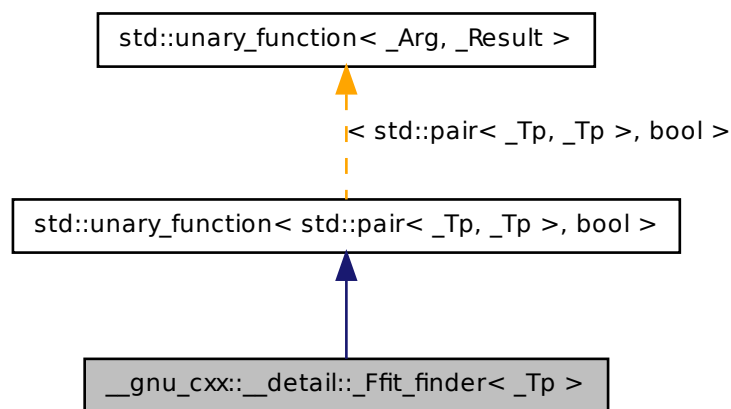
The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

5.11 `__gnu_cxx::__detail::_Ffit_finder< _Tp >` Class Template Reference

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Inheritance diagram for `__gnu_cxx::__detail::_Ffit_finder< _Tp >`:



Public Types

- typedef `std::pair< _Tp, _Tp >` `argument_type`
- typedef `bool` `result_type`

Public Member Functions

- `size_t * _M_get () const throw ()`
- `_Counter_type _M_offset () const throw ()`
- `bool operator() (_Block_pair __bp) throw ()`

5.11.1 Detailed Description

template<typename _Tp> class __gnu_cxx::__detail::_Ffit_finder< _Tp >

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 334 of file `bitmap_allocator.h`.

5.11.2 Member Typedef Documentation

5.11.2.1 typedef std::pair< _Tp, _Tp > std::unary_function< std::pair< _Tp, _Tp >, bool >::argument_type [inherited]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.11.2.2 typedef bool std::unary_function< std::pair< _Tp, _Tp >, bool >::result_type [inherited]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

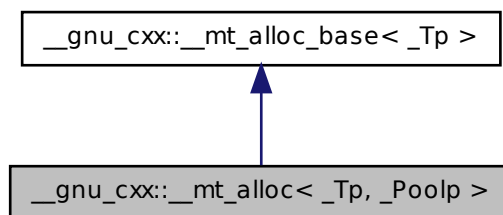
- [bitmap_allocator.h](#)

5.12 __gnu_cxx::__mt_alloc< _Tp, _Poolp > Class Template Reference

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



Public Types

- `typedef _Poolp __policy_type`
- `typedef _Poolp::pool_type __pool_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `__mt_alloc` (const `__mt_alloc` &) throw ()
- `template<typename _Tp1, typename _Poolp1>`
`__mt_alloc` (const `__mt_alloc<_Tp1, _Poolp1>` &) throw ()
- `const __pool_base::_Tune _M_get_options` ()
- `void _M_set_options` (__pool_base::_Tune __t)
- `pointer address` (reference __x) const
- `const_pointer address` (const_reference __x) const
- `pointer allocate` (size_type __n, const void *=0)
- `template<typename... _Args>`
`void construct` (pointer __p, _Args &&...__args)
- `void construct` (pointer __p, const _Tp &__val)
- `void deallocate` (pointer __p, size_type __n)

- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

5.12.1 Detailed Description

template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >> class __gnu_cxx::__mt_alloc< _Tp, _Poolp >

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

Definition at line 625 of file mt_allocator.h.

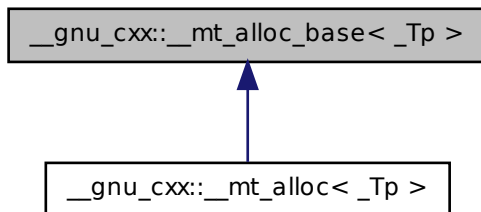
The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

5.13 __gnu_cxx::__mt_alloc_base< _Tp > Class Template Reference

Base class for _Tp dependent member functions.

Inheritance diagram for __gnu_cxx::__mt_alloc_base< _Tp >:



Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `pointer address (reference __x) const`
- `const_pointer address (const_reference __x) const`
- `template<typename... _Args>
void construct (pointer __p, _Args &&...__args)`
- `void construct (pointer __p, const _Tp &__val)`
- `void destroy (pointer __p)`
- `size_type max_size () const throw ()`

5.13.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__mt_alloc_base< _Tp >`

Base class for `_Tp` dependent member functions.

Definition at line 566 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

5.14 `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct` Template Reference

Policy for individual `__pool` objects.

Inherits `__per_type_pool_base< _Tp, _PoolTp, _Thread >`.

5.14.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread> struct
__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >
```

Policy for individual __pool objects.

Definition at line 551 of file mt_allocator.h.

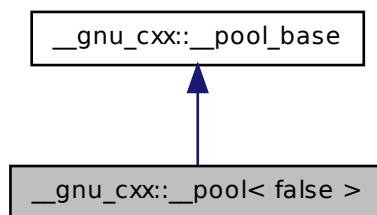
The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

5.15 __gnu_cxx::__pool< false > Class Template Reference

Specialization for single thread.

Inheritance diagram for __gnu_cxx::__pool< false >:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool** (const __pool_base::Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &, _Block_record *, size_t)

- `bool _M_check_threshold (size_t __bytes)`
- `void _M_destroy () throw ()`
- `size_t _M_get_align ()`
- `const _Bin_record & _M_get_bin (size_t __which)`
- `size_t _M_get_binmap (size_t __bytes)`
- `const _Tune & _M_get_options () const`
- `size_t _M_get_thread_id ()`
- `void _M_initialize_once ()`
- `void _M_reclaim_block (char *__p, size_t __bytes) throw ()`
- `char * _M_reserve_block (size_t __bytes, const size_t __thread_id)`
- `void _M_set_options (_Tune __t)`

Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

5.15.1 Detailed Description

`template<> class __gnu_cxx::__pool< false >`

Specialization for single thread.

Definition at line 192 of file `mt_allocator.h`.

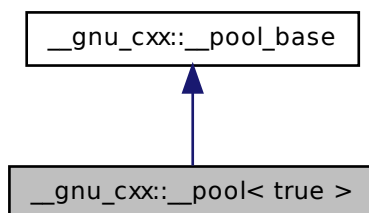
The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

5.16 `__gnu_cxx::__pool< true >` Class Template Reference

Specialization for thread enabled, via `gthreads.h`.

Inheritance diagram for `__gnu_cxx::__pool< true >`:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool** (const __pool_base::Tune &__tune)
- **__attribute__** ((__const__)) void **_M_destroy_thread_key**(void *) throw ()
- void **_M_adjust_freelist** (const _Bin_record &__bin, _Block_record *__block, size_t __thread_id)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize** (__destroy_handler)
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char *__p, size_t __bytes) throw ()
- char * **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

5.16.1 Detailed Description

`template<> class __gnu_cxx::__pool< true >`

Specialization for thread enabled, via `gthreads.h`.

Definition at line 259 of file `mt_allocator.h`.

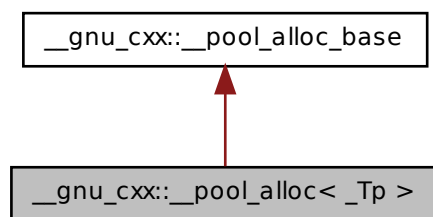
The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

5.17 `__gnu_cxx::__pool_alloc<_Tp>` Class Template Reference

Allocator using a memory pool with a single lock.

Inheritance diagram for `__gnu_cxx::__pool_alloc<_Tp>`:



Public Types

- `typedef const _Tp * const_pointer`

- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **__pool_alloc** (const [__pool_alloc](#) &) throw ()
- template<typename _Tp1 >
 __pool_alloc (const [__pool_alloc](#)< _Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, const void * = 0)
- template<typename... _Args>
 void **construct** (pointer __p, _Args &&... __args)
- void **construct** (pointer __p, const _Tp & __val)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

Private Types

- enum { **_S_align** }
- enum { **_S_max_bytes** }
- enum { **_S_free_list_size** }

Private Member Functions

- **__attribute__** ((__const__)) _Obj *volatile *_M_get_free_list(size_t __bytes) throw ()
- char * **_M_allocate_chunk** (size_t __n, int & __nobjs)
- __mutex & **_M_get_mutex** () throw ()
- void * **_M_refill** (size_t __n)
- size_t **_M_round_up** (size_t __bytes)

Static Private Attributes

- static char * **_S_end_free**
- static _Obj *volatile **_S_free_list** [_S_free_list_size]
- static size_t **_S_heap_size**
- static char * **_S_start_free**

5.17.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::__pool_alloc< _Tp >`

Allocator using a memory pool with a single lock.

Definition at line 122 of file `pool_allocator.h`.

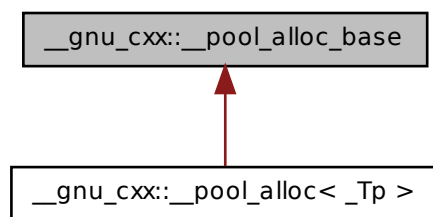
The documentation for this class was generated from the following file:

- [pool_allocator.h](#)

5.18 __gnu_cxx::__pool_alloc_base Class Reference

Base class for [__pool_alloc](#).

Inheritance diagram for `__gnu_cxx::__pool_alloc_base`:



Protected Types

- enum { `_S_align` }
- enum { `_S_max_bytes` }
- enum { `_S_free_list_size` }

Protected Member Functions

- `__attribute__((__const__)) _Obj *volatile *_M_get_free_list(size_t __bytes) throw ()`

- `char * _M_allocate_chunk (size_t __n, int &__nobjs)`
- `__mutex & _M_get_mutex () throw ()`
- `void * _M_refill (size_t __n)`
- `size_t _M_round_up (size_t __bytes)`

Static Protected Attributes

- `static char * _S_end_free`
- `static _Obj *volatile _S_free_list [_S_free_list_size]`
- `static size_t _S_heap_size`
- `static char * _S_start_free`

5.18.1 Detailed Description

Base class for [__pool_alloc](#). Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size > `_S_max_bytes`, the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly `_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

Definition at line 74 of file `pool_allocator.h`.

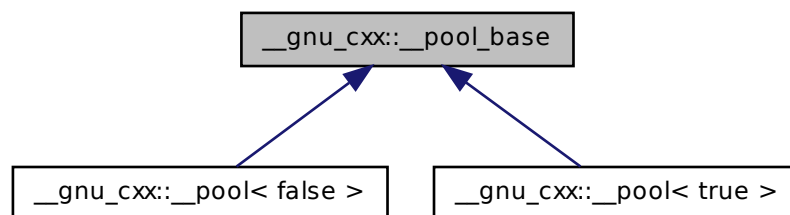
The documentation for this class was generated from the following file:

- [pool_allocator.h](#)

5.19 `__gnu_cxx::__pool_base` Struct Reference

Base class for pool object.

Inheritance diagram for `__gnu_cxx::__pool_base`:



Public Types

- typedef unsigned short int **`_Binmap_type`**

Public Member Functions

- **`__pool_base`** (const `_Tune` & `__options`)
- **`_M_check_threshold`** (size_t `__bytes`)
- **`_M_get_align`** ()
- **`_M_get_binmap`** (size_t `__bytes`)
- **`_M_get_options`** () const
- **`_M_set_options`** (_Tune `__t`)

Protected Attributes

- `_Binmap_type` * **`_M_binmap`**
- bool **`_M_init`**
- `_Tune` **`_M_options`**

5.19.1 Detailed Description

Base class for pool object.

Definition at line 47 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

5.20 `__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::__vstring_utility< _CharT, _Traits, _Alloc >`.

Public Types

- `typedef _Util_Base::_CharT_alloc_type _CharT_alloc_type`
- `typedef __vstring_utility< _CharT, _Traits, _Alloc > _Util_Base`
- `typedef _Alloc allocator_type`
- `typedef _CharT_alloc_type::size_type size_type`
- `typedef _Traits traits_type`
- `typedef _Traits::char_type value_type`

Public Member Functions

- `__rc_string_base (const _Alloc &__a)`
- `__rc_string_base (const __rc_string_base &__rcs)`
- `__rc_string_base (__rc_string_base &&__rcs)`
- `__rc_string_base (size_type __n, _CharT __c, const _Alloc &__a)`
- `template<typename _InputIterator >
__rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)`
- `void _M_assign (const __rc_string_base &__rcs)`
- `size_type _M_capacity () const`
- `void _M_clear ()`
- `bool _M_compare (const __rc_string_base &) const`
- `template<>
bool _M_compare (const __rc_string_base &__rcs) const`
- `template<>
bool _M_compare (const __rc_string_base &__rcs) const`
- `_CharT * _M_data () const`
- `void _M_erase (size_type __pos, size_type __n)`
- `allocator_type & _M_get_allocator ()`
- `const allocator_type & _M_get_allocator () const`
- `bool _M_is_shared () const`
- `void _M_leak ()`
- `size_type _M_length () const`

- `size_type _M_max_size () const`
- `void _M_mutate (size_type __pos, size_type __len1, const _CharT *__s, size_type __len2)`
- `void _M_reserve (size_type __res)`
- `void _M_set_leaked ()`
- `void _M_set_length (size_type __n)`
- `void _M_swap (__rc_string_base &__rcs)`
- `template<typename _InIterator >
_CharT * _S_construct (_InIterator __beg, _InIterator __end, const _Alloc &__a, std::forward_iterator_tag)`

Protected Types

- `typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __rc_string_base> > > __const_rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __sso_string_base> > > __const_sso_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __rc_string_base> > > __rc_iterator`
- `typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, __sso_string_base> > > __sso_iterator`
- `typedef _CharT_alloc_type::const_pointer const_pointer`
- `typedef _CharT_alloc_type::difference_type difference_type`
- `typedef _CharT_alloc_type::pointer pointer`

Static Protected Member Functions

- `static void _S_assign (_CharT *__d, size_type __n, _CharT __c)`
- `static int _S_compare (size_type __n1, size_type __n2)`
- `static void _S_copy (_CharT *__d, const _CharT *__s, size_type __n)`
- `static void _S_copy_chars (_CharT *__p, _CharT *__k1, _CharT *__k2)`
- `static void _S_copy_chars (_CharT *__p, __sso_iterator __k1, __sso_iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __rc_iterator __k1, __rc_iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __const_sso_iterator __k1, __const_sso_iterator __k2)`
- `template<typename _Iterator >
static void _S_copy_chars (_CharT *__p, _Iterator __k1, _Iterator __k2)`
- `static void _S_copy_chars (_CharT *__p, __const_rc_iterator __k1, __const_rc_iterator __k2)`

- static void `_S_copy_chars` (`_CharT *__p`, const `_CharT *__k1`, const `_CharT *__k2`)
- static void `_S_move` (`_CharT *__d`, const `_CharT *__s`, `size_type __n`)

5.20.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >`

Documentation? What's that? Nathan Myers <ncm@cantrip.org>.

A string looks like this:

	<code>[_Rep]</code>
	<code>_M_length</code>
<code>[_rc_string_base<char_type>]</code>	<code>_M_capacity</code>
<code>_M_dataplus</code>	<code>_M_refcount</code>
<code>_M_p -----></code>	unnamed array of <code>char_type</code>

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 82 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc_string_base.h](#)

5.21 `__gnu_cxx::__scoped_lock` Class Reference

Scoped lock idiom.

Public Types

- typedef `__mutex` `__mutex_type`

Public Member Functions

- `__scoped_lock` (`__mutex_type` & `__name`)

5.21.1 Detailed Description

Scoped lock idiom.

Definition at line 243 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

5.22 `__gnu_cxx::__versa_string< _CharT, _Traits, _-Alloc, _Base >` Class Template Reference

Template class [__versa_string](#).

Data structure managing sequences of characters and character-like objects.

Inherits `_Base< _CharT, _Traits, _Alloc >`.

Public Types

- typedef `_Alloc` `allocator_type`
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` `const_iterator`
- typedef `_CharT_alloc_type::const_pointer` `const_pointer`
- typedef `const value_type &` `const_reference`
- typedef `std::reverse_iterator< const_iterator >` `const_reverse_iterator`
- typedef `_CharT_alloc_type::difference_type` `difference_type`
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` `iterator`
- typedef `_CharT_alloc_type::pointer` `pointer`

- typedef value_type & **reference**
- typedef [std::reverse_iterator](#)< iterator > **reverse_iterator**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- [__versa_string](#) ()
- [__versa_string](#) (const _Alloc &__a)
- [__versa_string](#) ([__versa_string](#) &&__str)
- [__versa_string](#) (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())
- [__versa_string](#) (const _CharT *__s, const _Alloc &__a=_Alloc())
- [__versa_string](#) ([std::initializer_list](#)< _CharT > __l, const _Alloc &__a=_Alloc())
- [__versa_string](#) (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- template<class _InputIterator >
 [__versa_string](#) (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- [__versa_string](#) (const [__versa_string](#) &__str)
- [__versa_string](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n=npos)
- [__versa_string](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n, const _Alloc &__a)
- [~__versa_string](#) ()
- [__versa_string](#) & [append](#) (const [__versa_string](#) &__str)
- [__versa_string](#) & [append](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n)
- [__versa_string](#) & [append](#) (const _CharT *__s, size_type __n)
- [__versa_string](#) & [append](#) (const _CharT *__s)
- [__versa_string](#) & [append](#) (size_type __n, _CharT __c)
- [__versa_string](#) & [append](#) ([std::initializer_list](#)< _CharT > __l)
- template<class _InputIterator >
 [__versa_string](#) & [append](#) (_InputIterator __first, _InputIterator __last)
- [__versa_string](#) & [assign](#) (const _CharT *__s)
- [__versa_string](#) & [assign](#) (size_type __n, _CharT __c)
- template<class _InputIterator >
 [__versa_string](#) & [assign](#) (_InputIterator __first, _InputIterator __last)
- [__versa_string](#) & [assign](#) ([std::initializer_list](#)< _CharT > __l)
- [__versa_string](#) & [assign](#) (const [__versa_string](#) &__str)
- [__versa_string](#) & [assign](#) ([__versa_string](#) &&__str)
- [__versa_string](#) & [assign](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n)

- `__versa_string` & `assign` (const `_CharT` *`__s`, `size_type` `__n`)
- `const_reference` `at` (`size_type` `__n`) const
- `reference` `at` (`size_type` `__n`)
- `reference` `back` ()
- `const_reference` `back` () const
- `iterator` `begin` ()
- `const_iterator` `begin` () const
- `const _CharT` * `c_str` () const
- `size_type` `capacity` () const
- `const_iterator` `cbegin` () const
- `const_iterator` `cend` () const
- `void` `clear` ()
- `int` `compare` (const `_CharT` *`__s`) const
- `int` `compare` (`size_type` `__pos`, `size_type` `__n1`, const `_CharT` *`__s`) const
- `int` `compare` (`size_type` `__pos`, `size_type` `__n1`, const `_CharT` *`__s`, `size_type` `__n2`) const
- `int` `compare` (`size_type` `__pos1`, `size_type` `__n1`, const `__versa_string` &`__str`, `size_type` `__pos2`, `size_type` `__n2`) const
- `int` `compare` (`size_type` `__pos`, `size_type` `__n`, const `__versa_string` &`__str`) const
- `int` `compare` (const `__versa_string` &`__str`) const
- `size_type` `copy` (`_CharT` *`__s`, `size_type` `__n`, `size_type` `__pos=0`) const
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- `const _CharT` * `data` () const
- `bool` `empty` () const
- `iterator` `end` ()
- `const_iterator` `end` () const
- `__versa_string` & `erase` (`size_type` `__pos=0`, `size_type` `__n=npos`)
- `iterator` `erase` (`iterator` `__position`)
- `iterator` `erase` (`iterator` `__first`, `iterator` `__last`)
- `size_type` `find` (`_CharT` `__c`, `size_type` `__pos=0`) const
- `size_type` `find` (const `_CharT` *`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type` `find` (const `__versa_string` &`__str`, `size_type` `__pos=0`) const
- `size_type` `find` (const `_CharT` *`__s`, `size_type` `__pos=0`) const
- `size_type` `find_first_not_of` (const `_CharT` *`__s`, `size_type` `__pos`, `size_type` `__n`) const
- `size_type` `find_first_not_of` (`_CharT` `__c`, `size_type` `__pos=0`) const
- `size_type` `find_first_not_of` (const `__versa_string` &`__str`, `size_type` `__pos=0`) const
- `size_type` `find_first_not_of` (const `_CharT` *`__s`, `size_type` `__pos=0`) const
- `size_type` `find_first_of` (`_CharT` `__c`, `size_type` `__pos=0`) const
- `size_type` `find_first_of` (const `_CharT` *`__s`, `size_type` `__pos`, `size_type` `__n`) const

- size_type [find_first_of](#) (const _CharT *__s, size_type __pos=0) const
- size_type [find_first_of](#) (const __versa_string &__str, size_type __pos=0) const
- size_type [find_last_not_of](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type [find_last_not_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find_last_not_of](#) (const __versa_string &__str, size_type __pos=[npos](#)) const
- size_type [find_last_not_of](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const __versa_string &__str, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const
- void [insert](#) (iterator __p, size_type __n, _CharT __c)
- template<class _InputIterator >
void [insert](#) (iterator __p, _InputIterator __beg, _InputIterator __end)
- void [insert](#) (iterator __p, [std::initializer_list](#)< _CharT > __l)
- __versa_string & [insert](#) (size_type __pos1, const __versa_string &__str, size_type __pos2, size_type __n)
- __versa_string & [insert](#) (size_type __pos, const _CharT *__s, size_type __n)
- __versa_string & [insert](#) (size_type __pos, const _CharT *__s)
- iterator [insert](#) (iterator __p, _CharT __c)
- __versa_string & [insert](#) (size_type __pos, size_type __n, _CharT __c)
- __versa_string & [insert](#) (size_type __pos1, const __versa_string &__str)
- size_type [length](#) () const
- size_type [max_size](#) () const
- __versa_string & [operator+=](#) ([std::initializer_list](#)< _CharT > __l)
- __versa_string & [operator+=](#) (const __versa_string &__str)
- __versa_string & [operator+=](#) (_CharT __c)
- __versa_string & [operator+=](#) (const _CharT *__s)
- __versa_string & [operator=](#) (__versa_string &&__str)
- __versa_string & [operator=](#) (const _CharT *__s)
- __versa_string & [operator=](#) (_CharT __c)
- __versa_string & [operator=](#) ([std::initializer_list](#)< _CharT > __l)
- __versa_string & [operator=](#) (const __versa_string &__str)
- const_reference [operator\[\]](#) (size_type __pos) const
- reference [operator\[\]](#) (size_type __pos)
- void [push_back](#) (_CharT __c)
- reverse_iterator [rbegin](#) ()

- `const_reverse_iterator rbegin` () const
- `reverse_iterator rend` ()
- `const_reverse_iterator rend` () const
- `__versa_string & replace` (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- `__versa_string & replace` (iterator __i1, iterator __i2, const __versa_string &__str)
- `__versa_string & replace` (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- `__versa_string & replace` (size_type __pos, size_type __n1, const _CharT *__s)
- `__versa_string & replace` (size_type __pos, size_type __n, const __versa_string &__str)
- `__versa_string & replace` (iterator __i1, iterator __i2, std::initializer_list<_CharT> __l)
- `template<class _InputIterator>`
`__versa_string & replace` (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- `__versa_string & replace` (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- `__versa_string & replace` (iterator __i1, iterator __i2, const _CharT *__s)
- `__versa_string & replace` (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- `__versa_string & replace` (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2)
- `__versa_string & replace` (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- `__versa_string & replace` (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)
- `__versa_string & replace` (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)
- `__versa_string & replace` (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)
- `void reserve` (size_type __res_arg=0)
- `void resize` (size_type __n)
- `void resize` (size_type __n, _CharT __c)
- `size_type rfind` (const _CharT *__s, size_type __pos, size_type __n) const
- `size_type rfind` (const __versa_string &__str, size_type __pos=npos) const
- `size_type rfind` (const _CharT *__s, size_type __pos=npos) const
- `size_type rfind` (_CharT __c, size_type __pos=npos) const
- `void shrink_to_fit` ()
- `size_type size` () const
- `__versa_string substr` (size_type __pos=0, size_type __n=npos) const
- `void swap` (__versa_string &__s)

Static Public Attributes

- static const size_type [npos](#)

5.22.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc, template<
typename, typename, typename > class _Base> class __gnu_cxx::__versa_
string< _CharT, _Traits, _Alloc, _Base >
```

Template class [__versa_string](#).

Data structure managing sequences of characters and character-like objects.

Definition at line 52 of file `vstring.h`.

5.22.2 Constructor & Destructor Documentation

5.22.2.1

```
template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( ) [inline]
```

Default constructor creates an empty string.

Definition at line 130 of file `vstring.h`.

5.22.2.2

```
template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( const _Alloc & __a ) [inline, explicit]
```

Construct an empty string using allocator *a*.

Definition at line 137 of file `vstring.h`.

5.22.2.3

```
template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str ) [inline]
```

Construct string with copy of value of *str*.

Parameters

`__str` Source string.

Definition at line 145 of file `vstring.h`.

```
5.22.2.4 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( __versa_string< _CharT, _Traits, _Alloc, _Base >
&& __str ) [inline]
```

String move constructor.

Parameters

`__str` Source string.

The newly-constructed string contains the exact contents of `str`. The contents of `str` are a valid, but unspecified string.

Definition at line 157 of file `vstring.h`.

```
5.22.2.5 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( std::initializer_list< _CharT > __l, const _Alloc
& __a = _Alloc() ) [inline]
```

Construct string from an initializer list.

Parameters

`__l` `std::initializer_list` of characters.

`__a` Allocator to use (default is default allocator).

Definition at line 165 of file `vstring.h`.

```
5.22.2.6 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos, size_type __n = npos )
[inline]
```

Construct string as copy of a substring.

Parameters

- `__str` Source string.
- `__pos` Index of first character to copy from.
- `__n` Number of characters to copy (default remainder).

Definition at line 176 of file `vstring.h`.

```
5.22.2.7 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos, size_type __n, const _Alloc & __a
) [inline]
```

Construct string as copy of a substring.

Parameters

- `__str` Source string.
- `__pos` Index of first character to copy from.
- `__n` Number of characters to copy.
- `__a` Allocator to use.

Definition at line 191 of file `vstring.h`.

```
5.22.2.8 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::__versa_string ( const _CharT * __s, size_type __n, const _Alloc
& __a = _Alloc() ) [inline]
```

Construct string initialized by a character array.

Parameters

- `__s` Source character array.
- `__n` Number of characters to copy.
- `__a` Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 208 of file `vstring.h`.


```
5.22.2.9  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::__versa_string ( const _CharT * __s, const _Alloc & __a =
            _Alloc() ) [inline]
```

Construct string as copy of a C string.

Parameters

`__s` Source C string.

`__a` Allocator to use (default is default allocator).

Definition at line 217 of file `vstring.h`.

```
5.22.2.10 template<typename _CharT, typename _Traits, typename _Alloc,
                 template< typename, typename, typename > class _Base>
                 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
                 >::__versa_string ( size_type __n, _CharT __c, const _Alloc & __a
                 = _Alloc() ) [inline]
```

Construct string as multiple characters.

Parameters

`__n` Number of characters.

`__c` Character to use.

`__a` Allocator to use (default is default allocator).

Definition at line 227 of file `vstring.h`.

```
5.22.2.11 template<typename _CharT, typename _Traits, typename _Alloc,
                 template< typename, typename, typename > class _Base>
                 template<class _InputIterator > __gnu_cxx::__versa_string<
                 _CharT, _Traits, _Alloc, _Base >::__versa_string ( _InputIterator
                 __beg, _InputIterator __end, const _Alloc & __a = _Alloc() )
                 [inline]
```

Construct string as copy of a range.

Parameters

`__beg` Start of range.

`__end` End of range.

`__a` Allocator to use (default is default allocator).

Definition at line 237 of file `vstring.h`.

```
5.22.2.12  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::~__versa_string( ) [inline]
```

Destroy the string instance.

Definition at line 244 of file `vstring.h`.

5.22.3 Member Function Documentation

```
5.22.3.1  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
            _Base >::append ( const __versa_string< _CharT, _Traits, _Alloc,
            _Base > & __str ) [inline]
```

Append a string to this string.

Parameters

`__str` The string to append.

Returns

Reference to this string.

Definition at line 676 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `std::getline()`, and `__gnu_cxx::operator+()`.

```
5.22.3.2  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
            _Base >::append ( const __versa_string< _CharT, _Traits, _Alloc,
            _Base > & __str, size_type __pos, size_type __n ) [inline]
```

Append a substring.

Parameters

- `__str` The string to append.
- `__pos` Index of the first character of `str` to append.
- `__n` The number of characters to append.

Returns

Reference to this string.

Exceptions

[*`std::out_of_range`*](#) if `pos` is not a valid index.

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 693 of file `vstring.h`.

```
5.22.3.3  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
            _Base >::append ( const _CharT * __s, size_type __n ) [inline]
```

Append a C substring.

Parameters

- `__s` The C string to append.
- `__n` The number of characters to append.

Returns

Reference to this string.

Definition at line 705 of file `vstring.h`.

```
5.22.3.4  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
            _Base >::append ( const _CharT * __s ) [inline]
```

Append a C string.

Parameters

`__s` The C string to append.

Returns

Reference to this string.

Definition at line 718 of file `vstring.h`.

5.22.3.5 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::append (size_type __n, _CharT __c) [inline]`

Append multiple characters.

Parameters

`__n` The number of characters to append.

`__c` The character to use.

Returns

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 735 of file `vstring.h`.

5.22.3.6 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::append (std::initializer_list< _CharT > __l) [inline]`

Append an `initializer_list` of characters.

Parameters

`__l` The `initializer_list` of characters to append.

Returns

Reference to this string.

Definition at line 745 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

5.22.3.7 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> template<class _InputIterator > __versa_string&
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::append (_InputIterator __first, _InputIterator __last)
[inline]`

Append a range of characters.

Parameters

`__first` Iterator referencing the first character to append.
`__last` Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 759 of file `vstring.h`.

5.22.3.8 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::assign (const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str) [inline]`

Set value to contents of another string.

Parameters

`__str` Source string to use.

Returns

Reference to this string.

Definition at line 782 of file `vstring.h`.

5.22.3.9 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::assign (__versa_string< _CharT, _Traits, _Alloc, _Base >
&& __str) [inline]`

Set value to contents of another string.

Parameters

`__str` Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 798 of file `vstring.h`.

```
5.22.3.10 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::assign ( const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __str, size_type __pos, size_type __n )
[inline]
```

Set value to a substring of a string.

Parameters

`__str` The string to use.

`__pos` Index of the first character of str.

`__n` Number of characters to use.

Returns

Reference to this string.

Exceptions

[*`std::out_of_range`*](#) if `__pos` is not a valid index.

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 819 of file `vstring.h`.

```
5.22.3.11 template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::assign ( const _CharT * __s, size_type __n )
[inline]
```

Set value to a C substring.

Parameters

- `__s` The C string to use.
- `__n` Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 836 of file `vstring.h`.

5.22.3.12 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base>`
`__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base >::assign (const _CharT * __s) [inline]`

Set value to contents of a C string.

Parameters

- `__s` The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 852 of file `vstring.h`.

5.22.3.13 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base>`
`__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base >::assign (size_type __n, _CharT __c) [inline]`

Set value to multiple characters.

Parameters

- `__n` Length of the resulting string.
- `__c` The character to use.

Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 869 of file `vstring.h`.

5.22.3.14 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator > __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (_InputIterator __first, _InputIterator __last) [inline]`

Set value to a range of characters.

Parameters

`__first` Iterator referencing the first character to append.

`__last` Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

Definition at line 883 of file `vstring.h`.

5.22.3.15 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (std::initializer_list< _CharT > __l) [inline]`

Set value to an `initializer_list` of characters.

Parameters

`__l` The `initializer_list` of characters to assign.

Returns

Reference to this string.

5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference 813

Definition at line 893 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`.

5.22.3.16 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::at (size_type __n) const [inline]`

Provides access to the data contained in the string.

Parameters

`__n` The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

[*`std::out_of_range`*](#) If `__n` is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 567 of file `vstring.h`.

5.22.3.17 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::at (size_type __n) [inline]`

Provides access to the data contained in the string.

Parameters

`__n` The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

[*`std::out_of_range`*](#) If `__n` is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 586 of file `vstring.h`.

5.22.3.18 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::back () const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 624 of file `vstring.h`.

5.22.3.19 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> reference
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back
() [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 616 of file `vstring.h`.

5.22.3.20 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> iterator
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin
() [inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 310 of file `vstring.h`.

5.22.3.21 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 321 of file `vstring.h`.

5.22.3.22 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> const
 _CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::c_str () const [inline]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1485 of file `vstring.h`.

5.22.3.23 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> size_type
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::capacity () const [inline]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 478 of file `vstring.h`.

5.22.3.24 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
 _Base >::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 385 of file `vstring.h`.

5.22.3.25 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
 _Base >::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 393 of file `vstring.h`.

5.22.3.26 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear
() [inline]`

Erases the string, making it empty.

Definition at line 506 of file `vstring.h`.

5.22.3.27 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::compare(const __versa_string< _CharT, _Traits, _Alloc, _Base
> & __str) const [inline]`

Compare to a string.

Parameters

`__str` String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1904 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

5.22.3.28 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::compare(size_type __pos, size_type __n, const __versa_string<
_CharT, _Traits, _Alloc, _Base > & __str) const`

Compare substring to a string.

Parameters

- `__pos` Index of first character of substring.
- `__n` Number of characters in substring.
- `__str` String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 458 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.29 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare (size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2, size_type __n2) const`

Compare substring to a substring.

Parameters

- `__pos1` Index of first character of substring.
- `__n1` Number of characters in substring.
- `__str` String to compare against.
- `__pos2` Index of first character of substring of `str`.
- `__n2` Number of characters in substring of `str`.

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer `< 0` if this substring is ordered before the substring of `__str`, `0` if their values are equivalent, or `> 0` if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 475 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `std::min()`.

5.22.3.30 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (const _CharT * __s) const`

Compare to a C string.

Parameters

`__s` C string to compare against.

Returns

Integer `< 0`, `0`, or `> 0`.

Returns an integer `< 0` if this string is ordered before `__s`, `0` if their values are equivalent, or `> 0` if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 494 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length()`, `std::min()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

5.22.3.31 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::compare (size_type __pos, size_type __n1, const _CharT * __s
) const`

Compare substring to a C string.

Parameters

`__pos` Index of first character of substring.
`__n1` Number of characters in substring.
`__s` C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 510 of file `vstring.tcc`.

References `std::min()`.

5.22.3.32 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::compare (size_type __pos, size_type __n1, const _CharT * __s,
size_type __n2) const`

Compare substring against a character array.

Parameters

`__pos1` Index of first character of substring.
`__n1` Number of characters in substring.
`__s` character array to compare against.
`__n2` Number of characters of `s`.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `r1en` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,r1en)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `\0` has no special meaning.

Definition at line 527 of file `vstring.tcc`.

References `std::min()`.

```
5.22.3.33  template<typename _CharT, typename _Traits, typename _Alloc
            , template< typename, typename, typename > class _Base>
            __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::copy
            ( _CharT * __s, size_type __n, size_type __pos = 0 ) const
```

Copy substring into C string.

Parameters

`__s` C string to copy value into.

`__n` Number of characters to copy.

`__pos` Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

[`std::out_of_range`](#) If `pos > size()`.

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 253 of file `vstring.tcc`.

5.22.3.34 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::crbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 402 of file vstring.h.

References `std::end()`.

5.22.3.35 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::crend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 411 of file vstring.h.

References `std::begin()`.

5.22.3.36 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> const
_CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::data () const [inline]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1495 of file vstring.h.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`.

5.22.3.37 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> bool
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::empty () const [inline]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 514 of file `vstring.h`.

5.22.3.38 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 340 of file `vstring.h`.

5.22.3.39 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> iterator
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () [inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 329 of file `vstring.h`.

5.22.3.40 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::erase (size_type __pos = 0, size_type __n = npos)
[inline]`

Remove characters.

Parameters

`__pos` Index of first character to remove (default 0).

`__n` Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

[std::out_of_range](#) If `__pos` is beyond the end of this string.

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1088 of file `vstring.h`.

Referenced by `std::getline()`.

5.22.3.41 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> iterator
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase
(iterator __position) [inline]`

Remove one character.

Parameters

`__position` Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1104 of file `vstring.h`.

5.22.3.42 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> iterator
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase
(iterator __first, iterator __last) [inline]`

Remove a range of characters.

Parameters

`__first` Iterator referencing the first character to remove.

`__last` Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1125 of file vstring.h.

```
5.22.3.43  template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
const _CharT * __s, size_type __pos, size_type __n ) const
```

Find position of a C substring.

Parameters

`__s` C string to locate.

`__pos` Index of character to search from.

`__n` Number of characters from `__s` to search for.

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 268 of file vstring.tcc.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`.

```
5.22.3.44  template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
size_type __pos = 0 ) const  [inline]
```

Find position of a string.

Parameters

`__str` String to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1531 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`.

5.22.3.45 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find (const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a C string.

Parameters

`__s` C string to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1545 of file `vstring.h`.

5.22.3.46 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find (_CharT __c, size_type __pos = 0) const`

Find position of a character.

Parameters

`__c` Character to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 292 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

```
5.22.3.47  template<typename _CharT, typename _Traits, typename _Alloc
            , template< typename, typename, typename > class _Base>
            __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::find_first_not_of ( const _CharT * __s, size_type __pos,
            size_type __n ) const
```

Find position of a character not in C substring.

Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search from.

`__n` Number of characters from `s` to consider.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 390 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.48 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_first_not_of (_CharT __c, size_type __pos = 0) const`

Find position of a different character.

Parameters

`__c` Character to avoid.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 403 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.49 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_first_not_of (const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos = 0) const [inline]`

Find position of a character not in string.

Parameters

`__str` String containing characters to avoid.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1759 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`.

5.22.3.50 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of(const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character not in C string.

Parameters

`__s` C string containing characters to avoid.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1789 of file `vstring.h`.

5.22.3.51 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of(const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = 0) const [inline]`

Find position of a character of string.

Parameters

`__str` String containing characters to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference

829

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1634 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`.

5.22.3.52 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base>::size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of(const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a character of C substring.

Parameters

`__s` String containing characters to locate.

`__pos` Index of character to search from.

`__n` Number of characters from `s` to search for.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 351 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.53 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of(const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character of C string.

Parameters

`__s` String containing characters to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1663 of file `vstring.h`.

```
5.22.3.54  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base> size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::find_first_of ( _CharT __c, size_type __pos = 0 ) const
            [inline]
```

Find position of a character.

Parameters

`__c` Character to locate.

`__pos` Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1682 of file `vstring.h`.

```
5.22.3.55  template<typename _CharT, typename _Traits , typename _Alloc
            , template< typename, typename, typename > class _Base>
            __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
            >::find_last_not_of ( _CharT __c, size_type __pos = npos ) const
```

Find last position of a different character.

Parameters

`__c` Character to avoid.

`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 437 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.56 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::find_last_not_of (const __versa_string< _CharT, _Traits, _Alloc,
_Base > & __str, size_type __pos = npos) const [inline]`

Find last position of a character not in string.

Parameters

`__str` String containing characters to avoid.

`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1820 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`.

5.22.3.57 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a character not in C substring.

Parameters

`__s` C string containing characters to avoid.
`__pos` Index of character to search back from.
`__n` Number of characters from `s` to consider.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 415 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

5.22.3.58 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a character not in C string.

Parameters

`__s` C string containing characters to avoid.
`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1851 of file `vstring.h`.

5.22.3.59 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a character of C substring.

Parameters

- `__s` C string containing characters to locate.
- `__pos` Index of character to search back from.
- `__n` Number of characters from `s` to search for.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 368 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.60 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = npos) const [inline]`

Find last position of a character of string.

Parameters

- `__str` String containing characters to locate.
- `__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1697 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

5.22.3.61 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of (const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a character of C string.

Parameters

`__s` C string containing characters to locate.

`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1726 of file `vstring.h`.

5.22.3.62 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of (_CharT __c, size_type __pos = npos) const [inline]`

Find last position of a character.

Parameters

`__c` Character to locate.

`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

5.22 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>` Class Template Reference 835

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1745 of file `vstring.h`.

5.22.3.63 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> reference
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front
() [inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 600 of file `vstring.h`.

5.22.3.64 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::front () const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 608 of file `vstring.h`.

5.22.3.65 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::get_allocator () const [inline]`

Return copy of allocator used to construct this string.

Definition at line 1502 of file `vstring.h`.

5.22.3.66 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::insert (size_type __pos, const _CharT * __s,
size_type __n) [inline]`

Insert a C substring.

Parameters

`__pos` Iterator referencing location in string to insert at.

`__s` The C string to insert.

`__n` The number of characters to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If `__pos` is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1002 of file `vstring.h`.

5.22.3.67 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> iterator
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert
(iterator __p, _CharT __c) [inline]`

Insert one character.

Parameters

`__p` Iterator referencing position in string to insert at.

`__c` The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1063 of file `vstring.h`.

5.22.3.68 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::insert (size_type __pos1, const __versa_string<
_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Insert value of a string.

Parameters

`__pos1` Iterator referencing location in string to insert at.

`__str` The string to insert.

Returns

Reference to this string.

Exceptions

[`std::length_error`](#) If new length exceeds `max_size()`.

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 956 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.69 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert
(iterator __p, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

`__p` Iterator referencing location in string to insert at.

`__n` Number of characters to insert

`__c` The character to insert.

Exceptions

[`std::length_error`](#) If new length exceeds `max_size()`.

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 911 of file `vstring.h`.

5.22.3.70 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (iterator __p, std::initializer_list< _CharT > __l) [inline]`

Insert an `initializer_list` of characters.

Parameters

`__p` Iterator referencing location in string to insert at.

`__l` The `initializer_list` of characters to insert.

Exceptions

`std::length_error` If new length exceeds `max_size()`.

Definition at line 939 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`.

5.22.3.71 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator > void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (iterator __p, _InputIterator __beg, _InputIterator __end) [inline]`

Insert a range of characters.

Parameters

`__p` Iterator referencing location in string to insert at.

`__beg` Start of range.

`__end` End of range.

Exceptions

`std::length_error` If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 928 of file `vstring.h`.

5.22.3.72 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

`__pos` Index in string to insert at.
`__n` Number of characters to insert
`__c` The character to insert.

Returns

Reference to this string.

Exceptions

`std::length_error` If new length exceeds `max_size()`.
`std::out_of_range` If `__pos` is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1045 of file `vstring.h`.

5.22.3.73 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2, size_type __n) [inline]`

Insert a substring.

Parameters

`__pos1` Iterator referencing location in string to insert at.

__str The string to insert.
__pos2 Start of characters in str to insert.
__n Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.
std::out_of_range If `__pos1 > size()` or `__pos2 > __str.size()`.

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 979 of file `vstring.h`.

5.22.3.74 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base>`
`__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base >::insert (size_type __pos, const _CharT * __s)`
`[inline]`

Insert a C string.

Parameters

__pos Iterator referencing location in string to insert at.
__s The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.
std::out_of_range If `__pos` is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1021 of file `vstring.h`.

5.22.3.75 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> size_type
 __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
 >::length () const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 426 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

5.22.3.76 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> size_type
 __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
 >::max_size () const [inline]`

Returns the `size()` of the largest possible string.

Definition at line 431 of file `vstring.h`.

Referenced by `std::getline()`.

5.22.3.77 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
 _Alloc, _Base>::operator+=(const __versa_string<_CharT,
 _Traits, _Alloc, _Base> & __str) [inline]`

Append a string to this string.

Parameters

`__str` The string to append.

Returns

Reference to this string.

Definition at line 635 of file `vstring.h`.

5.22.3.78 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
 _Alloc, _Base>::operator+=(const _CharT * __s) [inline]`

Append a C string.

Parameters

`__s` The C string to append.

Returns

Reference to this string.

Definition at line 644 of file `vstring.h`.

```
5.22.3.79  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::operator+=( _CharT __c ) [inline]
```

Append a character.

Parameters

`__c` The character to append.

Returns

Reference to this string.

Definition at line 653 of file `vstring.h`.

```
5.22.3.80  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
            _Alloc, _Base >::operator+=( std::initializer_list< _CharT > __l )
            [inline]
```

Append an `initializer_list` of characters.

Parameters

`__l` The `initializer_list` of characters to be appended.

Returns

Reference to this string.

Definition at line 666 of file `vstring.h`.

5.22.3.81 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator= (const _CharT * __s) [inline]`

Copy contents of `__s` into this string.

Parameters

`__s` Source null-terminated string.

Definition at line 287 of file `vstring.h`.

5.22.3.82 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator= (std::initializer_list< _CharT > __l)
[inline]`

Set value to string constructed from initializer list.

Parameters

`__l` [std::initializer_list](#).

Definition at line 275 of file `vstring.h`.

5.22.3.83 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator= (__versa_string< _CharT, _Traits,
_Alloc, _Base > && __str) [inline]`

String move assignment operator.

Parameters

`__str` Source string.

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 263 of file `vstring.h`.

5.22.3.84 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator= (_CharT __c) [inline]`

Set value to string of length 1.

Parameters

`__c` Source character.

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 298 of file `vstring.h`.

5.22.3.85 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::operator= (const __versa_string< _CharT, _Traits,
_Alloc, _Base > & __str) [inline]`

Assign the value of `str` to this string.

Parameters

`__str` Source string.

Definition at line 251 of file `vstring.h`.

5.22.3.86 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,
_Base >::operator[] (size_type __pos) const [inline]`

Subscript access to the data contained in the string.

Parameters

`__pos` The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 529 of file `vstring.h`.

5.22.3.87 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> reference
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::operator[] (size_type __pos) [inline]`

Subscript access to the data contained in the string.

Parameters

`__pos` The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see [at\(\)](#).) Unshares the string.

Definition at line 546 of file `vstring.h`.

5.22.3.88 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base> void
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
 >::push_back (_CharT __c) [inline]`

Append a single character.

Parameters

`__c` Character to append.

Definition at line 767 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

5.22.3.89 `template<typename _CharT, typename _Traits, typename _Alloc,
 template< typename, typename, typename > class _Base>
 reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,
 _Alloc, _Base >::rbegin () [inline]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 349 of file `vstring.h`.

References `std::end()`.

5.22.3.90 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 358 of file `vstring.h`.

References `std::end()`.

5.22.3.91 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 376 of file `vstring.h`.

References `std::begin()`.

5.22.3.92 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 367 of file `vstring.h`.

References `std::begin()`.

5.22.3.93 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT,
_Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2,
std::initializer_list< _CharT > __l) [inline]`

Replace range of characters with `initializer_list`.

Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__l` The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1421 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

5.22.3.94 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::replace (iterator __i1, iterator __i2, const
__versa_string<_CharT, _Traits, _Alloc, _Base> & __str)
[inline]`

Replace range of characters with string.

Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__str` String value to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1270 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

```
5.22.3.95  template<typename _CharT, typename _Traits, typename _Alloc,
            template< typename, typename, typename > class _Base>
            __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
            _Alloc, _Base>::replace ( size_type __pos, size_type __n, const
            __versa_string<_CharT, _Traits, _Alloc, _Base> & __str )
            [inline]
```

Replace characters with value from another string.

Parameters

`__pos` Index of first character to replace.
`__n` Number of characters to be replaced.
`__str` String to insert.

Returns

Reference to this string.

Exceptions

[*`std::out_of_range`*](#) If `__pos` is beyond the end of this string.
[*`std::length_error`*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1153 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

5.22.3.96 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::replace (size_type __pos, size_type __n1,
size_type __n2, _CharT __c) [inline]`

Replace characters with multiple characters.

Parameters

`__pos` Index of first character to replace.
`__n1` Number of characters to be replaced.
`__n2` Number of characters to insert.
`__c` Character to insert.

Returns

Reference to this string.

Exceptions

[std::out_of_range](#) If `__pos > size()`.
[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1252 of file `vstring.h`.

5.22.3.97 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::replace (iterator __i1, iterator __i2, size_type
__n, _CharT __c) [inline]`

Replace range of characters with multiple characters.

Parameters

`__i1` Iterator referencing start of range to replace.
`__i2` Iterator referencing end of range to replace.
`__n` Number of characters to insert.
`__c` Character to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1330 of file `vstring.h`.

5.22.3.98 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits,
_Alloc, _Base >::replace (iterator __i1, iterator __i2, const
_CharT* __s) [inline]`

Replace range of characters with C string.

Parameters

`__i1` Iterator referencing start of range to replace.

`__i2` Iterator referencing end of range to replace.

`__s` C string value to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1309 of file `vstring.h`.

5.22.3.99 `template<typename _CharT, typename _Traits, typename
_Alloc, template< typename, typename, typename > class
_Base> template<class _InputIterator > __versa_string&
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base
>::replace (iterator __i1, iterator __i2, _InputIterator __k1,
_InputIterator __k2) [inline]`

Replace range of characters with range.

Parameters

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.
- `__k1` Iterator referencing start of range to insert.
- `__k2` Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1353 of file `vstring.h`.

5.22.3.100 `template<typename _CharT, typename _Traits, typename _Alloc,
template< typename, typename, typename > class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits,
_Alloc, _Base>::replace (size_type __pos, size_type __n1, const
_CharT * __s, size_type __n2) [inline]`

Replace characters with value of a C substring.

Parameters

- `__pos` Index of first character to replace.
- `__n1` Number of characters to be replaced.
- `__s` C string to insert.
- `__n2` Number of characters from `__s` to use.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If `__pos1 > size()`.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1204 of file `vstring.h`.

5.22.3.101 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (size_type __pos, size_type __n1, const _CharT * __s) [inline]`

Replace characters with value of a C string.

Parameters

`__pos` Index of first character to replace.

`__n1` Number of characters to be replaced.

`__s` C string to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If `__pos > size()`.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1228 of file `vstring.h`.

5.22.3.102 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (size_type __pos1, size_type __n1, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2, size_type __n2) [inline]`

Replace characters with value from another string.

Parameters

- `__pos1` Index of first character to replace.
- `__n1` Number of characters to be replaced.
- `__str` String to insert.
- `__pos2` Index of first character of str to use.
- `__n2` Number of characters from str to use.

Returns

Reference to this string.

Exceptions

- [*std::out_of_range*](#) If `__pos1 > size()` or `__pos2 > str.size()`.
- [*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos1, pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1176 of file `vstring.h`.

5.22.3.103 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n) [inline]`

Replace range of characters with C substring.

Parameters

- `__i1` Iterator referencing start of range to replace.
- `__i2` Iterator referencing end of range to replace.

`__s` C string value to insert.

`__n` Number of characters from `s` to insert.

Returns

Reference to this string.

Exceptions

`std::length_error` If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first `n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1288 of file `vstring.h`.

5.22.3.104 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve (size_type __res_arg = 0) [inline]`

Attempt to preallocate enough memory for specified number of characters.

Parameters

`__res_arg` Number of characters required.

Exceptions

`std::length_error` If `__res_arg` exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 499 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

5.22.3.105 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::resize (size_type __n, _CharT __c)`

Resizes the string to the specified number of characters.

Parameters

`__n` Number of characters the string should contain.

`__c` Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 49 of file `vstring.tcc`.

5.22.3.106 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::resize (size_type __n) [inline]`

Resizes the string to the specified number of characters.

Parameters

`__n` Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 458 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize()`.

5.22.3.107 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::rfind (const _CharT * __s, size_type __pos, size_type __n)
const`

Find last position of a C substring.

Parameters

`__s` C string to locate.

`__pos` Index of character to search back from.

`__n` Number of characters from `s` to search for.

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 311 of file `vstring.tcc`.

References `std::min()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

5.22.3.108 `template<typename _CharT, typename _Traits, typename _Alloc
, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
>::rfind (_CharT __c, size_type __pos = npos) const`

Find last position of a character.

Parameters

`__c` Character to locate.

`__pos` Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 333 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::npos`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

5.22.3.109 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> size_type`
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`
`>::rfind (const _CharT * __s, size_type __pos = npos) const`
`[inline]`

Find last position of a C string.

Parameters

`__s` C string to locate.

`__pos` Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1604 of file `vstring.h`.

5.22.3.110 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> size_type`
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`
`>::rfind (const __versa_string< _CharT, _Traits, _Alloc, _Base >`
`& __str, size_type __pos = npos) const [inline]`

Find last position of a string.

Parameters

`__str` String to locate.

`__pos` Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1575 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`.

5.22.3.111 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> void`
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`
`>::shrink_to_fit () [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 464 of file `vstring.h`.

5.22.3.112 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> size_type`
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size`
`() const [inline]`

Returns the number of characters in the string, not including any `///` null-termination.

Definition at line 420 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`, `__gnu_cxx::operator+`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`.

5.22.3.113 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base>`
`__versa_string __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base >::substr (size_type __pos = 0, size_type __n = npos`
`) const [inline]`

Get a substring.

Parameters

- `__pos` Index of first character (default 0).
- `__n` Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

[*`std::out_of_range`*](#) If `pos > size()`.

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 1883 of file `vstring.h`.

5.22.3.114 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> void`
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`
`>::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > & __s`
`) [inline]`

Swap contents with another string.

Parameters

- `__s` String to swap with.

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1474 of file `vstring.h`.

Referenced by `__gnu_cxx::swap()`.

5.22.4 Member Data Documentation

5.22.4.1 `template<typename _CharT, typename _Traits, typename _Alloc,`
`template< typename, typename, typename > class _Base> const`
`__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type`
`__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos`
`[static]`

Value returned by various member functions when they fail.

Definition at line 77 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

5.23 `__gnu_cxx::_Caster<_ToType>` Struct Template Reference

Public Types

- `typedef _ToType::element_type * type`

5.23.1 Detailed Description

`template<typename _ToType> struct __gnu_cxx::_Caster<_ToType>`

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 45 of file `cast.h`.

The documentation for this struct was generated from the following file:

- `cast.h`

5.24 `__gnu_cxx::_Char_types<_CharT>` Struct Template Reference

Mapping from character type to associated types.

Public Types

- typedef unsigned long **int_type**
- typedef [std::streamoff](#) **off_type**
- typedef [std::streampos](#) **pos_type**
- typedef [std::mbstate_t](#) **state_type**

5.24.1 Detailed Description

```
template<typename _CharT> struct __gnu_cxx::_Char_types< _CharT >
```

Mapping from character type to associated types.

Note

This is an implementation class for the generic version of [char_traits](#). It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, [streamoff](#), [streampos](#), and [mbstate_t](#). Users who need a different set of types, but who don't need to change the definitions of any function defined in [char_traits](#), can specialize `__gnu_cxx::_Char_types` while leaving `__gnu_cxx::char_traits` alone.

Definition at line 58 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.25 `__gnu_cxx::_ExtPtr_allocator< _Tp >` Class Template Reference

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).

Public Types

- typedef [_Pointer_adapter< _Relative_pointer_impl< const _Tp > >](#) **const_pointer**
- typedef `const _Tp &` **const_reference**
- typedef `std::ptrdiff_t` **difference_type**
- typedef [_Pointer_adapter< _Relative_pointer_impl< _Tp > >](#) **pointer**
- typedef `_Tp &` **reference**

- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **_ExtPtr_allocator** (const [_ExtPtr_allocator](#) &__rarg) throw ()
- template<typename _Up >
 _ExtPtr_allocator (const [_ExtPtr_allocator](#)< _Up > &__rarg) throw ()
- const [std::allocator](#)< _Tp > & **_M_getUnderlyingImp** () const
- [pointer](#) **address** (reference __x) const
- [const_pointer](#) **address** (const_reference __x) const
- [pointer](#) **allocate** (size_type __n, void *__hint=0)
- void **construct** ([pointer](#) __p, const _Tp &__val)
- template<typename... _Args>
 void **construct** ([pointer](#) __p, _Args &&...__args)
- void **deallocate** ([pointer](#) __p, size_type __n)
- void **destroy** ([pointer](#) __p)
- size_type **max_size** () const throw ()
- template<typename _Up >
 bool **operator!=** (const [_ExtPtr_allocator](#)< _Up > &__rarg)
- bool **operator!=** (const [_ExtPtr_allocator](#) &__rarg)
- template<typename _Up >
 bool **operator==** (const [_ExtPtr_allocator](#)< _Up > &__rarg)
- bool **operator==** (const [_ExtPtr_allocator](#) &__rarg)

Friends

- template<typename _Up >
 void **swap** ([_ExtPtr_allocator](#)< _Up > &, [_ExtPtr_allocator](#)< _Up > &)

5.25.1 Detailed Description

template<typename _Tp> class [__gnu_cxx::_ExtPtr_allocator](#)< _Tp >

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).

Definition at line 52 of file [extptr_allocator.h](#).

The documentation for this class was generated from the following file:

- [extptr_allocator.h](#)

5.26 `__gnu_cxx::_Invalid_type` Struct Reference

5.26.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of void*, const void*, and so on.

Definition at line 205 of file pointer.h.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

5.27 `__gnu_cxx::_Pointer_adapter< _Storage_policy >` Class Template Reference

Inherits `_Storage_policy`.

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef `_Storage_policy::element_type` **element_type**
- typedef [std::random_access_iterator_tag](#) **iterator_category**
- typedef `_Pointer_adapter` **pointer**
- typedef `_Reference_type< element_type >::reference` **reference**
- typedef `_Unqualified_type< element_type >::type` **value_type**

Public Member Functions

- `_Pointer_adapter` (element_type * __arg=0)
- `_Pointer_adapter` (const [_Pointer_adapter](#) & __arg)
- template<typename _Up >
 `_Pointer_adapter` (const [_Pointer_adapter](#)< _Up > & __arg)
- template<typename _Up >
 `_Pointer_adapter` (_Up * __arg)
- `operator __unspecified_bool_type` () const
- `bool operator!` () const
- `reference operator*` () const
- [_Pointer_adapter](#) & `operator++` ()
- [_Pointer_adapter](#) `operator++` (int)

- [_Pointer_adapter](#) & **operator+=** (unsigned short __offset)
- [_Pointer_adapter](#) & **operator+=** (int __offset)
- [_Pointer_adapter](#) & **operator+=** (unsigned int __offset)
- [_Pointer_adapter](#) & **operator+=** (long __offset)
- [_Pointer_adapter](#) & **operator+=** (unsigned long __offset)
- [_Pointer_adapter](#) & **operator+=** (short __offset)
- template<typename _Up >
std::ptrdiff_t **operator-** (const [_Pointer_adapter](#)< _Up > &__rhs) const
- [_Pointer_adapter](#) **operator--** (int)
- [_Pointer_adapter](#) & **operator--** ()
- [_Pointer_adapter](#) & **operator-=** (unsigned short __offset)
- [_Pointer_adapter](#) & **operator-=** (short __offset)
- [_Pointer_adapter](#) & **operator-=** (int __offset)
- [_Pointer_adapter](#) & **operator-=** (long __offset)
- [_Pointer_adapter](#) & **operator-=** (unsigned long __offset)
- [_Pointer_adapter](#) & **operator-=** (unsigned int __offset)
- element_type * **operator->** () const
- [_Pointer_adapter](#) & **operator=** (const [_Pointer_adapter](#) &__arg)
- template<typename _Up >
[_Pointer_adapter](#) & **operator=** (_Up *__arg)
- template<typename _Up >
[_Pointer_adapter](#) & **operator=** (const [_Pointer_adapter](#)< _Up > &__arg)
- reference **operator[]** (std::ptrdiff_t __index) const

Friends

- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) &__lhs, short __offset)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) &__lhs, unsigned short __offset)
- [_Pointer_adapter](#) **operator+** (unsigned long __offset, const [_Pointer_adapter](#) &__rhs)
- [_Pointer_adapter](#) **operator+** (unsigned short __offset, const [_Pointer_adapter](#) &__rhs)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) &__lhs, long __offset)
- [_Pointer_adapter](#) **operator+** (long __offset, const [_Pointer_adapter](#) &__rhs)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) &__lhs, int __offset)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) &__lhs, unsigned long __offset)
- [_Pointer_adapter](#) **operator+** (unsigned int __offset, const [_Pointer_adapter](#) &__rhs)
- [_Pointer_adapter](#) **operator+** (short __offset, const [_Pointer_adapter](#) &__rhs)
- [_Pointer_adapter](#) **operator+** (int __offset, const [_Pointer_adapter](#) &__rhs)

- `Pointer_adapter operator+` (const `Pointer_adapter` &__lhs, unsigned int __offset)
- `std::ptrdiff_t operator-` (element_type *__lhs, const `Pointer_adapter` &__rhs)
- `std::ptrdiff_t operator-` (const `Pointer_adapter` &__lhs, element_type *__rhs)
- `Pointer_adapter operator-` (const `Pointer_adapter` &__lhs, unsigned short __offset)
- `Pointer_adapter operator-` (const `Pointer_adapter` &__lhs, unsigned long __offset)
- `Pointer_adapter operator-` (const `Pointer_adapter` &__lhs, int __offset)
- `Pointer_adapter operator-` (const `Pointer_adapter` &__lhs, short __offset)
- `template<typename _Up> std::ptrdiff_t operator-` (_Up *__lhs, const `Pointer_adapter` &__rhs)
- `Pointer_adapter operator-` (const `Pointer_adapter` &__lhs, long __offset)
- `template<typename _Up> std::ptrdiff_t operator-` (const `Pointer_adapter` &__lhs, _Up *__rhs)
- `Pointer_adapter operator-` (const `Pointer_adapter` &__lhs, unsigned int __offset)

5.27.1 Detailed Description

`template<typename _Storage_policy> class __gnu_cxx::Pointer_adapter<_Storage_policy>`

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from `ext/cast.h`. This allows pointer cast operations to be overloaded is necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == Pointer_adapter<_Std_pointer_impl<_Tp>>>; const _Tp* == Pointer_adapter<_Std_pointer_impl<const _Tp>>>;
_Tp* const == const Pointer_adapter<_Std_pointer_impl<_Tp>>>; const _Tp* const == const Pointer_adapter<_Std_pointer_impl<const _Tp>>>;
```

Definition at line 281 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.28 `__gnu_cxx::_Relative_pointer_impl< _Tp >` Class Template Reference

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `_Tp * get () const`
- `bool operator< (const _Relative_pointer_impl &__rarg) const`
- `bool operator== (const _Relative_pointer_impl &__rarg) const`
- `void set (_Tp *__arg)`

5.28.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::_Relative_pointer_impl< _Tp >`

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address. This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null. As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 101 of file pointer.h.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.29 `__gnu_cxx::_Relative_pointer_impl< const _Tp >` Class Template Reference

Public Types

- `typedef const _Tp element_type`

Public Member Functions

- `const _Tp * get () const`
- `bool operator< (const _Relative_pointer_impl &__rarg) const`
- `bool operator== (const _Relative_pointer_impl &__rarg) const`
- `void set (const _Tp *__arg)`

5.29.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::_Relative_pointer_impl< const _Tp >`

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic.

Definition at line 153 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.30 `__gnu_cxx::_Std_pointer_impl< _Tp >` Class Template Reference

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `_Tp * get () const`

- `bool operator< (const _Std_pointer_impl &__rarg) const`
- `bool operator== (const _Std_pointer_impl &__rarg) const`
- `void set (element_type *__arg)`

5.30.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::_Std_pointer_impl< _Tp >`

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer. A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==()` to support pointer comparison.

Definition at line 58 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

5.31 __gnu_cxx::_Unqualified_type< _Tp > Struct Template Reference

Public Types

- `typedef _Tp type`

5.31.1 Detailed Description

`template<typename _Tp> struct __gnu_cxx::_Unqualified_type< _Tp >`

This structure accomodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_type` is still `T`.

Definition at line 233 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

5.32 __gnu_cxx::annotate_base Struct Reference

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Inheritance diagram for __gnu_cxx::annotate_base:



Public Member Functions

- void **check_allocated** (size_t label)
- void **check_allocated** (void *p, size_t size)
- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)

Friends

- [std::ostream](#) & **operator<<** ([std::ostream](#) &, const [annotate_base](#) &)

5.32.1 Detailed Description

Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Definition at line 94 of file [throw_allocator.h](#).

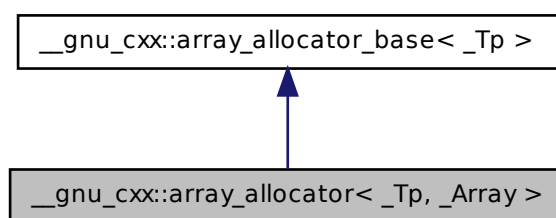
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.33 `__gnu_cxx::array_allocator< _Tp, _Array >` Class Template Reference

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

Inheritance diagram for `__gnu_cxx::array_allocator< _Tp, _Array >`:



Public Types

- `typedef _Array array_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `array_allocator` (`array_type *__array=0`) `throw ()`
- `array_allocator` (`const array_allocator &__o`) `throw ()`
- `template<typename _Tp1, typename _Array1 >`
`array_allocator` (`const array_allocator< _Tp1, _Array1 > &`) `throw ()`
- `pointer address` (`reference __x`) `const`
- `const_pointer address` (`const_reference __x`) `const`
- `pointer allocate` (`size_type __n, const void *==0`)

- void **construct** (pointer `__p`, const `_Tp` &`__val`)
- template<typename... `_Args`>
void **construct** (pointer `__p`, `_Args` &&...`__args`)
- void **deallocate** (pointer, size_type)
- void **destroy** (pointer `__p`)
- size_type **max_size** () const throw ()

5.33.1 Detailed Description

template<typename `_Tp`, typename `_Array` = `std::tr1::array<_Tp, 1>`> class `__gnu_cxx::array_allocator< _Tp, _Array >`

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

Definition at line 96 of file `array_allocator.h`.

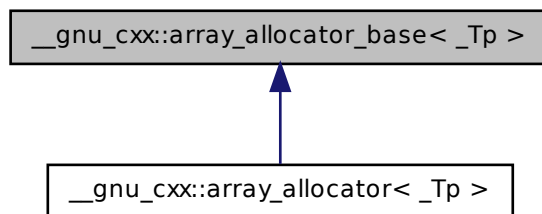
The documentation for this class was generated from the following file:

- [array_allocator.h](#)

5.34 `__gnu_cxx::array_allocator_base< _Tp >` Class Template Reference

Base class.

Inheritance diagram for `__gnu_cxx::array_allocator_base< _Tp >`:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- void **construct** (pointer __p, const _Tp &__val)
- template<typename... _Args>
void **construct** (pointer __p, _Args &&...__args)
- void **deallocate** (pointer, size_type)
- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

5.34.1 Detailed Description

template<typename _Tp> class __gnu_cxx::array_allocator_base< _Tp >

Base class.

Definition at line 46 of file array_allocator.h.

The documentation for this class was generated from the following file:

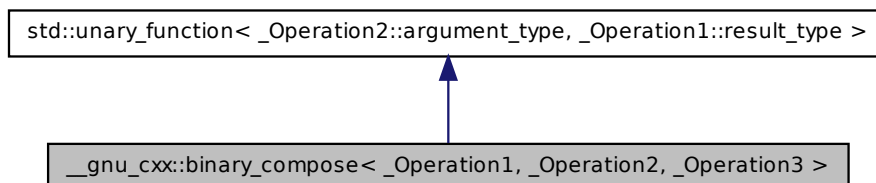
- [array_allocator.h](#)

5.35 __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 > Class Template Reference

An [SGI extension](#) .

5.35 `__gnu_cxx::binary_compose<_Operation1, _Operation2, _Operation3>` Class Template Reference 873

Inheritance diagram for `__gnu_cxx::binary_compose<_Operation1, _Operation2, _Operation3>`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- **binary_compose** (const `_Operation1` &__x, const `_Operation2` &__y, const `_Operation3` &__z)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &__x) const

Protected Attributes

- `_Operation1` **_M_fn1**
- `_Operation2` **_M_fn2**
- `_Operation3` **_M_fn3**

5.35.1 Detailed Description

`template<class _Operation1, class _Operation2, class _Operation3> class __gnu_cxx::binary_compose<_Operation1, _Operation2, _Operation3>`

An [SGI extension](#) .

Definition at line 149 of file `ext/functional`.

5.35.2 Member Typedef Documentation

5.35.2.1 `template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.35.2.2 `template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

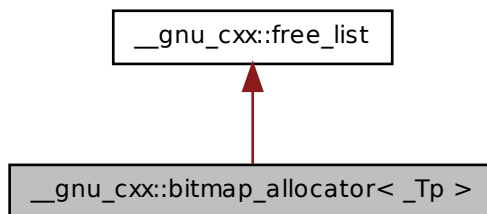
The documentation for this class was generated from the following file:

- [ext/functional](#)

5.36 __gnu_cxx::bitmap_allocator< _Tp > Class Template Reference

Bitmap Allocator, primary template.

Inheritance diagram for `__gnu_cxx::bitmap_allocator< _Tp >`:



Public Types

- typedef `free_list::__mutex_type` **__mutex_type**
- typedef `const _Tp *` **const_pointer**
- typedef `const _Tp &` **const_reference**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **bitmap_allocator** (const [bitmap_allocator](#) &)
- template<typename `_Tp1` >
 bitmap_allocator (const [bitmap_allocator](#)< `_Tp1` > &) throw ()
- pointer [_M_allocate_single_object](#) () throw (std::bad_alloc)
- void [_M_deallocate_single_object](#) (pointer `__p`) throw ()
- const_pointer **address** (const_reference `__r`) const
- pointer **address** (reference `__r`) const
- pointer **allocate** (size_type `__n`, typename [bitmap_allocator](#)< void >::const_pointer)
- pointer **allocate** (size_type `__n`)
- template<typename... `_Args`>
 void **construct** (pointer `__p`, `_Args` &&...`__args`)
- void **construct** (pointer `__p`, const_reference `__data`)
- void **deallocate** (pointer `__p`, size_type `__n`) throw ()
- void **destroy** (pointer `__p`)
- size_type **max_size** () const throw ()

Private Types

- typedef `vector_type::iterator` **iterator**
- typedef `__detail::__mini_vector< value_type >` **vector_type**

Private Member Functions

- void [_M_clear](#) ()
- size_t * [_M_get](#) (size_t `__sz`) throw (std::bad_alloc)
- void [_M_insert](#) (size_t * `__addr`) throw ()

5.36.1 Detailed Description

template<typename _Tp> class __gnu_cxx::bitmap_allocator< _Tp >

Bitmap Allocator, primary template.

Definition at line 686 of file bitmap_allocator.h.

5.36.2 Member Function Documentation

5.36.2.1 template<typename _Tp > pointer __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object () throw (std::bad_alloc) [inline]

Allocates memory for a single object of size sizeof(_Tp).

Exceptions

std::bad_alloc. If memory can not be allocated.

Complexity: Worst case complexity is O(N), but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of O(1)! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 820 of file bitmap_allocator.h.

References __gnu_cxx::__detail::__bit_allocate(), __gnu_cxx::__detail::__num_bitmaps(), and __gnu_cxx::_Bit_scan_forward().

5.36.2.2 template<typename _Tp > void __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object (pointer __p) throw () [inline]

Deallocates memory that belongs to a single object of size sizeof(_Tp).

Complexity: O(lg(N)), but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in O(1) time by the deallocate function.

Definition at line 910 of file bitmap_allocator.h.

References __gnu_cxx::__detail::__bit_free(), __gnu_cxx::__detail::__num_bitmaps(), and std::__rotate().

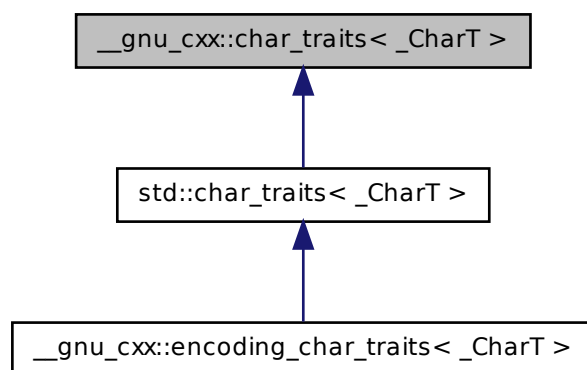
The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

5.37 `__gnu_cxx::char_traits< _CharT >` Struct Template Reference

Base class used to implement [std::char_traits](#).

Inheritance diagram for `__gnu_cxx::char_traits< _CharT >`:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types< _CharT >::int_type` **int_type**
- typedef `_Char_types< _CharT >::off_type` **off_type**
- typedef `_Char_types< _CharT >::pos_type` **pos_type**
- typedef `_Char_types< _CharT >::state_type` **state_type**

Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)

- static int_type **eof** ()
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static std::size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.37.1 Detailed Description

template<typename _CharT> struct `__gnu_cxx::char_traits`< _CharT >

Base class used to implement `std::char_traits`.

Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out [include/ext/pod_char_traits.h](#).

Definition at line 83 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.38 `__gnu_cxx::character`< V, I, S > Struct Template Reference

A POD class that serves as a character abstraction class.

Public Types

- typedef `character`< V, I, S > `char_type`
- typedef I `int_type`
- typedef S `state_type`
- typedef V `value_type`

Static Public Member Functions

- template<typename V2 >
static `char_type` `from` (const V2 &v)
- template<typename V2 >
static V2 `to` (const `char_type` &c)

Public Attributes

- `value_type` `value`

5.38.1 Detailed Description

`template<typename V, typename I, typename S = std::mbstate_t> struct __gnu_cxx::character< V, I, S >`

A POD class that serves as a character abstraction class.

Definition at line 45 of file `pod_char_traits.h`.

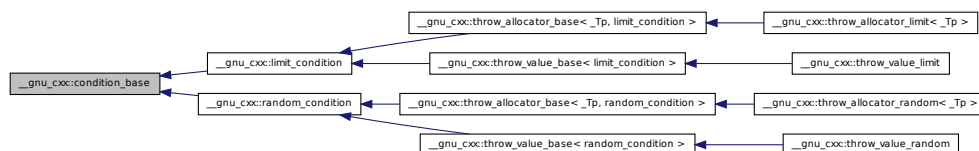
The documentation for this struct was generated from the following file:

- [pod_char_traits.h](#)

5.39 `__gnu_cxx::condition_base` Struct Reference

Base struct for condition policy.

Inheritance diagram for `__gnu_cxx::condition_base`:



5.39.1 Detailed Description

Base struct for condition policy. Requires a public member function with the signature `void throw_conditionally()`

Definition at line 253 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.40 `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >` Struct Template Reference

An [SGI extension](#).

Inherits `__gnu_cxx::_Constant_binary_fun< _Result, _Arg1, _Arg2 >`.

Public Types

- `typedef _Arg1 first_argument_type`
- `typedef _Result result_type`
- `typedef _Arg2 second_argument_type`

Public Member Functions

- `constant_binary_fun` (`const _Result &__v`)
- `const result_type &operator()` (`const _Arg1 &`, `const _Arg2 &`) `const`

Public Attributes

- `_Result _M_val`

5.40.1 Detailed Description

`template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1> struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`

An [SGI extension](#) .

Definition at line 315 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.41 __gnu_cxx::constant_unary_fun< _Result, _Argument > Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Constant_unary_fun< _Result, _Argument >`.

Public Types

- `typedef _Argument argument_type`
- `typedef _Result result_type`

Public Member Functions

- `constant_unary_fun (const _Result &__v)`
- `const result_type & operator() (const _Argument &) const`

Public Attributes

- `result_type _M_val`

5.41.1 Detailed Description

```
template<class _Result, class _Argument = _Result> struct __gnu_cxx::constant_unary_fun< _Result, _Argument >
```

An [SGI extension](#) .

Definition at line 307 of file ext/functional.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.42 __gnu_cxx::constant_void_fun< _Result > Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Constant_void_fun< _Result >`.

Public Types

- typedef `_Result` **result_type**

Public Member Functions

- **constant_void_fun** (const `_Result` &__v)
- const `result_type` & **operator**() () const

Public Attributes

- `result_type` **_M_val**

5.42.1 Detailed Description

```
template<class _Result> struct __gnu_cxx::constant_void_fun< _Result >
```

An [SGI extension](#) .

Definition at line 298 of file ext/functional.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.43 `__gnu_cxx::debug_allocator<_Alloc>` Class Template Reference

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Public Types

- `typedef _Alloc::const_pointer` **const_pointer**
- `typedef _Alloc::const_reference` **const_reference**
- `typedef _Alloc::difference_type` **difference_type**
- `typedef _Alloc::pointer` **pointer**
- `typedef _Alloc::reference` **reference**
- `typedef _Alloc::size_type` **size_type**
- `typedef _Alloc::value_type` **value_type**

Public Member Functions

- `pointer` **allocate** (`size_type __n`)
- `pointer` **allocate** (`size_type __n`, `const void *__hint`)
- `void` **deallocate** (`pointer __p`, `size_type __n`)

5.43.1 Detailed Description

`template<typename _Alloc> class __gnu_cxx::debug_allocator<_Alloc>`

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Definition at line 61 of file `debug_allocator.h`.

The documentation for this class was generated from the following file:

- [debug_allocator.h](#)

Protected Member Functions

- `void _M_allocate_internal_buffer ()`
- `bool _M_convert_to_external (char_type *, streamsize)`
- `void _M_create_pback ()`
- `void _M_destroy_internal_buffer () throw ()`
- `void _M_destroy_pback () throw ()`
- `int _M_get_ext_pos (__state_type &__state)`
- `pos_type _M_seek (off_type __off, ios_base::seekdir __way, __state_type __state)`
- `void _M_set_buffer (streamsize __off)`
- `bool _M_terminate_output ()`
- `void gbump (int __n)`
- `virtual void imbue (const locale &__loc)`
- `virtual int_type overflow (int_type __c=encoding_char_traits<_CharT>::eof())`
- `virtual int_type overflow (int_type=traits_type::eof())`
- `virtual int_type pbackfail (int_type=traits_type::eof())`
- `virtual int_type pbackfail (int_type __c=encoding_char_traits<_CharT>::eof())`
- `void pbump (int __n)`
- `virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)`
- `virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)`
- `virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)`
- `virtual basic_streambuf< char_type, encoding_char_traits<_CharT> > * setbuf (char_type *, streamsize)`
- `void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)`
- `void setp (char_type *__pbeg, char_type *__pend)`
- `virtual streamsize showmanyc ()`
- `virtual int sync ()`
- `virtual int_type uflow ()`
- `virtual int_type underflow ()`
- `virtual streamsize xsgetn (char_type *__s, streamsize __n)`
- `virtual streamsize xsgetn (char_type *__s, streamsize __n)`
- `virtual streamsize xspun (const char_type *__s, streamsize __n)`
- `virtual streamsize xspun (const char_type *__s, streamsize __n)`

- [char_type * eback \(\) const](#)
- [char_type * gptr \(\) const](#)
- [char_type * egptr \(\) const](#)
- [char_type * pbase \(\) const](#)
- [char_type * pptr \(\) const](#)
- [char_type * epptr \(\) const](#)

Protected Attributes

- [char_type * _M_buf](#)
- [bool _M_buf_allocated](#)
- [size_t _M_buf_size](#)
- [const __codecvt_type * _M_codecvt](#)
- [char * _M_ext_buf](#)
- [streamsize _M_ext_buf_size](#)
- [char * _M_ext_end](#)
- [const char * _M_ext_next](#)
- [__file_type _M_file](#)
- [__c_lock _M_lock](#)
- [ios_base::openmode _M_mode](#)
- [bool _M_reading](#)
- [__state_type _M_state_beg](#)
- [__state_type _M_state_cur](#)
- [__state_type _M_state_last](#)
- [bool _M_writing](#)
- [char_type _M_pback](#)
- [char_type * _M_pback_cur_save](#)
- [char_type * _M_pback_end_save](#)
- [bool _M_pback_init](#)

Friends

- [__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-
type __copy_move_a2 \(istreambuf_iterator< _CharT2 >, istreambuf_iterator<
_CharT2 >, _CharT2 *\)](#)
- [streamsize __copy_streambufs_eof \(__streambuf_type *, __streambuf_type *,
bool &\)](#)
- [class basic_ios< char_type, traits_type >](#)
- [class basic_istream< char_type, traits_type >](#)
- [class basic_ostream< char_type, traits_type >](#)

- `__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type` **find** (`istreambuf_iterator< _CharT2 >`, `istreambuf_iterator< _CharT2 >`, `const _CharT2 &`)
- `basic_istream< _CharT2, _Traits2 > & getline` (`basic_istream< _CharT2, _Traits2 > &`, `basic_string< _CharT2, _Traits2, _Alloc > &`, `_CharT2`)
- class **ios_base**
- class **istreambuf_iterator**< **char_type**, **traits_type** >
- `basic_istream< _CharT2, _Traits2 > & operator>>` (`basic_istream< _CharT2, _Traits2 > &`, `_CharT2 *`)
- `basic_istream< _CharT2, _Traits2 > & operator>>` (`basic_istream< _CharT2, _Traits2 > &`, `basic_string< _CharT2, _Traits2, _Alloc > &`)
- class **ostreambuf_iterator**< **char_type**, **traits_type** >
- locale [pubimbue](#) (`const locale &__loc`)
- locale [getloc](#) () `const`
- `__streambuf_type * pubsetbuf` (`char_type *__s`, `streamsize __n`)
- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()
- `char_type * _M_in_beg`
- `char_type * _M_in_cur`
- `char_type * _M_in_end`
- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- locale `_M_buf_locale`

5.44.1 Detailed Description

`template<typename _CharT> class __gnu_cxx::enc_filebuf<_CharT>`

class [enc_filebuf](#).

Definition at line 40 of file `enc_filebuf.h`.

5.44.2 Member Typedef Documentation

5.44.2.1 `typedef basic_streambuf<char_type, traits_type> std::basic_filebuf<_CharT, encoding_char_traits<_CharT> >::__streambuf_type`
[[inherited](#)]

This is a non-standard type.

Reimplemented from [std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >](#).

Definition at line 77 of file fstream.

5.44.2.2 **typedef _CharT std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::char_type [inherited]**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >](#).

Definition at line 71 of file fstream.

5.44.2.3 **typedef traits_type::int_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type [inherited]**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >](#).

Definition at line 73 of file fstream.

5.44.2.4 **typedef traits_type::off_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::off_type [inherited]**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >](#).

Definition at line 75 of file fstream.

5.44.2.5 **template<typename _CharT > typedef traits_type::pos_type __gnu_cxx::enc_filebuf< _CharT >::pos_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 46 of file `enc_filebuf.h`.

5.44.2.6 `template<typename _CharT> typedef encoding_char_traits<_CharT> __gnu_cxx::enc_filebuf<_CharT>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Definition at line 44 of file `enc_filebuf.h`.

5.44.3 Member Function Documentation

5.44.3.1 `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_create_pback() [inline, protected, inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file `fstream`.

5.44.3.2 `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_destroy_pback() throw() [inline, protected, inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file `fstream`.

5.44.3.3 `void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_set_buffer(streamsize __off) [inline, protected, inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically: `__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `eptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 390 of file `fstream`.

5.44.3.4 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::close() [inherited]`

Closes the currently associated file.

Returns

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

5.44.3.5 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eback() const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

5.44.3.6 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::egptr() const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file `streambuf`.

5.44.3.7 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::eptr() const` [`inline`, `protected`, `inherited`]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `pptr()` returns the end pointer for the output sequence

Definition at line 513 of file `streambuf`.

5.44.3.8 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gbump(int __n)` [`inline`, `protected`, `inherited`]

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

5.44.3.9 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::getloc() const` [`inline`, `inherited`]

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

5.44.3.10 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::gptr() const` [`inline`, `protected`, `inherited`]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file `streambuf`.

5.44.3.11 `virtual void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::imbue(const locale &)` [`protected`, `virtual`, `inherited`]

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.


```
5.44.3.12 streamsize std::basic_streambuf<_CharT, encoding_char_traits<
CharT>>::in_avail( ) [inline, inherited]
```

Returns

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

```
5.44.3.13  bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT >
>::is_open() const throw() [inline, inherited]
```

Definition at line 222 of file fstream.

```
5.44.3.14 __filebuf_type* std::basic_filebuf< _CharT, encoding_char_traits<
          _CharT > >::open ( const char * __s, ios_base::openmode __mode
          ) [inherited]
```

Parameters

Returns

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

5.44.3.15 `__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT> >::open (const std::string & __s, ios_base::openmode __mode) [inline, inherited]`

Opens an external file.

Parameters

s The name of the file.
mode The open mode flags.

Returns

this on success, NULL on failure

Definition at line 275 of file fstream.

5.44.3.16 `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::overflow (int_type = traits_type::eof()) [inline, protected, virtual, inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

c An additional character to consume.

Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Definition at line 746 of file streambuf.

5.44.3.17 `virtual int_type std::basic_streambuf<_CharT,
encoding_char_traits<_CharT>>::pbackfail (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

`eof()` on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns `eof()`.

Definition at line 702 of file `streambuf`.

5.44.3.18 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<
_CharT>>::pbase () const [inline, protected,
inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 507 of file `streambuf`.

5.44.3.19 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT
>>::pbump (int __n) [inline, protected, inherited]`

Moving the write position.

Parameters

n The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

5.44.3.20 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pptr() const [inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

5.44.3.21 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubimbue(const locale & __loc) [inline, inherited]`

Entry point for imbue().

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 205 of file streambuf.

5.44.3.22 `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekoff(off_type __off, ios_base::seekdir __way, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline, inherited]`

Entry point for imbue().

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

5.44.3.23 `pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubseekpos (pos_type __sp, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

5.44.3.24 `__streambuf_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsetbuf (char_type * __s, streamsize __n) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.44.3.25 `int std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::pubsync () [inline, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 249 of file streambuf.

5.44.3.26 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sbumpc() [inline, inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file streambuf.

5.44.3.27 `virtual pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::seekoff(off_type, ios_base::seekdir, ios_base::openmode = ios_base::in | ios_base::out) [inline, protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file streambuf.

5.44.3.28 `virtual pos_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::seekpos(pos_type, ios_base::openmode = ios_base::in | ios_base::out) [inline, protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

5.44.3.29 `virtual __streambuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::setbuf(char_type * __s, streamsize __n)` [`protected`, `virtual`, `inherited`]

Manipulates the buffer.

Parameters

s Pointer to a buffer area.

n Size of *s*.

Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

5.44.3.30 `virtual basic_streambuf<char_type,encoding_char_traits<_CharT>>* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setbuf(char_type *, streamsize)` [`inline`, `protected`, `virtual`, `inherited`]

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

5.44.3.31 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setg (char_type * __gbeg, char_type * __gnext, char_type * __gend) [inline, protected, inherited]`

Setting the three read area pointers.

Parameters

gbeg A pointer.
gnext A pointer.
gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

5.44.3.32 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::setp (char_type * __pbeg, char_type * __pend) [inline, protected, inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.
pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file `streambuf`.

5.44.3.33 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetc () [inline, inherited]`

Getting the next character.

Returns

The next character, or `eof`.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

5.44.3.34 `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sgetn(char_type * __s, streamsize __n)`
`[inline, inherited]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

5.44.3.35 `virtual streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::showmanyc()` `[protected, virtual, inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
 [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

5.44.3.36 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::snextc()` `[inline, inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

5.44.3.37 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::sputback(char_type __c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

c The character to push back.

Returns

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file `streambuf`.

5.44.3.38 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::putc(char_type __c) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

c A character to output.

Returns

c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

5.44.3.39 `streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::sputn (const char_type * __s, streamsize __n)`
[inline, inherited]

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.44.3.40 `void std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::stoss ()` [inline, inherited]

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

5.44.3.41 `int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT> >::sungetc ()` [inline, inherited]

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

5.44.3.42 `virtual int std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::sync(void) [protected, virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

5.44.3.43 `virtual int_type std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::uflow() [inline, protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Definition at line 678 of file `streambuf`.

5.44.3.44 `virtual int_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::underflow() [protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>`.

5.44.3.45 `virtual streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::xsgetn (char_type * __s, streamsize __n)` [protected, virtual, inherited]

Multiple character extraction.

Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

5.44.3.46 `virtual streamsize std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::xsputn (const char_type * __s, streamsize __n)` [protected, virtual, inherited]

Multiple character insertion.

Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

5.44.4 Member Data Documentation**5.44.4.1** `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf` **[protected, inherited]**

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

5.44.4.2 `locale std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_locale` **[protected, inherited]**

Current locale setting.

Definition at line 188 of file `streambuf`.

5.44.4.3 `size_t std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_size` **[protected, inherited]**

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 116 of file `fstream`.

5.44.4.4 `char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf` **[protected, inherited]**

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 151 of file `fstream`.

5.44.4.5 `streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf_size` `[protected, inherited]`

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file `fstream`.

5.44.4.6 `const char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_next` `[protected, inherited]`

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 163 of file `fstream`.

5.44.4.7 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_beg` `[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

5.44.4.8 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_cur` `[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

5.44.4.9 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_in_end` `[protected, inherited]`

Entry point for imbue().

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 182 of file streambuf.

5.44.4.10 `ios_base::openmode std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_mode` `[protected, inherited]`

Place to stash in || out || in | out settings for current filebuf.

Definition at line 94 of file fstream.

5.44.4.11 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_beg` `[protected, inherited]`

Entry point for imbue().

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 183 of file streambuf.

5.44.4.12 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_cur` `[protected, inherited]`

Entry point for imbue().

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

5.44.4.13 `char_type* std::basic_streambuf<_CharT, encoding_char_traits<_CharT>>::_M_out_end` [protected, inherited]

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

5.44.4.14 `char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback` [protected, inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 137 of file `fstream`.

5.44.4.15 `char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback_cur_save` [protected, inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 138 of file `fstream`.

5.44.4.16 `char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_end_save` [protected, inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 139 of file `fstream`.

5.44.4.17 `bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_init` [protected, inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 140 of file `fstream`.

5.44.4.18 `bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_reading` [protected, inherited]

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file `fstream`.

The documentation for this class was generated from the following file:

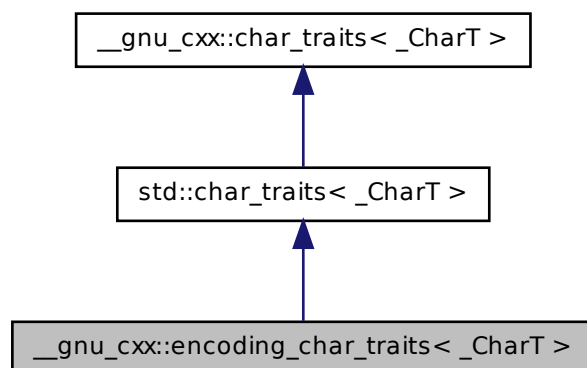
- [enc_filebuf.h](#)

5.45 `__gnu_cxx::encoding_char_traits< _CharT >` Struct Template Reference

[encoding_char_traits](#)

5.45 `__gnu_cxx::encoding_char_traits<_CharT>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::encoding_char_traits<_CharT>`:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types<_CharT>::int_type` **int_type**
- typedef `_Char_types<_CharT>::off_type` **off_type**
- typedef `std::fpos<state_type>` **pos_type**
- typedef `encoding_state` **state_type**

Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static int_type **eof** ()
- static bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static bool **eq_int_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)

- static std::size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.45.1 Detailed Description

template<typename _CharT> struct __gnu_cxx::encoding_char_traits< _CharT >

[encoding_char_traits](#)

Definition at line 210 of file codecvt_specializations.h.

The documentation for this struct was generated from the following file:

- [codecvt_specializations.h](#)

5.46 __gnu_cxx::encoding_state Class Reference

Extension to use iconv for dealing with character encodings.

Public Types

- typedef iconv_t **descriptor_type**

Public Member Functions

- **encoding_state** (const char *__int, const char *__ext, int __ibom=0, int __ebom=0, int __bytes=1)
- **encoding_state** (const [encoding_state](#) &__obj)
- int **character_ratio** () const
- int **external_bom** () const
- const std::string **external_encoding** () const
- bool **good** () const throw ()
- const descriptor_type & **in_descriptor** () const
- int **internal_bom** () const
- const std::string **internal_encoding** () const
- [encoding_state](#) & **operator=** (const [encoding_state](#) &__obj)
- const descriptor_type & **out_descriptor** () const

Protected Member Functions

- void **construct** (const [encoding_state](#) &__obj)
- void **destroy** () throw ()
- void **init** ()

Protected Attributes

- int **_M_bytes**
- int **_M_ext_bom**
- [std::string](#) **_M_ext_enc**
- descriptor_type **_M_in_desc**
- int **_M_int_bom**
- [std::string](#) **_M_int_enc**
- descriptor_type **_M_out_desc**

5.46.1 Detailed Description

Extension to use `iconv` for dealing with character encodings.

Definition at line 50 of file `codecvt_specializations.h`.

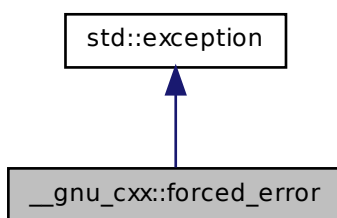
The documentation for this class was generated from the following file:

- [codecvt_specializations.h](#)

5.47 `__gnu_cxx::forced_error` Struct Reference

Thrown by exception safety machinery.

Inheritance diagram for `__gnu_cxx::forced_error`:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.47.1 Detailed Description

Thrown by exception safety machinery.

Definition at line 73 of file `throw_allocator.h`.

5.47.2 Member Function Documentation

5.47.2.1 virtual const char* `std::exception::what` () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_exception](#), [std::bad_alloc](#), [std::bad_cast](#), [std::bad_typeid](#), [std::future_error](#), [std::logic_error](#), [std::runtime_error](#), [std::tr1::bad_weak_ptr](#), and [std::ios_base::failure](#).

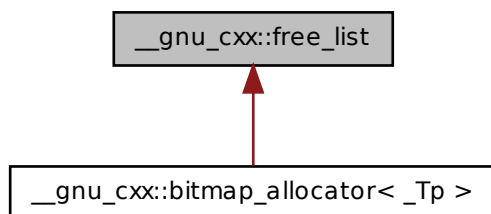
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.48 `__gnu_cxx::free_list` Class Reference

The free list class for managing chunks of memory to be given to and returned by the [bitmap_allocator](#).

Inheritance diagram for `__gnu_cxx::free_list`:



Public Types

- typedef `__mutex` **`mutex_type`**
- typedef `vector_type::iterator` **`iterator`**
- typedef `size_t * value_type`
- typedef `__detail::__mini_vector< value_type >` **`vector_type`**

Public Member Functions

- void `_M_clear()`
- `size_t * _M_get(size_t __sz) throw (std::bad_alloc)`
- void `_M_insert(size_t *__addr) throw ()`

5.48.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the [bitmap_allocator](#).

Definition at line 520 of file `bitmap_allocator.h`.

5.48.2 Member Function Documentation

5.48.2.1 void __gnu_cxx::free_list::_M_clear ()

This function just clears the internal Free List, and gives back all the memory to the OS.

5.48.2.2 size_t* __gnu_cxx::free_list::_M_get (size_t __sz) throw (std::bad_alloc)

This function gets a block of memory of the specified size from the free list.

Parameters

`__sz` The size in bytes of the memory required.

Returns

A pointer to the new memory block of size at least equal to that requested.

5.48.2.3 void __gnu_cxx::free_list::_M_insert (size_t * __addr) throw () [inline]

This function returns the block of memory to the internal free list.

Parameters

`__addr` The pointer to the memory block that was given by a call to the `_M_get` function.

Definition at line 630 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

5.49 __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference

Public Types

- typedef `_Ht::allocator_type` **allocator_type**
- typedef `_Ht::const_iterator` **const_iterator**

- typedef `_Ht::const_pointer` `const_pointer`
- typedef `_Ht::const_reference` `const_reference`
- typedef `_Tp` `data_type`
- typedef `_Ht::difference_type` `difference_type`
- typedef `_Ht::hasher` `hasher`
- typedef `_Ht::iterator` `iterator`
- typedef `_Ht::key_equal` `key_equal`
- typedef `_Ht::key_type` `key_type`
- typedef `_Tp` `mapped_type`
- typedef `_Ht::pointer` `pointer`
- typedef `_Ht::reference` `reference`
- typedef `_Ht::size_type` `size_type`
- typedef `_Ht::value_type` `value_type`

Public Member Functions

- `hash_map` (`size_type __n`)
- `template<class _InputIterator>`
`hash_map` (`_InputIterator __f, _InputIterator __l`)
- `template<class _InputIterator>`
`hash_map` (`_InputIterator __f, _InputIterator __l, size_type __n`)
- `template<class _InputIterator>`
`hash_map` (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf`)
- `template<class _InputIterator>`
`hash_map` (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- `hash_map` (`size_type __n, const hasher &__hf`)
- `hash_map` (`size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type()`)
- iterator `begin` ()
- `const_iterator` `begin` () const
- `size_type` `bucket_count` () const
- void `clear` ()
- `size_type` `count` (const `key_type` &__key) const
- `size_type` `elems_in_bucket` (`size_type __n`) const
- bool `empty` () const
- iterator `end` ()
- `const_iterator` `end` () const
- `pair< iterator, iterator >` `equal_range` (const `key_type` &__key)
- `pair< const_iterator, const_iterator >` `equal_range` (const `key_type` &__key) const
- `size_type` `erase` (const `key_type` &__key)

- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator >
void **insert** (_InputIterator __f, _InputIterator __l)
- pair< iterator, bool > **insert** (const value_type &__obj)
- pair< iterator, bool > **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- _Tp & **operator[]** (const key_type &__key)
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_map &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al >
bool **operator==** (const hash_map< _K1, _T1, _HF, _EqK, _Al > &, const
hash_map< _K1, _T1, _HF, _EqK, _Al > &)

5.49.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey
= equal_to<_Key>, class _Alloc = allocator<_Tp>> class __gnu_cxx::hash_
map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 76 of file hash_map.

The documentation for this class was generated from the following file:

- [hash_map](#)

5.50 `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >` Class Template Reference

Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Ht::const_pointer const_pointer`
- `typedef _Ht::const_reference const_reference`
- `typedef _Tp data_type`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Tp mapped_type`
- `typedef _Ht::pointer pointer`
- `typedef _Ht::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

Public Member Functions

- `hash_multimap (size_type __n)`
- `template<class _InputIterator >
hash_multimap (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator >
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator >
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const
hasher &__hf)`
- `template<class _InputIterator >
hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const
hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_
type())`
- `hash_multimap (size_type __n, const hasher &__hf)`
- `hash_multimap (size_type __n, const hasher &__hf, const key_equal &__eq,
const allocator_type &__a=allocator_type())`
- `iterator begin ()`
- `const_iterator begin () const`
- `size_type bucket_count () const`

- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- [pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- [pair](#)< iterator, iterator > **equal_range** (const key_type &__key)
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator >
void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert** (const [value_type](#) &__obj)
- iterator **insert_noresize** (const [value_type](#) &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** ([hash_multimap](#) &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al >
bool **operator==** (const [hash_multimap](#)< _K1, _T1, _HF, _EqK, _Al > &, const [hash_multimap](#)< _K1, _T1, _HF, _EqK, _Al > &)

5.50.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey
= equal_to<_Key>, class _Alloc = allocator<_Tp>>> class __gnu_cxx::hash-
multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
```

This is an SGI extension.

5.51 `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>` Class Template Reference 921

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 289 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash_map](#)

5.51 `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>` Class Template Reference

Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::const_iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

Public Member Functions

- `hash_multiset (size_type __n)`
- `template<class _InputIterator >
hash_multiset (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator >
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator >
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)`
- `template<class _InputIterator >
hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`

- **hash_multiset** (size_type __n, const hasher &__hf)
- **hash_multiset** (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())
- iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- [pair](#)< iterator, iterator > **equal_range** (const key_type &__key) const
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- size_type **erase** (const key_type &__key)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_func** () const
- template<class _InputIterator >
void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert** (const value_type &__obj)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** ([hash_multiset](#) &hs)

Friends

- template<class _Val, class _HF, class _EqK, class _Al >
bool **operator==** (const [hash_multiset](#)< _Val, _HF, _EqK, _Al > &, const [hash_multiset](#)< _Val, _HF, _EqK, _Al > &)

5.51.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey =
equal_to<_Value>, class _Alloc = allocator<_Value>> class __gnu_cxx::hash_-
multiset< _Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 284 of file hash_set.

The documentation for this class was generated from the following file:

- [hash_set](#)

5.52 __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Alloc::const_pointer **const_pointer**
- typedef _Alloc::const_reference **const_reference**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::const_iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Alloc::pointer **pointer**
- typedef _Alloc::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- **hash_set** (size_type __n)
- template<class _InputIterator >
hash_set (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator >
hash_set (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator >
hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &_hf)
- template<class _InputIterator >
hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &_hf, const key_equal &_eq, const allocator_type &_a=allocator_type())

- **hash_set** (size_type __n, const hasher &__hf)
- **hash_set** (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())
- iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- [pair](#)< iterator, iterator > **equal_range** (const key_type &__key) const
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- size_type **erase** (const key_type &__key)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_func** () const
- template<class _InputIterator >
void **insert** (_InputIterator __f, _InputIterator __l)
- [pair](#)< iterator, bool > **insert** (const value_type &__obj)
- [pair](#)< iterator, bool > **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** ([hash_set](#) &__hs)

Friends

- template<class _Val, class _HF, class _EqK, class _Al >
bool **operator==** (const [hash_set](#)< _Val, _HF, _EqK, _Al > &, const [hash_set](#)< _Val, _HF, _EqK, _Al > &)

5.52.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey =
equal_to<_Value>, class _Alloc = allocator<_Value>> class __gnu_cxx::hash_-
set< _Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 83 of file `hash_set`.

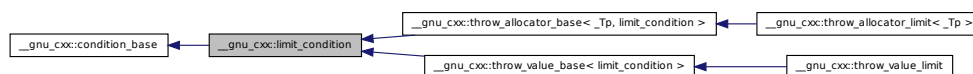
The documentation for this class was generated from the following file:

- [hash_set](#)

5.53 `__gnu_cxx::limit_condition` Struct Reference

Base class for incremental control and throw.

Inheritance diagram for `__gnu_cxx::limit_condition`:

**Classes**

- struct [always_adjustor](#)
Always enter the condition.
- struct [limit_adjustor](#)
*Enter the *n*th condition.*
- struct [never_adjustor](#)
Never enter the condition.

Static Public Member Functions

- static `size_t` & **count** ()
- static `size_t` & **limit** ()
- static void **set_limit** (const `size_t` __l)
- static void **throw_conditionally** ()

5.53.1 Detailed Description

Base class for incremental control and throw.

Definition at line 262 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.54 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

5.54.1 Detailed Description

Always enter the condition.

Definition at line 286 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.55 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Enter the nth condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

Public Member Functions

- `limit_adjustor` (const size_t __l)

5.55.1 Detailed Description

Enter the nth condition.

Definition at line 292 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.56 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

5.56.1 Detailed Description

Never enter the condition.

Definition at line 280 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.57 `__gnu_cxx::malloc_allocator< _Tp >` Class Template Reference

An allocator that uses `malloc`.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls `malloc`
- all deallocation calls `free`.

Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- **malloc_allocator** (const [malloc_allocator](#) &) throw ()
- template<typename _Tp1 >
 malloc_allocator (const [malloc_allocator](#)< _Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, const void * = 0)
- void **construct** (pointer __p, const _Tp & __val)
- template<typename... _Args>
 void **construct** (pointer __p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type)
- void **destroy** (pointer __p)
- size_type **max_size** () const throw ()

5.57.1 Detailed Description

template<typename _Tp> class [__gnu_cxx::malloc_allocator](#)< _Tp >

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free.

Definition at line 52 of file [malloc_allocator.h](#).

The documentation for this class was generated from the following file:

- [malloc_allocator.h](#)

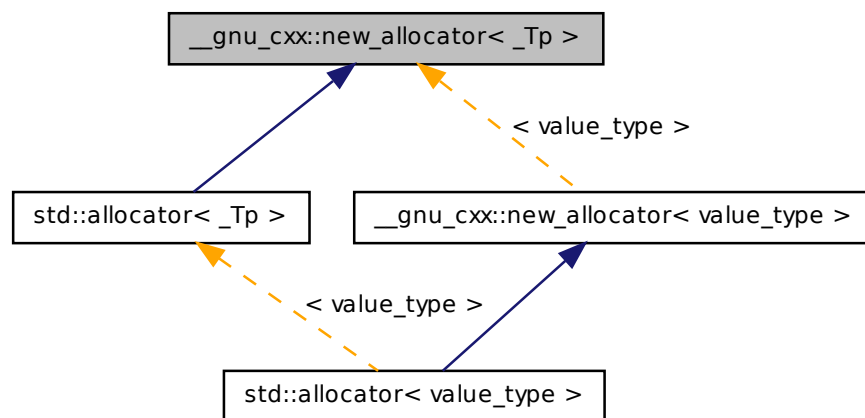
5.58 [__gnu_cxx::new_allocator](#)< _Tp > Class Template Reference

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `new_allocator (const new_allocator &) throw ()`
- `template<typename _Tp1 >
 new_allocator (const new_allocator<_Tp1 > &) throw ()`
- `pointer address (reference __x) const`
- `const_pointer address (const_reference __x) const`
- `pointer allocate (size_type __n, const void *=0)`
- `void construct (pointer __p, const _Tp &__val)`

- `template<typename... _Args>`
`void construct (pointer __p, _Args &&...__args)`
- `void deallocate (pointer __p, size_type)`
- `void destroy (pointer __p)`
- `size_type max_size () const throw ()`

5.58.1 Detailed Description

`template<typename _Tp> class __gnu_cxx::new_allocator< _Tp >`

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete.

Definition at line 52 of file `new_allocator.h`.

The documentation for this class was generated from the following file:

- [new_allocator.h](#)

5.59 __gnu_cxx::project1st< _Arg1, _Arg2 > Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::_Project1st< _Arg1, _Arg2 >`.

Public Types

- `typedef _Arg1 first_argument_type`
- `typedef _Result result_type`
- `typedef _Arg2 second_argument_type`

Public Member Functions

- `_Arg1 operator() (const _Arg1 &__x, const _Arg2 &) const`

5.59.1 Detailed Description

```
template<class _Arg1, class _Arg2> struct __gnu_cxx::project1st< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 232 of file `ext/functional`.

5.59.2 Member Typedef Documentation

5.59.2.1 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`
`>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.59.2.2 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`
`>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.59.2.3 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`
`>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.60 `__gnu_cxx::project2nd< _Arg1, _Arg2 >` Struct Template Reference

An [SGI extension](#) .

Inherits `__gnu_cxx::Project2nd< _Arg1, _Arg2 >`.

Public Types

- typedef `_Arg1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Arg2` [second_argument_type](#)

Public Member Functions

- `_Arg2 operator()` (`const _Arg1 &`, `const _Arg2 &``__y`) `const`

5.60.1 Detailed Description

`template<class _Arg1, class _Arg2> struct __gnu_cxx::project2nd< _Arg1, _Arg2 >`

An [SGI extension](#) .

Definition at line 236 of file `ext/functional`.

5.60.2 Member Typedef Documentation

5.60.2.1 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`
`>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.60.2.2 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`
`>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.60.2.3 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`
`>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

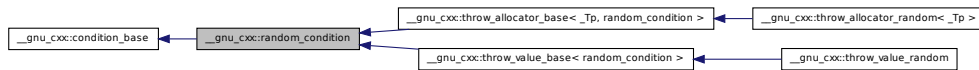
The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.61 `__gnu_cxx::random_condition` Struct Reference

Base class for random probability control and throw.

Inheritance diagram for `__gnu_cxx::random_condition`:



Classes

- struct [always_adjustor](#)
Always enter the condition.
- struct [group_adjustor](#)
Group condition.
- struct [never_adjustor](#)
Never enter the condition.

Public Member Functions

- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

5.61.1 Detailed Description

Base class for random probability control and throw.

Definition at line 334 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.62 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

Always enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

5.62.1 Detailed Description

Always enter the condition.

Definition at line 367 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.63 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

Group condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

Public Member Functions

- `group_adjustor` (size_t size)

5.63.1 Detailed Description

Group condition.

Definition at line 352 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.64 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Never enter the condition.

Inherits `__gnu_cxx::random_condition::adjustor_base`.

5.64.1 Detailed Description

Never enter the condition.

Definition at line 361 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.65 `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >` Struct Template Reference

Inherits `std::_Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`.

Public Types

- `typedef _Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef const _Rb_tree_node< _Val > * _Const_Link_type`
- `typedef _Rb_tree_node< _Val > * _Link_type`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Rb_tree_const_iterator< value_type > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rb_tree_iterator< value_type > iterator`

- typedef `_Key` **key_type**
- typedef `value_type *` **pointer**
- typedef `value_type &` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `_Val` **value_type**

Public Member Functions

- **rb_tree** (const `_Compare` &__comp=_Compare(), const `allocator_type` &__a=allocator_type())
- bool **__rb_verify** () const
- const `_Node_allocator` & **_M_get_Node_allocator** () const
- `_Node_allocator` & **_M_get_Node_allocator** ()
- `iterator` **_M_insert_equal** (const `value_type` &__x)
- template<typename `_InputIterator` >
void **_M_insert_equal** (`_InputIterator` __first, `_InputIterator` __last)
- template<class `_II` >
void **_M_insert_equal** (`_II` __first, `_II` __last)
- `iterator` **_M_insert_equal_** (const `iterator` __position, const `value_type` &__x)
- `pair`< `iterator`, bool > **_M_insert_unique** (const `value_type` &__x)
- template<class `_II` >
void **_M_insert_unique** (`_II` __first, `_II` __last)
- template<typename `_InputIterator` >
void **_M_insert_unique** (`_InputIterator` __first, `_InputIterator` __last)
- `iterator` **_M_insert_unique_** (const `iterator` __position, const `value_type` &__x)
- const `iterator` **begin** () const
- `iterator` **begin** ()
- void **clear** ()
- `size_type` **count** (const `key_type` &__k) const
- bool **empty** () const
- `iterator` **end** ()
- const `iterator` **end** () const
- `pair`< const `iterator`, const `iterator` > **equal_range** (const `key_type` &__k) const
- `pair`< `iterator`, `iterator` > **equal_range** (const `key_type` &__k)
- void **erase** (const `key_type` *__first, const `key_type` *__last)
- const `iterator` **erase** (const `iterator` __first, const `iterator` __last)
- `iterator` **erase** (`iterator` __position)
- const `iterator` **erase** (const `iterator` __position)
- `size_type` **erase** (const `key_type` &__x)
- `iterator` **erase** (`iterator` __first, `iterator` __last)

- iterator **find** (const key_type &__k)
- const_iterator **find** (const key_type &__k) const
- allocator_type **get_allocator** () const
- _Compare **key_comp** () const
- const_iterator **lower_bound** (const key_type &__k) const
- iterator **lower_bound** (const key_type &__k)
- size_type **max_size** () const
- [reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rbegin** () const
- [const_reverse_iterator](#) **rend** () const
- [reverse_iterator](#) **rend** ()
- size_type **size** () const
- void **swap** (_Rb_tree &__t)
- iterator **upper_bound** (const key_type &__k)
- const_iterator **upper_bound** (const key_type &__k) const

Protected Types

- typedef _Rb_tree_node_base * **_Base_ptr**
- typedef const _Rb_tree_node_base * **_Const_Base_ptr**

Protected Member Functions

- _Link_type **_M_begin** ()
- _Const_Link_type **_M_begin** () const
- _Link_type **_M_clone_node** (_Const_Link_type __x)
- template<typename... _Args>
_Link_type **_M_create_node** (_Args &&...__args)
- void **_M_destroy_node** (_Link_type __p)
- _Const_Link_type **_M_end** () const
- _Link_type **_M_end** ()
- _Link_type **_M_get_node** ()
- _Base_ptr & **_M_leftmost** ()
- _Const_Base_ptr **_M_leftmost** () const
- void **_M_put_node** (_Link_type __p)
- _Base_ptr & **_M_rightmost** ()
- _Const_Base_ptr **_M_rightmost** () const
- _Base_ptr & **_M_root** ()
- _Const_Base_ptr **_M_root** () const

Static Protected Member Functions

- static const `_Key` & `_S_key` (`_Const_Link_type __x`)
- static const `_Key` & `_S_key` (`_Const_Base_ptr __x`)
- static `_Link_type` `_S_left` (`_Base_ptr __x`)
- static `_Const_Link_type` `_S_left` (`_Const_Base_ptr __x`)
- static `_Base_ptr` `_S_maximum` (`_Base_ptr __x`)
- static `_Const_Base_ptr` `_S_maximum` (`_Const_Base_ptr __x`)
- static `_Base_ptr` `_S_minimum` (`_Base_ptr __x`)
- static `_Const_Base_ptr` `_S_minimum` (`_Const_Base_ptr __x`)
- static `_Const_Link_type` `_S_right` (`_Const_Base_ptr __x`)
- static `_Link_type` `_S_right` (`_Base_ptr __x`)
- static const_reference `_S_value` (`_Const_Link_type __x`)
- static const_reference `_S_value` (`_Const_Base_ptr __x`)

Protected Attributes

- `_Rb_tree_impl<_Compare>` `_M_impl`

5.65.1 Detailed Description

```
template<class _Key, class _Value, class _KeyOfValue, class _Compare, class
_Alloc = allocator<_Value>> struct __gnu_cxx::rb_tree< _Key, _Value, _-
KeyOfValue, _Compare, _Alloc >
```

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 78 of file `rb_tree`.

The documentation for this struct was generated from the following file:

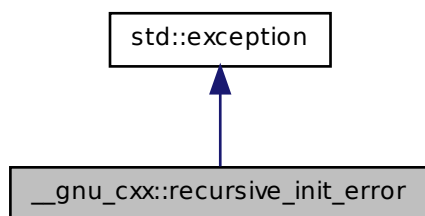
- [rb_tree](#)

5.66 __gnu_cxx::recursive_init_error Class Reference

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Inheritance diagram for `__gnu_cxx::recursive_init_error`:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.66.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined. Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 620 of file `cxxabi.h`.

5.66.2 Member Function Documentation

5.66.2.1 virtual const char* `std::exception::what` () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_exception](#), [std::bad_alloc](#), [std::bad_cast](#), [std::bad_typeid](#), [std::future_error](#), [std::logic_error](#), [std::runtime_error](#), [std::tr1::bad_weak_ptr](#), and

[std::ios_base::failure](#).

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

5.67 `__gnu_cxx::rope< _CharT, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::Rope_base< _CharT, _Alloc >`.

Public Types

- `typedef _Rope_RopeConcatenation< _CharT, _Alloc > __C`
- `typedef _Rope_RopeFunction< _CharT, _Alloc > __F`
- `typedef _Rope_RopeLeaf< _CharT, _Alloc > __L`
- `typedef _Rope_RopeSubstring< _CharT, _Alloc > __S`
- `typedef _Alloc::template rebind< __C >::other _CAlloc`
- `typedef _Alloc::template rebind< _CharT >::other _DataAlloc`
- `typedef _Alloc::template rebind< __F >::other _FAlloc`
- `typedef _Alloc::template rebind< __L >::other _LAlloc`
- `typedef _Alloc::template rebind< __S >::other _SAlloc`
- `typedef _Rope_const_iterator< _CharT, _Alloc > const_iterator`
- `typedef const _CharT * const_pointer`
- `typedef _CharT const_reference`
- `typedef std::reverse_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rope_iterator< _CharT, _Alloc > iterator`
- `typedef _Rope_char_ptr_proxy< _CharT, _Alloc > pointer`
- `typedef _Rope_char_ref_proxy< _CharT, _Alloc > reference`
- `typedef std::reverse_iterator< iterator > reverse_iterator`
- `typedef size_t size_type`
- `typedef _CharT value_type`

Public Member Functions

- **rope** (const _CharT * __s, const allocator_type & __a=allocator_type())
- **rope** (const _CharT * __s, size_t __len, const allocator_type & __a=allocator_type())
- **rope** (const iterator & __s, const iterator & __e, const allocator_type & __a=allocator_type())

- **rope** (`_CharT __c`, `const allocator_type &__a=allocator_type()`)
- **rope** (`size_t __n`, `_CharT __c`, `const allocator_type &__a=allocator_type()`)
- **rope** (`const allocator_type &__a=allocator_type()`)
- **rope** (`const _CharT *__s`, `const _CharT *__e`, `const allocator_type &__a=allocator_type()`)
- **rope** (`char_producer<_CharT> *__fn`, `size_t __len`, `bool __delete_fn`, `const allocator_type &__a=allocator_type()`)
- **rope** (`const rope &__x`, `const allocator_type &__a=allocator_type()`)
- **rope** (`const const_iterator &__s`, `const const_iterator &__e`, `const allocator_type &__a=allocator_type()`)
- `allocator_type &_M_get_allocator ()`
- `const allocator_type &_M_get_allocator () const`
- **rope** & **append** (`const _CharT *__iter`, `size_t __n`)
- **rope** & **append** (`const _CharT *__c_string`)
- **rope** & **append** (`const _CharT *__s`, `const _CharT *__e`)
- **rope** & **append** (`const_iterator __s`, `const_iterator __e`)
- **rope** & **append** (`_CharT __c`)
- **rope** & **append** ()
- **rope** & **append** (`const rope &__y`)
- **rope** & **append** (`size_t __n`, `_CharT __c`)
- `void apply_to_pieces (size_t __begin`, `size_t __end`, `_Rope_char_consumer<_CharT> &__c`) `const`
- `_CharT at (size_type __pos)` `const`
- `_CharT back ()` `const`
- `void balance ()`
- `const_iterator begin ()` `const`
- `const_iterator begin ()`
- `const _CharT * c_str ()` `const`
- `void clear ()`
- `int compare (const rope &__y)` `const`
- `const_iterator const_begin ()` `const`
- `const_iterator const_end ()` `const`
- `const_reverse_iterator const_rbegin ()` `const`
- `const_reverse_iterator const_rend ()` `const`
- `void copy (_CharT *__buffer)` `const`
- `size_type copy (size_type __pos`, `size_type __n`, `_CharT *__buffer)` `const`
- `void delete_c_str ()`
- `void dump ()`
- `bool empty ()` `const`
- `const_iterator end ()` `const`
- `const_iterator end ()`
- `void erase (size_t __p`, `size_t __n)`
- `void erase (size_t __p)`

- iterator **erase** (const iterator &__p, const iterator &__q)
- iterator **erase** (const iterator &__p)
- size_type **find** (_CharT __c, size_type __pos=0) const
- size_type **find** (const _CharT *__s, size_type __pos=0) const
- _CharT **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- iterator **insert** (const iterator &__p, const [rope](#) &__r)
- iterator **insert** (const iterator &__p, size_t __n, _CharT __c)
- iterator **insert** (const iterator &__p, _CharT __c)
- iterator **insert** (const iterator &__p)
- iterator **insert** (const iterator &__p, const _CharT *c_string)
- iterator **insert** (const iterator &__p, const _CharT *__i, size_t __n)
- iterator **insert** (const iterator &__p, const const_iterator &__i, const const_iterator &__j)
- iterator **insert** (const iterator &__p, const iterator &__i, const iterator &__j)
- void **insert** (size_t __p, const const_iterator &__i, const const_iterator &__j)
- void **insert** (size_t __p, size_t __n, _CharT __c)
- void **insert** (size_t __p, const _CharT *__i, size_t __n)
- void **insert** (size_t __p)
- void **insert** (size_t __p, const [rope](#) &__r)
- void **insert** (size_t __p, _CharT __c)
- void **insert** (size_t __p, const _CharT *__c_string)
- void **insert** (size_t __p, const _CharT *__i, const _CharT *__j)
- void **insert** (size_t __p, const iterator &__i, const iterator &__j)
- size_type **length** () const
- size_type **max_size** () const
- iterator **mutable_begin** ()
- iterator **mutable_end** ()
- [reverse_iterator](#) **mutable_rbegin** ()
- reference **mutable_reference_at** (size_type __pos)
- [reverse_iterator](#) **mutable_rend** ()
- [rope](#) & **operator=** (const [rope](#) &__x)
- _CharT **operator[]** (size_type __pos) const
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (_CharT __x)
- void **push_front** (_CharT __x)
- [const_reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rbegin** () const
- [const_reverse_iterator](#) **rend** ()
- [const_reverse_iterator](#) **rend** () const
- void **replace** (size_t __p, const iterator &__i, const iterator &__j)

- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, const _CharT *__j)
- void **replace** (size_t __p, const _CharT *__i, size_t __i_len)
- void **replace** (const iterator &__p, const iterator &__q, _CharT __c)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__c_string)
- void **replace** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, iterator __i, iterator __j)
- void **replace** (const iterator &__p, const [rope](#) &__r)
- void **replace** (const iterator &__p, const iterator &__q, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, size_t __n, const [rope](#) &__r)
- void **replace** (size_t __p, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const iterator &__q, const [rope](#) &__r)
- void **replace** (const iterator &__p, _CharT __c)
- void **replace** (const iterator &__p, const _CharT *__i, size_t __n)
- void **replace** (size_t __p, size_t __n, const _CharT *__c_string)
- void **replace** (size_t __p, size_t __n, _CharT __c)
- void **replace** (size_t __p, size_t __n, const _CharT *__i, size_t __i_len)
- void **replace** (size_t __p, const [rope](#) &__r)
- void **replace** (size_t __p, const _CharT *__c_string)
- void **replace** (size_t __p, size_t __n, const iterator &__i, const iterator &__j)
- void **replace** (size_t __p, size_t __n, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, _CharT __c)
- void **replace** (const iterator &__p, const _CharT *__c_string)
- void **replace** (size_t __p, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, size_t __n, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const iterator &__q, const iterator &__i, const iterator &__j)
- void **replace** (const iterator &__p, const iterator __i, const_iterator __j)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, size_t __n)
- const _CharT * **replace_with_c_str** ()
- size_type **size** () const
- [rope](#)<_CharT, _Alloc> **substr** (const_iterator __start)
- [rope](#) **substr** (const_iterator __start, const_iterator __end) const
- [rope](#) **substr** (iterator __start) const
- [rope](#) **substr** (size_t __start, size_t __len=1) const
- [rope](#) **substr** (iterator __start, iterator __end) const
- void **swap** ([rope](#) &__b)

Static Public Member Functions

- static `__C * _C_allocate (size_t __n)`
- static void `_C_deallocate (__C *__p, size_t __n)`
- static `_CharT * _Data_allocate (size_t __n)`
- static void `_Data_deallocate (_CharT *__p, size_t __n)`
- static `__F * _F_allocate (size_t __n)`
- static void `_F_deallocate (__F *__p, size_t __n)`
- static `__L * _L_allocate (size_t __n)`
- static void `_L_deallocate (__L *__p, size_t __n)`
- static `__S * _S_allocate (size_t __n)`
- static void `_S_deallocate (__S *__p, size_t __n)`

Public Attributes

- `_RopeRep * _M_tree_ptr`

Static Public Attributes

- static const size_type `npos`

Protected Types

- enum { `_S_copy_max` }
- typedef `_Rope_base< _CharT, _Alloc > _Base`
- typedef `_CharT * _Cstrptr`
- typedef `_Rope_RopeConcatenation< _CharT, _Alloc > _RopeConcatenation`
- typedef `_Rope_RopeFunction< _CharT, _Alloc > _RopeFunction`
- typedef `_Rope_RopeLeaf< _CharT, _Alloc > _RopeLeaf`
- typedef `_Rope_RopeRep< _CharT, _Alloc > _RopeRep`
- typedef `_Rope_RopeSubstring< _CharT, _Alloc > _RopeSubstring`
- typedef `_Rope_self_destruct_ptr< _CharT, _Alloc > _Self_destruct_ptr`
- typedef `_Base::allocator_type allocator_type`

Static Protected Member Functions

- static size_t `_S_allocated_capacity (size_t __n)`
- static bool `_S_apply_to_pieces (_Rope_char_consumer< _CharT > &__c, const _RopeRep * __r, size_t __begin, size_t __end)`
- static `_RopeRep * _S_concat (_RopeRep * __left, _RopeRep * __right)`

- static `_RopeRep * _S_concat_char_iter` (`_RopeRep * __r`, const `_CharT * __iter`, `size_t __slen`)
- static `_RopeRep * _S_destr_concat_char_iter` (`_RopeRep * __r`, const `_CharT * __iter`, `size_t __slen`)
- static `_RopeLeaf * _S_destr_leaf_concat_char_iter` (`_RopeLeaf * __r`, const `_CharT * __iter`, `size_t __slen`)
- static `_CharT _S_fetch` (`_RopeRep * __r`, `size_type __pos`)
- static `_CharT * _S_fetch_ptr` (`_RopeRep * __r`, `size_type __pos`)
- static bool `_S_is0` (`_CharT __c`)
- static `_RopeLeaf * _S_leaf_concat_char_iter` (`_RopeLeaf * __r`, const `_CharT * __iter`, `size_t __slen`)
- static `_RopeConcatenation * _S_new_RopeConcatenation` (`_RopeRep * __left`, `_RopeRep * __right`, `allocator_type & __a`)
- static `_RopeFunction * _S_new_RopeFunction` (`char_producer<_CharT> * __f`, `size_t __size`, bool `__d`, `allocator_type & __a`)
- static `_RopeLeaf * _S_new_RopeLeaf` (`_CharT * __s`, `size_t __size`, `allocator_type & __a`)
- static `_RopeSubstring * _S_new_RopeSubstring` (`_Rope_RopeRep<_CharT, _Alloc> * __b`, `size_t __s`, `size_t __l`, `allocator_type & __a`)
- static void `_S_ref` (`_RopeRep * __t`)
- static `_RopeLeaf * _S_RopeLeaf_from_unowned_char_ptr` (const `_CharT * __s`, `size_t __size`, `allocator_type & __a`)
- static `size_t _S_rounded_up_size` (`size_t __n`)
- static `_RopeRep * _S_substring` (`_RopeRep * __base`, `size_t __start`, `size_t __endp1`)
- static `_RopeRep * _S_tree_concat` (`_RopeRep * __left`, `_RopeRep * __right`)
- static void `_S_unref` (`_RopeRep * __t`)
- static `_RopeRep * replace` (`_RopeRep * __old`, `size_t __pos1`, `size_t __pos2`, `_RopeRep * __r`)

Static Protected Attributes

- static `_CharT _S_empty_c_str` [1]

Friends

- class `_Rope_char_ptr_proxy<_CharT, _Alloc>`
- class `_Rope_char_ref_proxy<_CharT, _Alloc>`
- class `_Rope_const_iterator<_CharT, _Alloc>`
- class `_Rope_iterator<_CharT, _Alloc>`
- class `_Rope_iterator_base<_CharT, _Alloc>`
- struct `_Rope_RopeRep<_CharT, _Alloc>`

- struct **_Rope_RopeSubstring**< **_CharT**, **_Alloc** >
- template<class **_CharT2** , class **_Alloc2** >
rope< **_CharT2**, **_Alloc2** > **operator+** (const **rope**< **_CharT2**, **_Alloc2** > &__-
left, const **rope**< **_CharT2**, **_Alloc2** > &__right)
- template<class **_CharT2** , class **_Alloc2** >
rope< **_CharT2**, **_Alloc2** > **operator+** (const **rope**< **_CharT2**, **_Alloc2** > &__-
left, **_CharT2** __right)
- template<class **_CharT2** , class **_Alloc2** >
rope< **_CharT2**, **_Alloc2** > **operator+** (const **rope**< **_CharT2**, **_Alloc2** > &__-
left, const **_CharT2** *__right)

5.67.1 Detailed Description

template<class **_CharT**, class **_Alloc**> class **__gnu_cxx::rope**< **_CharT**, **_Alloc** >

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 1508 of file rope.

The documentation for this class was generated from the following files:

- [rope](#)
- [ropeimpl.h](#)

5.68 **__gnu_cxx::select1st**< **_Pair** > Struct Template Reference

An [SGI extension](#) .

Inherits `std::_Select1st< _Pair >`.

Public Types

- typedef **_Pair** [argument_type](#)
- typedef **_Pair::first_type** [result_type](#)

Public Member Functions

- `_Pair::first_type & operator() (_Pair &__x) const`
- `const _Pair::first_type & operator() (const _Pair &__x) const`

5.68.1 Detailed Description

`template<class _Pair> struct __gnu_cxx::select1st< _Pair >`

An [SGI extension](#) .

Definition at line 198 of file `ext/functional`.

5.68.2 Member Typedef Documentation

5.68.2.1 `typedef _Pair std::unary_function< _Pair , _Pair::first_type >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.68.2.2 `typedef _Pair::first_type std::unary_function< _Pair , _Pair::first_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.69 `__gnu_cxx::select2nd< _Pair >` Struct Template Reference

An [SGI extension](#) .

Inherits `std::_Select2nd< _Pair >`.

Public Types

- `typedef _Pair argument_type`

- `typedef _Pair::second_type result_type`

Public Member Functions

- `_Pair::second_type & operator() (_Pair &__x) const`
- `const _Pair::second_type & operator() (const _Pair &__x) const`

5.69.1 Detailed Description

`template<class _Pair> struct __gnu_cxx::select2nd< _Pair >`

An [SGI extension](#) .

Definition at line 202 of file `ext/functional`.

5.69.2 Member Typedef Documentation

5.69.2.1 `typedef _Pair std::unary_function< _Pair , _Pair::second_type >::argument_type \[inherited\]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.69.2.2 `typedef _Pair::second_type std::unary_function< _Pair , _Pair::second_type >::result_type \[inherited\]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

5.70 `__gnu_cxx::slist< _Tp, _Alloc >` Class Template Reference

Inherits `__gnu_cxx::_Slist_base< _Tp, _Alloc >`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Slist_iterator<_Tp, const _Tp &, const _Tp * > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Slist_iterator<_Tp, _Tp &, _Tp * > iterator`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `slist (const allocator_type &__a=allocator_type())`
- `slist (size_type __n)`
- `template<class _InputIterator >`
`slist (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `slist (size_type __n, const value_type &__x, const allocator_type &__a=allocator_type())`
- `slist (const slist &__x)`
- `template<class _Integer >`
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<class _InputIterator >`
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_fill_assign (size_type __n, const _Tp &__val)`
- `void assign (size_type __n, const _Tp &__val)`
- `template<class _InputIterator >`
`void assign (_InputIterator __first, _InputIterator __last)`
- `iterator before_begin ()`
- `const_iterator before_begin () const`
- `iterator begin ()`
- `const_iterator begin () const`
- `void clear ()`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `iterator erase (iterator __pos)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase_after (iterator __pos)`

- iterator **erase_after** (iterator __before_first, iterator __last)
- reference **front** ()
- const_reference **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (iterator __pos, const value_type &__x)
- iterator **insert** (iterator __pos)
- void **insert** (iterator __pos, size_type __n, const value_type &__x)
- template<class _InIterator >
void **insert** (iterator __pos, _InIterator __first, _InIterator __last)
- iterator **insert_after** (iterator __pos)
- void **insert_after** (iterator __pos, size_type __n, const value_type &__x)
- template<class _InIterator >
void **insert_after** (iterator __pos, _InIterator __first, _InIterator __last)
- iterator **insert_after** (iterator __pos, const value_type &__x)
- size_type **max_size** () const
- void **merge** (slist &__x)
- template<class _StrictWeakOrdering >
void **merge** (slist &, _StrictWeakOrdering)
- slist & **operator=** (const slist &__x)
- void **pop_front** ()
- iterator **previous** (const_iterator __pos)
- const_iterator **previous** (const_iterator __pos) const
- void **push_front** ()
- void **push_front** (const value_type &__x)
- void **remove** (const_Tp &__val)
- template<class _Predicate >
void **remove_if** (_Predicate __pred)
- void **resize** (size_type new_size, const_Tp &__x)
- void **resize** (size_type new_size)
- void **reverse** ()
- size_type **size** () const
- void **sort** ()
- template<class _StrictWeakOrdering >
void **sort** (_StrictWeakOrdering __comp)
- void **splice** (iterator __pos, slist &__x, iterator __i)
- void **splice** (iterator __pos, slist &__x)
- void **splice** (iterator __pos, slist &__x, iterator __first, iterator __last)
- void **splice_after** (iterator __pos, slist &__x)
- void **splice_after** (iterator __pos, iterator __prev)
- void **splice_after** (iterator __pos, iterator __before_first, iterator __before_last)
- void **swap** (slist &__x)
- template<class _BinaryPredicate >
void **unique** (_BinaryPredicate __pred)
- void **unique** ()

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 951

Private Types

- `typedef _Alloc::template rebind< _Slist_node< _Tp > >::other _Node_alloc`

Private Member Functions

- `_Slist_node_base * _M_erase_after (_Slist_node_base *__pos)`
- `_Slist_node_base * _M_erase_after (_Slist_node_base *, _Slist_node_base *)`
- `_Slist_node< _Tp > * _M_get_node ()`
- `void _M_put_node (_Slist_node< _Tp > *__p)`

Private Attributes

- `_Slist_node_base _M_head`

5.70.1 Detailed Description

`template<class _Tp, class _Alloc = allocator<_Tp>> class __gnu_cxx::slist< _Tp, _Alloc >`

This is an SGI extension.

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 291 of file `slist`.

The documentation for this class was generated from the following file:

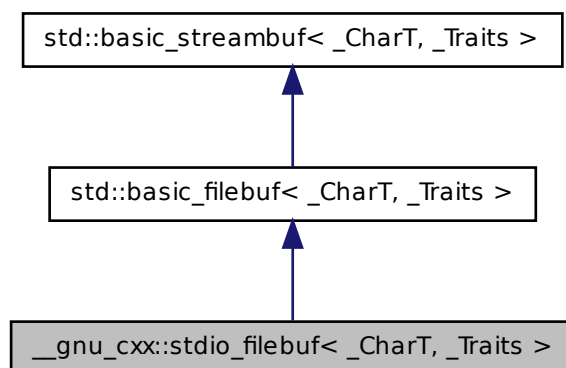
- [slist](#)

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C `FILE*`'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:



Public Types

- typedef `codecvt< char_type, char, __state_type >` **__codecvt_type**
- typedef `__basic_file< char >` **__file_type**
- typedef `basic_filebuf< char_type, traits_type >` **__filebuf_type**
- typedef `traits_type::state_type` **__state_type**
- typedef `basic_streambuf< char_type, traits_type >` **__streambuf_type**
- typedef `_CharT` **char_type**
- typedef `traits_type::int_type` **int_type**
- typedef `traits_type::off_type` **off_type**
- typedef `traits_type::pos_type` **pos_type**
- typedef `std::size_t` **size_t**
- typedef `_Traits` **traits_type**

Public Member Functions

- `stdio_filebuf()`
- `stdio_filebuf(int __fd, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`
- `stdio_filebuf(std::_c_file * __f, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`

- virtual `~stdio_filebuf()`
- `__filebuf_type * close()`
- `int fd()`
- `std::__c_file * file()`
- `streamsize in_avail()`
- `bool is_open() const` throw `()`
- `__filebuf_type * open(const char * __s, ios_base::openmode __mode)`
- `__filebuf_type * open(const std::string & __s, ios_base::openmode __mode)`
- `int_type sbumpc()`
- `int_type sgetc()`
- `streamsize sgetn(char_type * __s, streamsize __n)`
- `int_type snextc()`
- `int_type sputbackc(char_type __c)`
- `int_type sputc(char_type __c)`
- `streamsize sputn(const char_type * __s, streamsize __n)`
- `void stoss()`
- `int_type sungetc()`

Protected Member Functions

- `void _M_allocate_internal_buffer()`
- `bool _M_convert_to_external(char_type *, streamsize)`
- `void _M_create_pback()`
- `void _M_destroy_internal_buffer() throw()`
- `void _M_destroy_pback() throw()`
- `int _M_get_ext_pos(__state_type & __state)`
- `pos_type _M_seek(off_type __off, ios_base::seekdir __way, __state_type __state)`
- `void _M_set_buffer(streamsize __off)`
- `bool _M_terminate_output()`
- `void gbump(int __n)`
- virtual `void imbue(const locale & __loc)`
- virtual `int_type overflow(int_type __c=_Traits::eof())`
- virtual `int_type overflow(int_type=__traits_type::eof())`
- virtual `int_type pbackfail(int_type=__traits_type::eof())`
- virtual `int_type pbackfail(int_type __c=_Traits::eof())`
- `void pbump(int __n)`
- virtual `pos_type seekoff(off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- virtual `pos_type seekoff(off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)`
- virtual `pos_type seekpos(pos_type, ios_base::openmode=ios_base::in|ios_base::out)`

- virtual [pos_type](#) **seekpos** ([pos_type](#) __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual [__streambuf_type](#) * **setbuf** ([char_type](#) *__s, streamsize __n)
- virtual basic_streambuf< [char_type](#), _Traits > * **setbuf** ([char_type](#) *, streamsize)
- void **setg** ([char_type](#) *__gbeg, [char_type](#) *__gnext, [char_type](#) *__gend)
- void **setp** ([char_type](#) *__pbeg, [char_type](#) *__pend)
- virtual streamsize **showmanyc** ()
- virtual int **sync** ()
- virtual [int_type](#) **uflow** ()
- virtual [int_type](#) **underflow** ()
- virtual streamsize **xsgetn** ([char_type](#) *__s, streamsize __n)
- virtual streamsize **xsgetn** ([char_type](#) *__s, streamsize __n)
- virtual streamsize **xspn** (const [char_type](#) *__s, streamsize __n)
- virtual streamsize **xspn** (const [char_type](#) *__s, streamsize __n)

- [char_type](#) * **eback** () const
- [char_type](#) * **gptr** () const
- [char_type](#) * **egptr** () const

- [char_type](#) * **pbase** () const
- [char_type](#) * **pptr** () const
- [char_type](#) * **epptr** () const

Protected Attributes

- [char_type](#) * **_M_buf**
- bool **_M_buf_allocated**
- size_t **_M_buf_size**
- const [__codecvt_type](#) * **_M_codecvt**
- char * **_M_ext_buf**
- streamsize **_M_ext_buf_size**
- char * **_M_ext_end**
- const char * **_M_ext_next**
- [__file_type](#) **_M_file**
- [__c_lock](#) **_M_lock**
- ios_base::openmode **_M_mode**
- bool **_M_reading**
- [__state_type](#) **_M_state_beg**
- [__state_type](#) **_M_state_cur**
- [__state_type](#) **_M_state_last**
- bool **_M_writing**

- [char_type](#) **_M_pback**
- [char_type](#) * **_M_pback_cur_save**
- [char_type](#) * **_M_pback_end_save**
- bool **_M_pback_init**

Friends

- `template<bool _IsMove, typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`
`_CharT2 >, _CharT2 *)`
 - `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`
`bool &)`
 - `class basic_ios< char_type, traits_type >`
 - `class basic_istream< char_type, traits_type >`
 - `class basic_ostream< char_type, traits_type >`
 - `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf-`
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >,`
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _-`
`Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
 - `class ios_base`
 - `class istreambuf_iterator< char_type, traits_type >`
 - `template<typename _CharT2, typename _Traits2 >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, _CharT2 *)`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
 - `class ostreambuf_iterator< char_type, traits_type >`
-
- `locale pubimbue (const locale &__loc)`
 - `locale getloc () const`
 - `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
 - `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_-`
`base::openmode __mode=ios_base::in|ios_base::out)`
 - `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_-`
`base::in|ios_base::out)`
 - `int pubsync ()`
 - `char_type * _M_in_beg`
 - `char_type * _M_in_cur`
 - `char_type * _M_in_end`
 - `char_type * _M_out_beg`
 - `char_type * _M_out_cur`
 - `char_type * _M_out_end`
 - `locale _M_buf_locale`

5.71.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>  
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 49 of file `stdio_filebuf.h`.

5.71.2 Member Typedef Documentation

5.71.2.1 `template<typename _CharT, typename _Traits> typedef`
`basic_streambuf<char_type, traits_type> std::basic_filebuf< _CharT,`
`_Traits >::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 77 of file `fstream`.

5.71.2.2 `template<typename _CharT , typename _Traits = std::char_traits<_`
`CharT>> typedef _CharT __gnu_cxx::stdio_filebuf< _CharT, _Traits`
`>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf< _CharT, _Traits >](#).

Definition at line 53 of file `stdio_filebuf.h`.

5.71.2.3 `template<typename _CharT , typename _Traits =`
`std::char_traits<_CharT>> typedef traits_type::int_type`
`__gnu_cxx::stdio_filebuf< _CharT, _Traits >::__int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf< _CharT, _Traits >](#).

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 957

Definition at line 55 of file `stdio_filebuf.h`.

5.71.2.4 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> typedef traits_type::off_type
__gnu_cxx::stdio_filebuf<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf<_CharT, _Traits>](#).

Definition at line 57 of file `stdio_filebuf.h`.

5.71.2.5 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> typedef traits_type::pos_type
__gnu_cxx::stdio_filebuf<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf<_CharT, _Traits>](#).

Definition at line 56 of file `stdio_filebuf.h`.

5.71.2.6 `template<typename _CharT, typename _Traits = std::char_traits<_
_CharT>> typedef _Traits __gnu_cxx::stdio_filebuf<_CharT, _Traits
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_filebuf<_CharT, _Traits>](#).

Definition at line 54 of file `stdio_filebuf.h`.

5.71.3 Constructor & Destructor Documentation

5.71.3.1 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> __gnu_cxx::stdio_filebuf<_CharT,
_Traits>::stdio_filebuf () [inline]`

deferred initialization

Definition at line 64 of file `stdio_filebuf.h`.

```
5.71.3.2  template<typename _CharT, typename _Traits >
          __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf
          ( int __fd, std::ios_base::openmode __mode, size_t __size =
            static_cast<size_t>(BUFSIZ) )
```

Parameters

fd An open file descriptor.

mode Same meaning as in a standard filebuf.

size Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 126 of file `stdio_filebuf.h`.

References `std::basic_filebuf< _CharT, _Traits >::M_buf_size`, `std::basic_filebuf< _CharT, _Traits >::M_mode`, `std::basic_filebuf< _CharT, _Traits >::M_reading`, `std::basic_filebuf< _CharT, _Traits >::M_set_buffer()`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

```
5.71.3.3  template<typename _CharT, typename _Traits >
          __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
          std::_c_file * __f, std::ios_base::openmode __mode, size_t __size =
            static_cast<size_t>(BUFSIZ) )
```

Parameters

f An open `FILE*`.

mode Same meaning as in a standard filebuf.

size Optimal or preferred size of internal buffer, in chars. Defaults to system's `BUFSIZ`.

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 142 of file `stdio_filebuf.h`.

References `std::basic_filebuf< _CharT, _Traits >::M_buf_size`, `std::basic_filebuf< _CharT, _Traits >::M_mode`, `std::basic_filebuf< _CharT, _Traits >::M_reading`, `std::basic_filebuf< _CharT, _Traits >::M_set_buffer()`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 959

5.71.3.4 `template<typename _CharT, typename _Traits>`
`__gnu_cxx::stdio_filebuf<_CharT, _Traits>::~~stdio_filebuf ()`
`[virtual]`

Closes the external data stream if the file descriptor constructor was used.

Definition at line 121 of file `stdio_filebuf.h`.

5.71.4 Member Function Documentation

5.71.4.1 `template<typename _CharT, typename _Traits> void`
`std::basic_filebuf<_CharT, _Traits>::_M_create_pback ()`
`[inline, protected, inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file `fstream`.

5.71.4.2 `template<typename _CharT, typename _Traits> void`
`std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback () throw`
`() [inline, protected, inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.4.3 `template<typename _CharT, typename _Traits> void`
`std::basic_filebuf<_CharT, _Traits>::_M_set_buffer (streamsize`
`__off) [inline, protected, inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 390 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::open()`, `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.4.4 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::_filebuf_type * std::basic_filebuf<_CharT, _Traits>::close () [inherited]`

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::is_open()`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::open()`.

5.71.4.5 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::eback () const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.4.6 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf<_CharT, _Traits>::egptr () const [inline, protected, inherited]`

Access to the get area.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 961

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.4.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::epptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 513 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.71.4.8 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> int __gnu_cxx::stdio_filebuf<_CharT,
_Traits>::fd () [inline]`

Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 107 of file `stdio_filebuf.h`.

5.71.4.9 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> std::_c_file* __gnu_cxx::stdio_filebuf<
_CharT, _Traits >::file () [inline]`

Returns

The underlying FILE*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 117 of file `stdio_filebuf.h`.

5.71.4.10 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits >::gbump (int __n)
[inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file `streambuf`.

5.71.4.11 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits >::getloc () const
[inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file `streambuf`.

5.71.4.12 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits >::gptr () const
[inline, protected, inherited]`

Access to the get area.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 963

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 463 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::imbue()`, `std::basic_filebuf<_CharT, _Traits>::showmanyc()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`, and `std::basic_filebuf<_CharT, _Traits>::underflow()`.

5.71.4.13 `template<typename _CharT, typename _Traits> void
std::basic_filebuf<_CharT, _Traits>::imbue (const locale &)
[protected, virtual, inherited]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 912 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

5.71.4.14 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::in_avail () [inline,
inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 262 of file streambuf.

5.71.4.15 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf< _CharT, _Traits >::is_open () const throw ()
[inline, inherited]`

Returns true if the external file is open.

Definition at line 222 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::close()`, `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::setbuf()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, and `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`.

5.71.4.16 `template<typename _CharT , typename _Traits > basic_filebuf<
_CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT,
_Traits >::open (const char * __s, ios_base::openmode __mode)
[inherited]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *s* using the flags given in *mode*.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 965

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +-----

```

-----+ | ios_base Flag combination stdio equivalent | |binary in out trunc app
| +-----+ | + w | | + + a | | + a | | + + w | | +
r | | + + r+ | | + + + w+ | | + + + a+ | | + + a+ | +-----
-----+ | + + wb | | + + + ab | | + + ab | | + + + wb | | + + rb | | + + + r+b | | + + + +
w+b | | + + + + a+b | | + + + a+b | +-----+

```

Definition at line 94 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::M_mode`, `std::basic_filebuf< _CharT, _Traits >::M_reading`, `std::basic_filebuf< _CharT, _Traits >::M_set_buffer()`, `std::ios_base::ate`, `std::basic_filebuf< _CharT, _Traits >::close()`, `std::ios_base::end`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

```
5.71.4.17 template<typename _CharT, typename _Traits> __filebuf_type*
std::basic_filebuf< _CharT, _Traits >::open ( const std::string &
__s, ios_base::openmode __mode ) [inline, inherited]
```

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Returns

this on success, NULL on failure

Definition at line 275 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::open()`.

```
5.71.4.18 template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::overflow ( int_type =
traits_type::eof() ) [inline, protected, virtual,
inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

c An additional character to consume.

Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 746 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.71.4.19 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail(int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

`eof()` on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 702 of file `streambuf`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 967

5.71.4.20 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::pbase () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`.

5.71.4.21 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::pbump (int __n)
[inline, protected, inherited]`

Moving the write position.

Parameters

- *n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.71.4.22 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::pptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence

- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.71.4.23 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file streambuf.

5.71.4.24 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file streambuf.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 969

5.71.4.25 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf<_CharT, _Traits>::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

5.71.4.26 `template<typename _CharT, typename _Traits> __streambuf_type*
std::basic_streambuf<_CharT, _Traits>::pubsetbuf (char_type *
__s, streamsize __n) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.71.4.27 `template<typename _CharT, typename _Traits> int
std::basic_streambuf<_CharT, _Traits>::pubsync () [inline,
inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

5.71.4.28 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sputc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

5.71.4.29 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf< _CharT, _Traits >::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

5.71.4.30 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf< _CharT, _Traits >::seekpos (pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 971

Definition at line 592 of file `streambuf`.

```
5.71.4.31 template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type, _Traits>* std::basic_streambuf<
_CharT, _Traits>::setbuf( char_type*, streamsize ) [inline,
protected, virtual, inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

```
5.71.4.32 template<typename _CharT, typename _Traits> basic_filebuf<
_CharT, _Traits>::__streambuf_type * std::basic_filebuf<
_CharT, _Traits>::setbuf( char_type* __s, streamsize __n )
[protected, virtual, inherited]
```

Manipulates the buffer.

Parameters

- s* Pointer to a buffer area.
- n* Size of *s*.

Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 686 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::__M_buf`, `std::basic_filebuf<_CharT, _Traits>::__M_buf_size`, and `std::basic_filebuf<_CharT, _Traits>::is_open()`.

5.71.4.33 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setg (char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline,
protected, inherited]`

Setting the three read area pointers.

Parameters

gbeg A pointer.
gnext A pointer.
gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

5.71.4.34 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setp (char_type *
__pbeg, char_type * __pend) [inline, protected,
inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.
pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file streambuf.

5.71.4.35 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sgetc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 973

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.71.4.36 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sgetn (char_type * __s,
streamsize __n) [inline, inherited]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

5.71.4.37 `template<typename _CharT, typename _Traits> streamsize
std::basic_filebuf<_CharT, _Traits>::showmanyc ()
[protected, virtual, inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
[27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 178 of file fstream.tcc.

References [std::basic_filebuf<_CharT, _Traits>::_M_mode](#), [std::ios_base::binary](#), [std::basic_streambuf<_CharT, _Traits>::egptr\(\)](#), [std::basic_streambuf<_CharT, _Traits>::gptr\(\)](#), [std::ios_base::in](#), and [std::basic_filebuf<_CharT, _Traits>::is_open\(\)](#).

5.71.4.38 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::snextc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls [sbumpc\(\)](#), and if that function returns `traits::eof()`, so does this function. Otherwise, [sgetc\(\)](#).

Definition at line 276 of file streambuf.

Referenced by [std::basic_istream<_CharT, _Traits>::get\(\)](#), [std::basic_istream<_CharT, _Traits>::getline\(\)](#), [std::basic_istream<_CharT, _Traits>::ignore\(\)](#), and [std::basic_istream<_CharT, _Traits>::sentry::sentry\(\)](#).

5.71.4.39 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputbackc (char_type
__c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

c The character to push back.

Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file streambuf.

Referenced by [std::basic_istream<_CharT, _Traits>::putback\(\)](#).

5.71.4.40 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputc (char_type __c)
[inline, inherited]`

Entry point for all single-character output functions.

Parameters

c A character to output.

Returns

c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

5.71.4.41 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sputn (const char_type *
__s, streamsize __n) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.71.4.42 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::stossc () [inline,
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file streambuf.

5.71.4.43 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sungetc () [inline,
inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns [pbackfail\(\)](#). The effect is to *unget* the last character *gotten*.

Definition at line 375 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

5.71.4.44 `template<typename _CharT, typename _Traits> int
std::basic_filebuf<_CharT, _Traits>::sync () [protected,
virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 895 of file fstream.tcc.

References `std::basic_streambuf<_CharT, _Traits>::pbase()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 977

5.71.4.45 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::uflow() [inline,
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 678 of file `streambuf`.

5.71.4.46 `template<typename _CharT, typename _Traits> basic_filebuf<
_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits
>::underflow() [protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 204 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_buf_size`, `std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback()`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_next`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in()`, `std::ios_base::in`, and `std::min()`.

5.71.4.47 `template<typename _CharT , typename _Traits > streamsize
std::basic_streambuf< _CharT, _Traits >::xsgetn (char_type * __s,
streamsize __n) [protected, virtual, inherited]`

Multiple character extraction.

Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills *s*[0] through *s*[*n*-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

5.71.4.48 `template<typename _CharT , typename _Traits > streamsize
std::basic_streambuf< _CharT, _Traits >::xsputn (const
char_type * __s, streamsize __n) [protected, virtual,
inherited]`

Multiple character insertion.

Parameters

- s* A buffer area.

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 979

n Maximum number of characters to write.

Returns

The number of characters written.

Writes `s[0]` through `s[n-1]` to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.71.5 Member Data Documentation

5.71.5.1 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf<_CharT, _Traits>::_M_buf` `[protected,
inherited]`

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::setbuf()`.

5.71.5.2 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits>::_M_buf_locale` `[protected,
inherited]`

Current locale setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

5.71.5.3 `template<typename _CharT, typename _Traits> size_t
std::basic_filebuf<_CharT, _Traits>::_M_buf_size` `[protected,
inherited]`

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 116 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::setbuf()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.5.4 `template<typename _CharT, typename _Traits> char*
std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected,
inherited]`

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to [eback\(\)](#).

Definition at line 151 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.5.5 `template<typename _CharT, typename _Traits> streamsize
std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size
[protected, inherited]`

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.5.6 `template<typename _CharT, typename _Traits> const char*
std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected,
inherited]`

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to [egptr\(\)](#).

Definition at line 163 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.5.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_beg
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

5.71 __gnu_cxx::stdio_filebuf<_CharT, _Traits> Class Template Reference 981

- get == input == read
- put == output == write

Definition at line 180 of file streambuf.

5.71.5.8 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file streambuf.

5.71.5.9 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file streambuf.

5.71.5.10 `template<typename _CharT, typename _Traits> ios_base::openmode
std::basic_filebuf<_CharT, _Traits>::_M_mode [protected,
inherited]`

Place to stash in || out || in | out settings for current filebuf.

Definition at line 94 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.71.5.11 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_beg
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file streambuf.

5.71.5.12 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file streambuf.

5.71.5.13 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

5.71 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference 983

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

5.71.5.14 `template<typename _CharT, typename _Traits> char_type
std::basic_filebuf<_CharT, _Traits>::_M_pback [protected,
inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 137 of file `fstream`.

5.71.5.15 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf<_CharT, _Traits>::_M_pback_cur_save
[protected, inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 138 of file `fstream`.

5.71.5.16 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf<_CharT, _Traits>::_M_pback_end_save
[protected, inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 139 of file `fstream`.

5.71.5.17 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf< _CharT, _Traits >::_M_pback_init
[protected, inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 140 of file fstream.

5.71.5.18 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf< _CharT, _Traits >::_M_reading [protected,
inherited]`

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

The documentation for this class was generated from the following file:

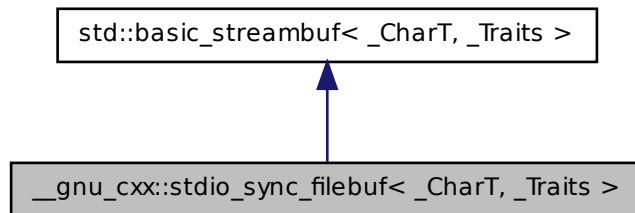
- [stdio_filebuf.h](#)

5.72 `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >` Class Template Reference

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE*’s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `traits_type::int_type` [int_type](#)
- typedef `traits_type::off_type` [off_type](#)
- typedef `traits_type::pos_type` [pos_type](#)
- typedef `_Traits` [traits_type](#)
- typedef `basic_streambuf<char_type, traits_type>` [__streambuf_type](#)

Public Member Functions

- **`stdio_sync_filebuf`** (`std::__c_file *__f`)
- `std::__c_file *const` [file](#) ()
- `streamsize` [in_avail](#) ()
- [int_type](#) [sbumpc](#) ()
- [int_type](#) [sgetc](#) ()
- `streamsize` [sgetn](#) ([char_type](#) *__s, `streamsize` __n)
- [int_type](#) [snextc](#) ()
- [int_type](#) [sputbackc](#) ([char_type](#) __c)
- [int_type](#) [sputc](#) ([char_type](#) __c)
- `streamsize` [sputn](#) (const [char_type](#) *__s, `streamsize` __n)
- void [stoss](#) ()
- [int_type](#) [sungetc](#) ()

Protected Member Functions

- void [gbump](#) (int __n)
- virtual void [imbue](#) (const locale &)
- virtual [int_type overflow](#) (int_type __c=traits_type::eof())
- virtual [int_type pbackfail](#) (int_type __c=traits_type::eof())
- void [pbump](#) (int __n)
- virtual [std::streampos seekoff](#) (std::streamoff __off, std::ios_base::seekdir __dir, std::ios_base::openmode=std::ios_base::in|std::ios_base::out)
- virtual [pos_type seekoff](#) (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual [std::streampos seekpos](#) (std::streampos __pos, std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out)
- virtual [pos_type seekpos](#) (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf< [char_type](#), _Traits > * [setbuf](#) ([char_type](#) *, streamsize)
- void [setg](#) ([char_type](#) * __gbeg, [char_type](#) * __gnext, [char_type](#) * __gend)
- void [setp](#) ([char_type](#) * __pbeg, [char_type](#) * __pend)
- virtual streamsize [showmanyc](#) ()
- virtual int [sync](#) ()
- template<>
[stdio_sync_filebuf](#)< wchar_t >::int_type [syncgetc](#) ()
- template<>
[stdio_sync_filebuf](#)< char >::int_type [syncgetc](#) ()
- [int_type syncgetc](#) ()
- template<>
[stdio_sync_filebuf](#)< wchar_t >::int_type [syncputc](#) (int_type __c)
- template<>
[stdio_sync_filebuf](#)< char >::int_type [syncputc](#) (int_type __c)
- [int_type syncputc](#) (int_type __c)
- template<>
[stdio_sync_filebuf](#)< wchar_t >::int_type [syncungetc](#) (int_type __c)
- [int_type syncungetc](#) (int_type __c)
- template<>
[stdio_sync_filebuf](#)< char >::int_type [syncungetc](#) (int_type __c)
- virtual [int_type uflow](#) ()
- virtual [int_type underflow](#) ()
- template<>
[std::streamsize xsgetn](#) (char *__s, std::streamsize __n)
- virtual [std::streamsize xsgetn](#) ([char_type](#) *__s, std::streamsize __n)
- template<>
[std::streamsize xsgetn](#) (wchar_t *__s, std::streamsize __n)
- template<>
[std::streamsize xsputn](#) (const char *__s, std::streamsize __n)

- `template<>`
`std::streamsize xspn (const wchar_t * __s, std::streamsize __n)`
- `virtual std::streamsize xspn (const char_type * __s, std::streamsize __n)`
- `char_type * eback () const`
- `char_type * gptr () const`
- `char_type * egptr () const`
- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * ep_ptr () const`

Friends

- `template<bool _IsMove, typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`
`_CharT2 >, _CharT2 *)`
- `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`
`bool &)`
- `class basic_ios< char_type, traits_type >`
- `class basic_istream< char_type, traits_type >`
- `class basic_ostream< char_type, traits_type >`
- `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_-`
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >,`
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _-`
`Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
- `class istreambuf_iterator< char_type, traits_type >`
- `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
- `template<typename _CharT2, typename _Traits2 >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, _CharT2 *)`
- `class ostreambuf_iterator< char_type, traits_type >`
- `locale pubimbue (const locale & __loc)`
- `locale getloc () const`
- `__streambuf_type * pubsetbuf (char_type * __s, streamsize __n)`

- [pos_type pubseekoff](#) ([off_type __off](#), [ios_base::seekdir __way](#), [ios_base::openmode __mode=ios_base::in|ios_base::out](#))
- [pos_type pubseekpos](#) ([pos_type __sp](#), [ios_base::openmode __mode=ios_base::in|ios_base::out](#))
- [int pubsync](#) ()
- [char_type * _M_in_beg](#)
- [char_type * _M_in_cur](#)
- [char_type * _M_in_end](#)
- [char_type * _M_out_beg](#)
- [char_type * _M_out_cur](#)
- [char_type * _M_out_end](#)
- [locale _M_buf_locale](#)

5.72.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE*’s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 55 of file `stdio_sync_filebuf.h`.

5.72.2 Member Typedef Documentation

5.72.2.1 `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_streambuf<_CharT, _Traits>::__streambuf_type [inherited]`

This is a non-standard type.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [std::basic_stringbuf< _CharT, _Traits, _Alloc >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 133 of file `streambuf`.

5.72.2.2 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _CharT __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 59 of file `stdio_sync_filebuf.h`.

5.72.2.3 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 61 of file `stdio_sync_filebuf.h`.

5.72.2.4 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::off_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 63 of file `stdio_sync_filebuf.h`.

5.72.2.5 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef traits_type::pos_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 62 of file `stdio_sync_filebuf.h`.

5.72.2.6 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> typedef _Traits __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 60 of file `stdio_sync_filebuf.h`.

5.72.3 Member Function Documentation

5.72.3.1 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::eback () const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 460 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.72.3.2 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::egptr () const [inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.72.3.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::epptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.72.3.4 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> std::__c_file* const
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::file ()
[inline]`

Returns

The underlying FILE*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 87 of file `stdio_sync_filebuf.h`.

5.72.3.5 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::gbump (int __n)
[inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

5.72.3.6 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::getloc () const
[inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

5.72.3.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::gptr () const [inline,
protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.72.3.8 `template<typename _CharT, typename _Traits> virtual void
std::basic_streambuf< _CharT, _Traits >::imbue (const locale &)
[inline, protected, virtual, inherited]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in [std::basic_filebuf<_CharT, _Traits>](#), [std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>](#), and [std::basic_filebuf<char_type, traits_type>](#).

Definition at line 554 of file `streambuf`.

5.72.3.9 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::in_avail () [inline,
inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived [showmanyc\(\)](#).

Definition at line 262 of file `streambuf`.

5.72.3.10 `template<typename _CharT, typename _Traits
= std::char_traits<_CharT>> virtual int_type
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::overflow (
int_type = traits_type::eof()) [inline, protected,
virtual]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

c An additional character to consume.

Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 140 of file `stdio_sync_filebuf.h`.

```
5.72.3.11  template<typename _CharT , typename _Traits
             = std::char_traits<_CharT>> virtual int_type
             __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::pbackfail (
             int_type = traits_type::eof() ) [inline, protected,
             virtual]
```

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

`eof()` on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 115 of file `stdio_sync_filebuf.h`.

5.72.3.12 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::pbase () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 507 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`.

5.72.3.13 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::pbump (int __n)
[inline, protected, inherited]`

Moving the write position.

Parameters

- *n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file `streambuf`.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.72.3.14 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::pptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence

- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.72.3.15 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file streambuf.

5.72.3.16 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file streambuf.

5.72.3.17 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf<_CharT, _Traits>::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

5.72.3.18 `template<typename _CharT, typename _Traits> __streambuf_type*
std::basic_streambuf<_CharT, _Traits>::pubsetbuf (char_type *
__s, streamsize __n) [inline, inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.72.3.19 `template<typename _CharT, typename _Traits> int
std::basic_streambuf<_CharT, _Traits>::pubsync () [inline,
inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::sync()`.

5.72.3.20 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

5.72.3.21 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf< _CharT, _Traits >::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

5.72.3.22 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf< _CharT, _Traits >::seekpos (pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

5.72.3.23 `template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type, _Traits>* std::basic_streambuf<
_CharT, _Traits>::setbuf(char_type*, streamsize) [inline,
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

5.72.3.24 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::setg(char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline,
protected, inherited]`

Setting the three read area pointers.

Parameters

gbeg A pointer.
gnext A pointer.
gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file `streambuf`.

5.72.3.25 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::setp(char_type *
__pbeg, char_type * __pend) [inline, protected,
inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.

pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file streambuf.

5.72.3.26 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sgetc() [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.72.3.27 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sgetn(char_type * __s,
streamsize __n) [inline, inherited]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file streambuf.

5.72.3.28 `template<typename _CharT, typename _Traits> virtual streamsize
std::basic_streambuf<_CharT, _Traits>::showmanyc ()
[inline, protected, virtual, inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 627 of file `streambuf`.

5.72.3.29 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::snextc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.72.3.30 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sputbackc (char_type
__c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

c The character to push back.

Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

5.72.3.31 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sputc (char_type __c)
[inline, inherited]`

Entry point for all single-character output functions.

Parameters

c A character to output.

Returns

c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

5.72.3.32 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sputn (const char_type *
__s, streamsize __n) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xsputn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.72.3.33 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::stossc () [inline,
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

5.72.3.34 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sungetc () [inline,
inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `ebackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::unget()`.

5.72.3.35 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::sync(void) [inline, protected, virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 159 of file `stdio_sync_filebuf.h`.

5.72.3.36 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::uflow() [inline, protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 107 of file `stdio_sync_filebuf.h`.

5.72.3.37 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::underflow() [inline, protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 100 of file `stdio_sync_filebuf.h`.

```
5.72.3.38  template<typename _CharT , typename _Traits =
            std::char_traits<_CharT>> virtual std::streamsize
            __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::xsgetn (
            char_type * __s, std::streamsize __n ) [protected, virtual]
```

Multiple character extraction.

Parameters

s A buffer area.

n Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by [sbumpc\(\)](#). Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

5.72.3.39 `template<typename _CharT, typename _Traits =
std::char_traits<_CharT>> virtual std::streamsize
__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::xspn(const
char_type * __s, std::streamsize __n) [protected, virtual]`

Multiple character insertion.

Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

5.72.4 Member Data Documentation

5.72.4.1 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits>::_M_buf_locale
[protected, inherited]`

Current locale setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf<_CharT, _Traits>::basic_filebuf()`.

5.72.4.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_beg
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

5.72.4.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

5.72.4.4 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_in_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.72.4.5 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_out_beg
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

5.72.4.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_cur
[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

5.72.4.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

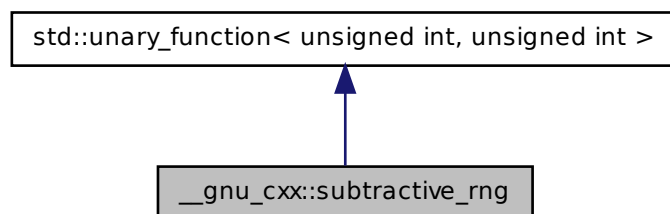
Definition at line 185 of file `streambuf`.

The documentation for this class was generated from the following file:

- [stdio_sync_filebuf.h](#)

5.73 `__gnu_cxx::subtractive_rng` Class Reference

Inheritance diagram for `__gnu_cxx::subtractive_rng`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- [subtractive_rng](#) (unsigned int `__seed`)
- [subtractive_rng](#) ()
- void `_M_initialize` (unsigned int `__seed`)
- unsigned int [operator\(\)](#) (unsigned int `__limit`)

5.73.1 Detailed Description

The [subtractive_rng](#) class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits.

Definition at line 347 of file `ext/functional`.

5.73.2 Member Typedef Documentation

5.73.2.1 `template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.73.2.2 `template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

5.73.3 Constructor & Destructor Documentation

5.73.3.1 `__gnu_cxx::subtractive_rng::subtractive_rng (unsigned int __seed)
[inline]`

Ctor allowing you to initialize the seed.

Definition at line 389 of file `ext/functional`.

5.73.3.2 `__gnu_cxx::subtractive_rng::subtractive_rng () [inline]`

Default ctor; initializes its state with some number you don't see.

Definition at line 393 of file `ext/functional`.

5.73.4 Member Function Documentation

5.73.4.1 `unsigned int __gnu_cxx::subtractive_rng::operator() (unsigned int
__limit) [inline]`

Returns a number less than the argument.

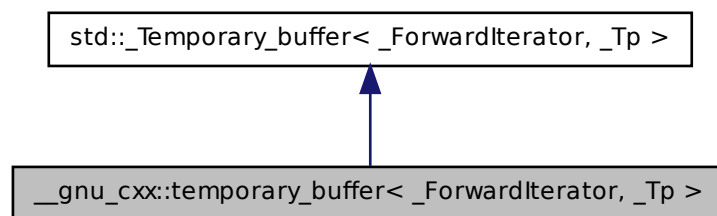
Definition at line 358 of file `ext/functional`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

5.74 `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>`:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [temporary_buffer](#) (_ForwardIterator __first, _ForwardIterator __last)
- [~temporary_buffer](#) ()
- iterator [begin](#) ()
- iterator [end](#) ()
- size_type [requested_size](#) () const
- size_type [size](#) () const

Protected Attributes

- pointer **_M_buffer**
- size_type **_M_len**
- size_type **_M_original_len**

5.74.1 Detailed Description

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_
ForwardIterator>::value_type> struct __gnu_cxx::temporary_buffer< _
ForwardIterator, _Tp >
```

This class provides similar behavior and semantics of the standard functions [get_temporary_buffer\(\)](#) and [return_temporary_buffer\(\)](#), but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the [begin\(\)](#), [end\(\)](#), [size\(\)](#) functions, as well as [requested_size\(\)](#). For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like [get_temporary_buffer\(\)](#), not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if [size\(\)](#) is less than [requested_size\(\)](#), then this didn't happen.

Definition at line 182 of file `ext/memory`.

5.74.2 Constructor & Destructor Documentation

```
5.74.2.1 template<class _ForwardIterator , class _Tp = typename
std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp
>::temporary_buffer ( _ForwardIterator __first, _ForwardIterator
__last ) [inline]
```

Requests storage large enough to hold a copy of `[first,last)`.

Definition at line 185 of file `ext/memory`.

```
5.74.2.2 template<class _ForwardIterator , class _Tp = typename
std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp
>::~~temporary_buffer ( ) [inline]
```

Destroys objects and frees storage.

Definition at line 189 of file `ext/memory`.

5.74.3 Member Function Documentation

5.74.3.1 `template<typename _ForwardIterator, typename _Tp> iterator
std::_Temporary_buffer<_ForwardIterator, _Tp>::begin ()
[inline, inherited]`

As per Table mumble.

Definition at line 150 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

5.74.3.2 `template<typename _ForwardIterator, typename _Tp> iterator
std::_Temporary_buffer<_ForwardIterator, _Tp>::end ()
[inline, inherited]`

As per Table mumble.

Definition at line 155 of file `stl_tempbuf.h`.

5.74.3.3 `template<typename _ForwardIterator, typename _Tp> size_type
std::_Temporary_buffer<_ForwardIterator, _Tp>::requested_size () const
[inline, inherited]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 145 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

5.74.3.4 `template<typename _ForwardIterator, typename _Tp> size_type
std::_Temporary_buffer<_ForwardIterator, _Tp>::size () const
[inline, inherited]`

As per Table mumble.

Definition at line 140 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this struct was generated from the following file:

- [ext/memory](#)

5.75 `__gnu_cxx::throw_allocator_base< _Tp, _Cond >` Class Template Reference

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

Inheritance diagram for `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`:



Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `pointer` **address** (reference __x) const
- `const_pointer` **address** (const_reference __x) const
- `pointer` **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- `void` **check_allocated** (pointer __p, size_type __n)
- `void` **check_allocated** (size_type __n)
- `void` **check_allocated** (void *p, size_t size)
- `void` **check_allocated** (size_t label)
- `void` **construct** (pointer __p, const value_type &val)
- `template<typename... _Args>`
`void` **construct** (pointer __p, _Args &&...__args)
- `void` **deallocate** (pointer __p, size_type __n)
- `void` **destroy** (pointer __p)
- `void` **erase** (void *p, size_t size)
- `void` **insert** (void *p, size_t size)
- `size_type` **max_size** () const throw ()

5.76 `__gnu_cxx::throw_allocator_limit<_Tp>` Struct Template Reference 1015

Static Public Member Functions

- static `size_t get_label()`
- static `void set_label(size_t l)`

5.75.1 Detailed Description

`template<typename _Tp, typename _Cond> class __gnu_cxx::throw_allocator_base<_Tp, _Cond>`

Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.

Note: Deallocate not allowed to throw.

Definition at line 598 of file `throw_allocator.h`.

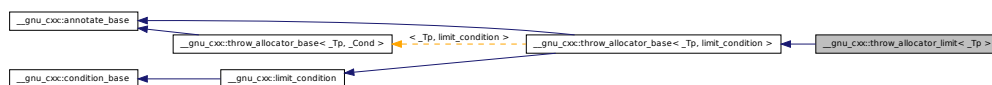
The documentation for this class was generated from the following file:

- [throw_allocator.h](#)

5.76 `__gnu_cxx::throw_allocator_limit<_Tp>` Struct Template Reference

Allocator throwing via limit condition.

Inheritance diagram for `__gnu_cxx::throw_allocator_limit<_Tp>`:



Public Types

- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- **throw_allocator_limit** (const [throw_allocator_limit](#) &) throw ()
- template<typename _Tp1 >
 throw_allocator_limit (const [throw_allocator_limit](#)< _Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- void **check_allocated** (size_t label)
- void **check_allocated** (size_type __n)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_allocated** (void *p, size_t size)
- void **construct** (pointer __p, const value_type &val)
- void **construct** (pointer __p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (pointer __p)
- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)
- size_type **max_size** () const throw ()

Static Public Member Functions

- static size_t & **count** ()
- static size_t **get_label** ()
- static size_t & **limit** ()
- static void **set_label** (size_t l)
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

5.76.1 Detailed Description

template<typename _Tp> struct __gnu_cxx::throw_allocator_limit< _Tp >

Allocator throwing via limit condition.

Definition at line 688 of file [throw_allocator.h](#).

The documentation for this struct was generated from the following file:

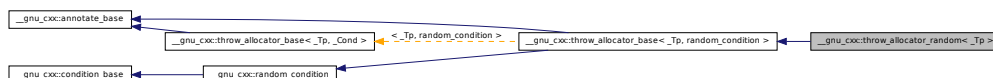
- [throw_allocator.h](#)

5.77 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference 1017

5.77 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

Allocator throwing via random condition.

Inheritance diagram for `__gnu_cxx::throw_allocator_random<_Tp>`:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_random** (const [throw_allocator_random](#) &) throw ()
- template<typename _Tp1 >
 throw_allocator_random (const [throw_allocator_random](#)< _Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- void **check_allocated** (size_t label)
- void **check_allocated** (void *p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_allocated** (size_type __n)
- void **construct** (pointer __p, const value_type &val)
- void **construct** (pointer __p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (pointer __p)
- void **erase** (void *p, size_t size)
- void **insert** (void *p, size_t size)

- size_type **max_size** () const throw ()
- void **seed** (unsigned long __s)

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)
- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

5.77.1 Detailed Description

template<typename _Tp> struct **__gnu_cxx::throw_allocator_random**< _Tp >

Allocator throwing via random condition.

Definition at line 707 of file throw_allocator.h.

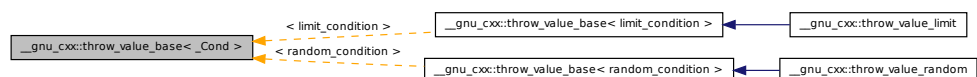
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.78 __gnu_cxx::throw_value_base< _Cond > Struct Template Reference

Class with exception generation control. Intended to be used as a value_type in templated code.

Inheritance diagram for **__gnu_cxx::throw_value_base**< _Cond >:



Public Types

- typedef _Cond **condition_type**

Public Member Functions

- `throw_value_base` (const [throw_value_base](#) &__v)
- `throw_value_base` (const std::size_t __i)
- `throw_value_base` & `operator++` ()
- `throw_value_base` & `operator=` (const [throw_value_base](#) &__v)

Public Attributes

- `std::size_t _M_i`

5.78.1 Detailed Description

`template<typename _Cond> struct __gnu_cxx::throw_value_base< _Cond >`

Class with exception generation control. Intended to be used as a `value_type` in templated code. Note: Destructor not allowed to throw.

Definition at line 453 of file `throw_allocator.h`.

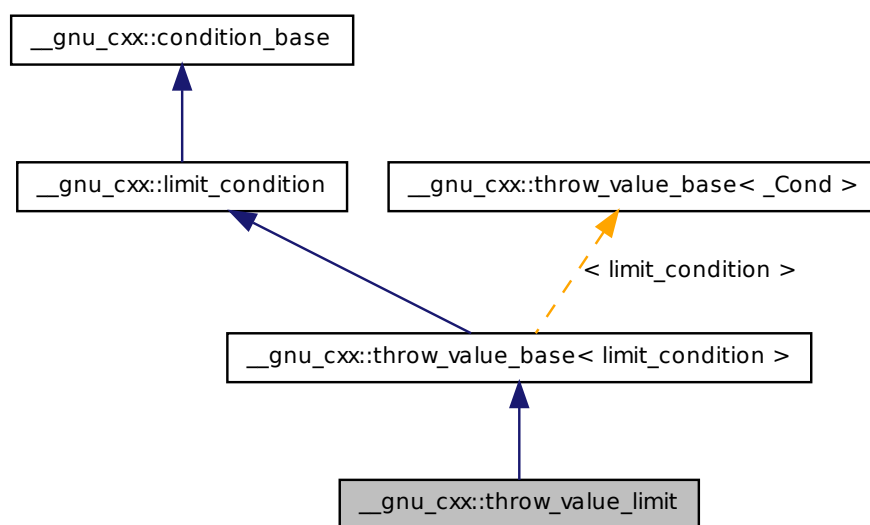
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.79 `__gnu_cxx::throw_value_limit` Struct Reference

Type throwing via limit condition.

Inheritance diagram for `__gnu_cxx::throw_value_limit`:



Public Types

- typedef `throw_value_base< limit_condition > base_type`
- typedef `limit_condition condition_type`

Public Member Functions

- `throw_value_limit` (const `throw_value_limit` &__other)
- `throw_value_limit` (const std::size_t __i)
- `throw_value_base` & `operator++` ()

Static Public Member Functions

- static size_t & `count` ()
- static size_t & `limit` ()
- static void `set_limit` (const size_t __l)
- static void `throw_conditionally` ()

Public Attributes

- `std::size_t _M_i`

5.79.1 Detailed Description

Type throwing via limit condition.

Definition at line 559 of file `throw_allocator.h`.

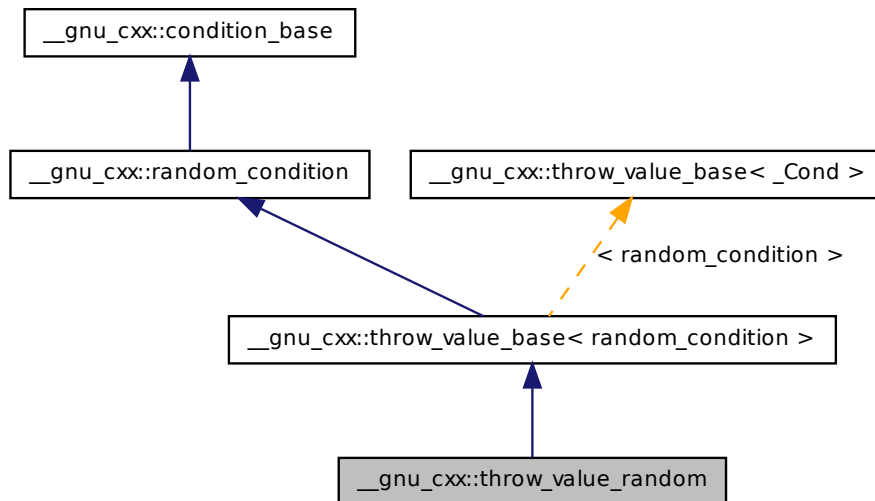
The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.80 `__gnu_cxx::throw_value_random` Struct Reference

Type throwing via random condition.

Inheritance diagram for `__gnu_cxx::throw_value_random`:



Public Types

- typedef [throw_value_base](#)< [random_condition](#) > **base_type**
- typedef [random_condition](#) **condition_type**

Public Member Functions

- **throw_value_random** (const [throw_value_random](#) &__other)
- **throw_value_random** (const std::size_t __i)
- [throw_value_base](#) & **operator++** ()
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

Public Attributes

- std::size_t **M_i**

5.80.1 Detailed Description

Type throwing via random condition.

Definition at line 574 of file [throw_allocator.h](#).

The documentation for this struct was generated from the following file:

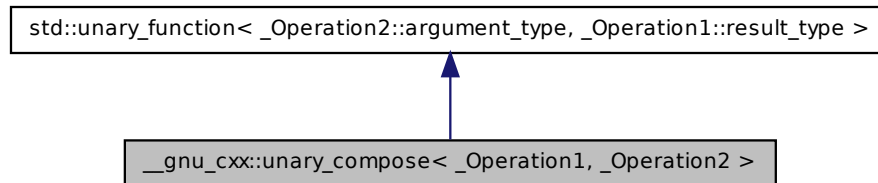
- [throw_allocator.h](#)

5.81 `__gnu_cxx::unary_compose< _Operation1, _Operation2 >` Class Template Reference

An [SGI extension](#) .

5.81 `__gnu_cxx::unary_compose<_Operation1, _Operation2>` Class Template Reference 1023

Inheritance diagram for `__gnu_cxx::unary_compose<_Operation1, _Operation2>`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- **unary_compose** (const `_Operation1` &__x, const `_Operation2` &__y)
- `_Operation1::result_type` **operator()** (const typename `_Operation2::argument_type` &__x) const

Protected Attributes

- `_Operation1` **_M_fn1**
- `_Operation2` **_M_fn2**

5.81.1 Detailed Description

```
template<class _Operation1, class _Operation2> class __gnu_cxx::unary_compose<_Operation1, _Operation2>
```

An [SGI extension](#) .

Definition at line 124 of file `ext/functional`.

5.81.2 Member Typedef Documentation

5.81.2.1 `template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.81.2.2 `template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [ext/functional](#)

5.82 `__gnu_debug::__is_same< _Type1, _Type2 >` Struct Template Reference

Static Public Attributes

- static const bool `value`

5.82.1 Detailed Description

`template<typename _Type1, typename _Type2> struct __gnu_debug::__is_ -
same< _Type1, _Type2 >`

Determine if the two types are the same.

Definition at line 45 of file `formatter.h`.

The documentation for this struct was generated from the following file:

- [formatter.h](#)

5.83 `__gnu_debug::_After_nth_from<_Iterator>` Class Template Reference

Public Member Functions

- `_After_nth_from` (const difference_type &__n, const _Iterator &__base)
- `bool operator()` (const _Iterator &__x) const

5.83.1 Detailed Description

`template<typename _Iterator> class __gnu_debug::_After_nth_from<_Iterator>`

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 63 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

5.84 `__gnu_debug::_BeforeBeginHelper<_Sequence>` Struct Template Reference

Public Types

- `typedef _Sequence::const_iterator _It`

Static Public Member Functions

- `static bool _M_Is` (_It __it, const _Sequence *__seq)

5.84.1 Detailed Description

`template<typename _Sequence> struct __gnu_debug::_BeforeBeginHelper<_Sequence>`

Helper struct to deal with sequence offering a `before_begin` iterator.

Definition at line 47 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe_iterator.h](#)

5.85 `__gnu_debug::_Not_equal_to< _Type >` Class Template Reference

Public Member Functions

- `_Not_equal_to` (const `_Type` &__v)
- `bool operator()` (const `_Type` &__x) const

5.85.1 Detailed Description

`template<typename _Type> class __gnu_debug::_Not_equal_to< _Type >`

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

Definition at line 48 of file `safe_sequence.h`.

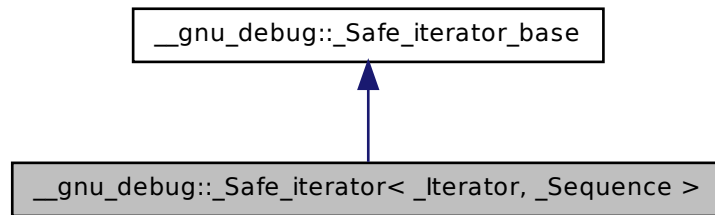
The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

5.86 `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >` Class Template Reference

Safe iterator wrapper.

Inheritance diagram for `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>`:



Public Types

- typedef `_Traits::difference_type` **difference_type**
- typedef `_Traits::iterator_category` **iterator_category**
- typedef `_Iterator` **iterator_type**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::value_type` **value_type**

Public Member Functions

- [_Safe_iterator](#) ()
- [_Safe_iterator](#) (const `_Iterator` &__i, const `_Sequence` *__seq)
- template<typename `_MutableIterator` >
[_Safe_iterator](#) (const [_Safe_iterator](#)< `_MutableIterator`, typename `__gnu_cxx::__enable_if`<(std::__are_same< `_MutableIterator`, typename `__gnu_cxx::iterator_type` >::__value), `_Sequence` >::__type > &__x)
- [_Safe_iterator](#) (const [_Safe_iterator](#) &__x)
- void [_M_attach](#) ([_Safe_sequence_base](#) *__seq, bool __constant)
- void [_M_attach](#) (const `_Sequence` *__seq)
- void [_M_attach_single](#) ([_Safe_sequence_base](#) *__seq, bool __constant) throw ()
- void [_M_attach_single](#) (const `_Sequence` *__seq)
- bool [_M_attached_to](#) (const [_Safe_sequence_base](#) *__seq) const
- bool [_M_before_dereferenceable](#) () const
- bool [_M_can_advance](#) (const `difference_type` &__n) const

- `_GLIBCXX_PURE bool _M_can_compare (const _Safe_iterator_base &__x) const throw ()`
- `bool _M_decrementable () const`
- `bool _M_dereferenceable () const`
- `void _M_detach ()`
- `void _M_detach_single () throw ()`
- `const _Sequence * _M_get_sequence () const`
- `bool _M_incrementable () const`
- `void _M_invalidate ()`
- `void _M_invalidate_single ()`
- `bool _M_is_before_begin () const`
- `bool _M_is_begin () const`
- `bool _M_is_end () const`
- `_GLIBCXX_PURE bool _M_singular () const throw ()`
- `template<typename _Other > bool _M_valid_range (const _Safe_iterator< _Other, _Sequence > &__rhs) const`
- `_Iterator base () const`
- `operator _Iterator () const`
- `reference operator* () const`
- `_Safe_iterator operator+ (const difference_type &__n) const`
- `_Safe_iterator & operator++ ()`
- `_Safe_iterator operator++ (int)`
- `_Safe_iterator & operator+= (const difference_type &__n)`
- `_Safe_iterator operator- (const difference_type &__n) const`
- `_Safe_iterator & operator-- ()`
- `_Safe_iterator operator-- (int)`
- `_Safe_iterator & operator-= (const difference_type &__n)`
- `pointer operator-> () const`
- `_Safe_iterator & operator= (const _Safe_iterator &__x)`
- `reference operator[] (const difference_type &__n) const`

Static Public Member Functions

- `template<typename _Iterator1, typename _Iterator2 > static std::pair< difference_type, _Distance_precision > _M_get_distance (const _Iterator1 &__lhs, const _Iterator2 &__rhs)`
- `template<typename _Iterator1, typename _Iterator2 > static std::pair< difference_type, _Distance_precision > _M_get_distance (const _Iterator1 &__lhs, const _Iterator2 &__rhs, std::forward_iterator_tag)`
- `template<typename _Iterator1, typename _Iterator2 > static std::pair< difference_type, _Distance_precision > _M_get_distance (const _Iterator1 &__lhs, const _Iterator2 &__rhs, std::random_access_iterator_tag)`

Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

Protected Member Functions

- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

5.86.1 Detailed Description

`template<typename _Iterator, typename _Sequence> class __gnu_debug::__Safe_iterator<_Iterator, _Sequence>`

Safe iterator wrapper. The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 76 of file `safe_iterator.h`.

5.86.2 Constructor & Destructor Documentation

5.86.2.1 `template<typename _Iterator, typename _Sequence>
__gnu_debug::__Safe_iterator<_Iterator, _Sequence>::__Safe_iterator
() [inline]`

Postcondition

the iterator is singular and unattached

Definition at line 112 of file `safe_iterator.h`.

5.86.2.2 `template<typename _Iterator, typename _Sequence>
__gnu_debug::__Safe_iterator<_Iterator, _Sequence>::__Safe_iterator
(const _Iterator & __i, const _Sequence * __seq) [inline]`

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

Definition at line 121 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

```
5.86.2.3 template<typename _Iterator, typename _Sequence>
    __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator
    ( const _Safe_iterator< _Iterator, _Sequence > & __x ) [inline]
```

Copy construction.

Definition at line 132 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

```
5.86.2.4 template<typename _Iterator, typename _Sequence>
    template<typename _MutableIterator > __gnu_debug::_Safe_iterator<
    _Iterator, _Sequence >::_Safe_iterator
    ( const _Safe_iterator< _MutableIterator, typename
    __gnu_cxx::__enable_if<(std::__are_same< _MutableIterator,
    typename _Sequence::iterator::iterator_type >::_value), _Sequence
    >::_type > & __x ) [inline]
```

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 149 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

5.86.3 Member Function Documentation

```
5.86.3.1 template<typename _Iterator, typename _Sequence> void
    __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach (
    const _Sequence * __seq ) [inline]
```

Attach iterator to the given sequence.

Definition at line 334 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.86 `__gnu_debug::Safe_iterator<_Iterator, _Sequence>` Class Template Reference 1031

5.86.3.2 `void __gnu_debug::Safe_iterator_base::M_attach (`
`_Safe_sequence_base * __seq, bool __constant) [inherited]`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::Safe_iterator_base::Safe_iterator_base()`.

5.86.3.3 `template<typename _Iterator, typename _Sequence>`
`void __gnu_debug::Safe_iterator<_Iterator, _Sequence`
`>::M_attach_single(const _Sequence * __seq) [inline]`

Likewise, but not thread-safe.

Definition at line 342 of file `safe_iterator.h`.

5.86.3.4 `void __gnu_debug::Safe_iterator_base::M_attach_single`
`(_Safe_sequence_base * __seq, bool __constant) throw ()`
`[inherited]`

Likewise, but not thread-safe.

5.86.3.5 `bool __gnu_debug::Safe_iterator_base::M_attached_to(const`
`_Safe_sequence_base * __seq) const [inline, inherited]`

Determines if we are attached to the given sequence.

Definition at line 130 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_sequence`.

5.86.3.6 `template<typename _Iterator, typename _Sequence>`
`bool __gnu_debug::Safe_iterator<_Iterator, _Sequence`
`>::M_before_dereferenceable() const [inline]`

Is the iterator before a dereferenceable one?

Definition at line 363 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_dereferenceable()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable()`.

5.86.3.7 `_GLIBCXX_PURE bool __gnu_debug::Safe_iterator_base::M_can_compare (const _Safe_iterator_base & __x) const throw ()`
[inherited]

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

5.86.3.8 `template<typename _Iterator, typename _Sequence>`
`bool __gnu_debug::Safe_iterator< _Iterator, _Sequence`
`>::M_dereferenceable () const` **[inline]**

Is the iterator dereferenceable?

Definition at line 358 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_is_before_begin()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_is_end()`, and `__gnu_debug::Safe_iterator_base::M_singular()`.

Referenced by `__gnu_debug::check_dereferenceable()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_before_dereferenceable()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator*()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::operator->()`.

5.86.3.9 `void __gnu_debug::Safe_iterator_base::M_detach ()`
[inherited]

Detach the iterator for whatever sequence it is attached to, if any.

5.86.3.10 `void __gnu_debug::Safe_iterator_base::M_detach_single ()`
`throw ()` **[inherited]**

Likewise, but not thread-safe.

5.86.3.11 `template<typename _Iterator, typename _Sequence>`
`template<typename _Iterator1, typename _Iterator2 >`
`static std::pair<difference_type, Distance_precision>`
`__gnu_debug::Safe_iterator< _Iterator, _Sequence`
`>::M_get_distance (const _Iterator1 & __lhs, const _Iterator2 &`
`__rhs)` **[inline, static]**

Determine the distance between two iterators with some known precision.

Definition at line 397 of file `safe_iterator.h`.

5.86 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>` Class Template Reference 1033

5.86.3.12 `__gnu_cxx::mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw () [protected, inherited]`

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate()`.

5.86.3.13 `template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_incrementable () const [inline]`

Is the iterator incrementable?

Definition at line 371 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_before_dereferenceable()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator++()`.

5.86.3.14 `template<typename _Iterator, typename _Sequence>
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate ()`

Invalidate the iterator, making it singular.

Definition at line 106 of file `safe_iterator.tcc`.

References `__gnu_debug::_Safe_iterator_base::_M_get_mutex()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`.

5.86.3.15 `template<typename _Iterator, typename _Sequence>
void __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single ()`

Likewise, but not thread-safe.

Definition at line 115 of file `safe_iterator.tcc`.

References `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_debug::_Safe_sequence_base::_M_iterators`, `__gnu_debug::_Safe_iterator_base::_M_next`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, `__gnu_debug::_Safe_iterator_base::_M_singular()`, `__gnu_debug::_Safe_iterator_base::_M_version`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::base()`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate()`.

5.86.3.16 `template<typename _Iterator, typename _Sequence>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence
>::_M_is_before_begin () const [inline]`

Is this iterator equal to the sequence's `before_begin()` iterator if /// any?

Definition at line 431 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

5.86.3.17 `template<typename _Iterator, typename _Sequence> bool
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin
() const [inline]`

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 422 of file `safe_iterator.h`.

5.86.3.18 `template<typename _Iterator, typename _Sequence> bool
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end (
) const [inline]`

Is this iterator equal to the sequence's `end()` iterator?

Definition at line 426 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`.

5.86.3.19 `_GLIBCXX_PURE bool __gnu_debug::_Safe_
iterator_base::_M_singular () const throw ()
[inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::__check_singular()`, `__gnu_debug::__check_singular_aux()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.86 `__gnu_debug::Safe_iterator<_Iterator, _Sequence>` Class Template Reference 1035

5.86.3.20 `template<typename _Iterator, typename _Sequence> _Iterator
__gnu_debug::Safe_iterator<_Iterator, _Sequence>::base ()
const [inline]`

Return the underlying iterator.

Definition at line 324 of file `safe_iterator.h`.

Referenced by `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.86.3.21 `template<typename _Iterator, typename _Sequence>
__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator
_Iterator () const [inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 330 of file `safe_iterator.h`.

5.86.3.22 `template<typename _Iterator, typename _Sequence> reference
__gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator* () const [inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 188 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_dereferenceable()`.

5.86.3.23 `template<typename _Iterator, typename _Sequence>
_Safe_iterator& __gnu_debug::Safe_iterator<_Iterator, _Sequence>::operator++ () [inline]`

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 217 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`.

5.86.3.24 `template<typename _Iterator, typename _Sequence> _Safe_iterator
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++ (
int) [inline]`

Iterator postincrement.

Precondition

iterator is incrementable

Definition at line 231 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`.

5.86.3.25 `template<typename _Iterator, typename _Sequence> _Safe_iterator
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-- (
int) [inline]`

Iterator postdecrement.

Precondition

iterator is decrementable

Definition at line 261 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

5.86.3.26 `template<typename _Iterator, typename _Sequence>
_Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence
>::operator-- () [inline]`

Iterator predecrement.

Precondition

iterator is decrementable

Definition at line 247 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`.

5.86 __gnu_debug::_Safe_iterator< _Iterator, _Sequence > Class Template Reference 1037

5.86.3.27 `template<typename _Iterator, typename _Sequence> pointer
__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-> (
) const [inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Todo

Make this correct w.r.t. iterators that return proxies

Use `addressof()` instead of `&` operator

Definition at line 203 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`.

5.86.3.28 `template<typename _Iterator, typename _Sequence>
__Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence
>::operator= (const __Safe_iterator< _Iterator, _Sequence > & __x
) [inline]`

Copy assignment.

Definition at line 169 of file `safe_iterator.h`.

References `_GLIBCXX_DEBUG_VERIFY`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach()`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

5.86.4 Member Data Documentation

5.86.4.1 `__Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next
[inherited]`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 73 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

5.86.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior` [`inherited`]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 69 of file `safe_base.h`.

5.86.4.3 `_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence` [`inherited`]

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 56 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`, `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.86.4.4 `unsigned int __gnu_debug::_Safe_iterator_base::_M_version` [`inherited`]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 65 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

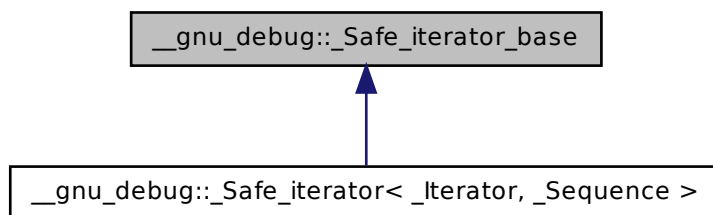
The documentation for this class was generated from the following files:

- [safe_iterator.h](#)
- [safe_iterator.tcc](#)

5.87 `__gnu_debug::_Safe_iterator_base` Class Reference

Basic functionality for a *safe* iterator.

Inheritance diagram for __gnu_debug::_Safe_iterator_base:



Public Member Functions

- void `_M_attach` (`_Safe_sequence_base *__seq`, bool `__constant`)
- void `_M_attach_single` (`_Safe_sequence_base *__seq`, bool `__constant`) throw ()
- bool `_M_attached_to` (const `_Safe_sequence_base *__seq`) const
- `_GLIBCXX_PURE` bool `_M_can_compare` (const `_Safe_iterator_base &__x`) const throw ()
- void `_M_detach` ()
- void `_M_detach_single` () throw ()
- `_GLIBCXX_PURE` bool `_M_singular` () const throw ()

Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- unsigned int `_M_version`

Protected Member Functions

- `_Safe_iterator_base` ()
- `_Safe_iterator_base` (const `_Safe_sequence_base *__seq`, bool `__constant`)
- `_Safe_iterator_base` (const `_Safe_iterator_base &`)
- `_Safe_iterator_base` (const `_Safe_iterator_base &__x`, bool `__constant`)
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `_Safe_iterator_base & operator=` (const `_Safe_iterator_base &`)

5.87.1 Detailed Description

Basic functionality for a *safe* iterator. The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 51 of file `safe_base.h`.

5.87.2 Constructor & Destructor Documentation

5.87.2.1 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ()` [`inline`, `protected`]

Initializes the iterator and makes it singular.

Definition at line 77 of file `safe_base.h`.

5.87.2.2 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (const _Safe_sequence_base * __seq, bool __constant)` [`inline`, `protected`]

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 88 of file `safe_base.h`.

References `_M_attach()`.

5.87.2.3 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (const _Safe_iterator_base & __x, bool __constant)` [`inline`, `protected`]

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 95 of file `safe_base.h`.

References `_M_attach()`, and `_M_sequence`.

5.87.3 Member Function Documentation

5.87.3.1 `void __gnu_debug::_Safe_iterator_base::_M_attach (
_Safe_sequence_base * __seq, bool __constant)`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `_Safe_iterator_base()`.

5.87.3.2 `void __gnu_debug::_Safe_iterator_base::_M_attach_single (
_Safe_sequence_base * __seq, bool __constant) throw ()`

Likewise, but not thread-safe.

5.87.3.3 `bool __gnu_debug::_Safe_iterator_base::_M_attached_to (const
_Safe_sequence_base * __seq) const [inline]`

Determines if we are attached to the given sequence.

Definition at line 130 of file `safe_base.h`.

References `_M_sequence`.

5.87.3.4 `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_
can_compare (const _Safe_iterator_base & __x) const throw
()`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

5.87.3.5 `void __gnu_debug::_Safe_iterator_base::_M_detach ()`

Detach the iterator for whatever sequence it is attached to, if any.

5.87.3.6 `void __gnu_debug::_Safe_iterator_base::_M_detach_single () throw
()`

Likewise, but not thread-safe.

5.87.3.7 `__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw () [protected]`

For use in [_Safe_iterator](#).

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate()`.

5.87.3.8 `_GLIBCXX_PURE bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw ()`

Is this iterator singular?

Referenced by `__gnu_debug::__check_singular()`, `__gnu_debug::__check_singular_aux()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.87.4 Member Data Documentation

5.87.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 73 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

5.87.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 69 of file `safe_base.h`.

5.87.4.3 `_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence`

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 56 of file `safe_base.h`.

5.88 __gnu_debug::_Safe_sequence< _Sequence > Class Template Reference

Referenced by `_M_attached_to()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`, `_Safe_iterator_base()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

5.87.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 65 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`.

The documentation for this class was generated from the following file:

- [safe_base.h](#)

5.88 __gnu_debug::_Safe_sequence< _Sequence > Class Template Reference

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

Inheritance diagram for `__gnu_debug::_Safe_sequence< _Sequence >`:



Public Member Functions

- `void _M_invalidate_all () const`

- `template<typename _Predicate >`
`void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Iterator >`
`void _M_transfer_iter (const _Safe_iterator< _Iterator, _Sequence > &__x)`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

5.88.1 Detailed Description

`template<typename _Sequence> class __gnu_debug::_Safe_sequence< _Sequence >`

Base class for constructing a *safe* sequence type that tracks iterators that reference it. The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

Definition at line 97 of file `safe_sequence.h`.

5.88.2 Member Function Documentation

5.88.2.1 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ()`
`[protected, inherited]`

Detach all iterators, leaving them singular.

5.88 `__gnu_debug::_Safe_sequence<_Sequence>` Class Template Reference 045

5.88.2.2 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.88.2.3 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw ()` [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.88.2.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const`
[inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.88.2.5 `template<typename _Sequence> template<typename _Predicate> void __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if (_Predicate __pred)`

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

Definition at line 121 of file `safe_sequence.h`.

References `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_debug::_Safe_sequence_base::_M_get_mutex()`, and `__gnu_debug::_Safe_sequence_base::_M_iterators`.

5.88.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ()`
[protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.88.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (` `_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.88.2.8 `template<typename _Sequence> template<typename _Iterator >` `void __gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter` `(const _Safe_iterator<_Iterator, _Sequence > & __x)`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

Definition at line 154 of file `safe_sequence.h`.

References `__gnu_debug::_Safe_sequence_base::_M_const_iterators`, `__gnu_debug::_Safe_sequence_base::_M_get_mutex()`, `__gnu_debug::_Safe_sequence_base::_M_iterators`, `__gnu_debug::_Safe_iterator_base::_M_sequence`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::base()`.

5.88.3 Member Data Documentation

5.88.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M-` `const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

5.88.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M-` `iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence >::_M_transfer_iter()`.

5.88.3.3 unsigned int __gnu_debug::__Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

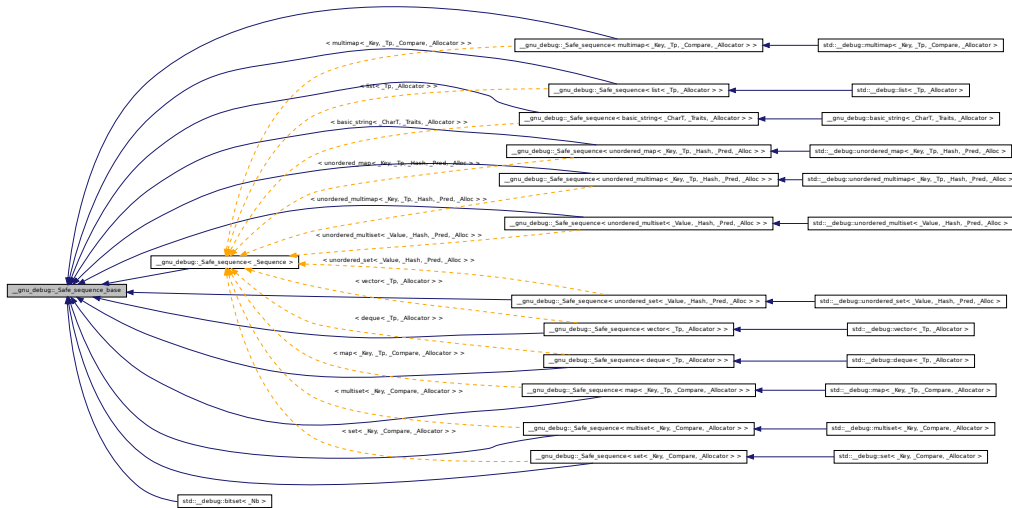
The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

5.89 __gnu_debug::__Safe_sequence_base Class Reference

Base class that supports tracking of iterators that reference a sequence.

Inheritance diagram for __gnu_debug::__Safe_sequence_base:



Public Member Functions

- [void _M_invalidate_all\(\) const](#)

Public Attributes

- [_Safe_iterator_base * _M_const_iterators](#)

- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- [~_Safe_sequence_base](#) ()
- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) ([_Safe_sequence_base](#) &__x)

5.89.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence. The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit [_Safe_iterator_base](#)) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 159 of file `safe_base.h`.

5.89.2 Constructor & Destructor Documentation

5.89.2.1 `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base ()` [inline, protected]

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 179 of file `safe_base.h`.

5.89.3 Member Function Documentation

5.89.3.1 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ()` [protected]

Detach all iterators, leaving them singular.

5.89.3.2 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ()` `[protected]`

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.89.3.3 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw ()` `[protected]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.89.3.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const` `[inline]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.89.3.5 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ()` `[protected]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.89.3.6 `void __gnu_debug::_Safe_sequence_base::_M_swap (` `_Safe_sequence_base & __x)` `[protected]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.89.4 Member Data Documentation

5.89.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.89.4.2 `_Safe_iterator_base*` `__gnu_debug::_Safe_sequence_base::_M_iterators`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.89.4.3 `unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` [mutable]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

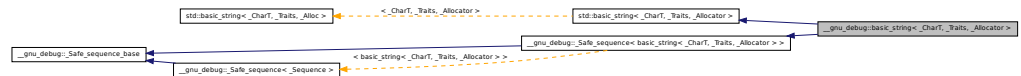
The documentation for this class was generated from the following file:

- [safe_base.h](#)

5.90 `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >` Class Template Reference

Class `std::basic_string` with safety/checking/debug instrumentation.

Inheritance diagram for `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`:



Public Types

- `typedef _Allocator allocator_type`

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::_Safe_iterator<typename _Base::const_iterator, basic_string>` **const_iterator**
- typedef `__gnu_cxx::__normal_iterator<const_pointer, basic_string>` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_CharT_alloc_type::const_pointer` **const_pointer**
- typedef `_CharT_alloc_type::const_reference` **const_reference**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator<const_iterator>` **const_reverse_iterator**
- typedef `std::reverse_iterator<const_iterator>` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `_CharT_alloc_type::difference_type` **difference_type**
- typedef `__gnu_cxx::__normal_iterator<pointer, basic_string>` **iterator**
- typedef `__gnu_debug::_Safe_iterator<typename _Base::iterator, basic_string>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse_iterator**
- typedef `std::reverse_iterator<iterator>` **reverse_iterator**
- typedef `_CharT_alloc_type::size_type` **size_type**
- typedef `_Base::size_type` **size_type**
- typedef `_Traits` **traits_type**
- typedef `_Traits` **traits_type**
- typedef `_Traits::char_type` **value_type**
- typedef `_Traits::char_type` **value_type**

Public Member Functions

- **basic_string** (`const _Allocator &__a=_Allocator()`)
- **basic_string** (`const _Base &__base`)
- **basic_string** (`const basic_string &__str, size_type __pos, size_type __n=_Base::npos, const _Allocator &__a=_Allocator()`)
- `template<typename _InputIterator>`
basic_string (`_InputIterator __begin, _InputIterator __end, const _Allocator &__a=_Allocator()`)
- **basic_string** (`basic_string &&__str`)
- **basic_string** (`const _CharT *__s, size_type __n, const _Allocator &__a=_Allocator()`)
- **basic_string** (`std::initializer_list<_CharT> __l, const _Allocator &__a=_Allocator()`)

- **basic_string** (const [basic_string](#) &__str)
- **basic_string** (const _CharT *__s, const _Allocator &__a=_Allocator())
- **basic_string** (size_type __n, _CharT __c, const _Allocator &__a=_Allocator())
- [_Base](#) & [_M_base](#) ()
- const [_Base](#) & [_M_base](#) () const
- void [_M_invalidate_all](#) () const
- void [_M_invalidate_if](#) (_Predicate __pred)
- void [_M_transfer_iter](#) (const [_Safe_iterator](#)< _Iterator, [basic_string](#)< _CharT, _Traits, _Allocator > > &__x)
- [basic_string](#) & **append** (const [basic_string](#) &__str)
- [basic_string](#) & **append** (const [basic_string](#) &__str, size_type __pos, size_type __n)
- [basic_string](#) & **append** (const [basic_string](#) &__str)
- [basic_string](#) & **append** (const _CharT *__s, size_type __n)
- [basic_string](#) & **append** (const [basic_string](#) &__str, size_type __pos, size_type __n)
- [basic_string](#) & **append** (const _CharT *__s, size_type __n)
- [basic_string](#) & **append** (const _CharT *__s)
- [basic_string](#) & **append** (size_type __n, _CharT __c)
- [basic_string](#) & **append** (const _CharT *__s)
- [basic_string](#) & **append** (initializer_list< _CharT > __l)
- [basic_string](#) & **append** (_InputIterator __first, _InputIterator __last)
- [basic_string](#) & **append** (size_type __n, _CharT __c)
- template<typename _InputIterator >
[basic_string](#) & **append** (_InputIterator __first, _InputIterator __last)
- [basic_string](#) & **assign** (const [basic_string](#) &__str)
- [basic_string](#) & **assign** ([basic_string](#) &&__str)
- [basic_string](#) & **assign** (const [basic_string](#) &__str, size_type __pos, size_type __n)
- [basic_string](#) & **assign** (const _CharT *__s, size_type __n)
- [basic_string](#) & **assign** (const _CharT *__s)
- [basic_string](#) & **assign** (size_type __n, _CharT __c)
- [basic_string](#) & **assign** (_InputIterator __first, _InputIterator __last)
- [basic_string](#) & **assign** (initializer_list< _CharT > __l)
- [basic_string](#) & **assign** (const [basic_string](#) &__x)
- [basic_string](#) & **assign** ([basic_string](#) &&__x)
- [basic_string](#) & **assign** (const [basic_string](#) &__str, size_type __pos, size_type __n)
- [basic_string](#) & **assign** (const _CharT *__s, size_type __n)
- [basic_string](#) & **assign** (const _CharT *__s)
- [basic_string](#) & **assign** (size_type __n, _CharT __c)

- `template<typename _InputIterator>`
`basic_string & assign (_InputIterator __first, _InputIterator __last)`
- `basic_string & assign (std::initializer_list<_CharT> __l)`
- `const_reference at (size_type __n) const`
- `reference at (size_type __n)`
- `reference back ()`
- `const_reference back () const`
- `iterator begin ()`
- `const_iterator begin () const`
- `iterator begin ()`
- `const_iterator begin () const`
- `const _CharT * c_str () const`
- `const _CharT * c_str () const`
- `size_type capacity () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `void clear ()`
- `int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const`
- `int compare (size_type __pos, size_type __n1, const _CharT *__s) const`
- `int compare (const basic_string &__str) const`
- `int compare (size_type __pos1, size_type __n1, const basic_string &__str) const`
- `int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2) const`
- `int compare (size_type __pos1, size_type __n1, const _CharT *__s) const`
- `int compare (const _CharT *__s) const`
- `int compare (size_type __pos1, size_type __n1, const _CharT *__s, size_type __n2) const`
- `int compare (size_type __pos, size_type __n, const basic_string &__str) const`
- `int compare (const _CharT *__s) const`
- `int compare (const basic_string &__str) const`
- `int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2) const`
- `size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const`
- `size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const`
- `const_reverse_iterator crbegin () const`
- `const_reverse_iterator crend () const`
- `const _CharT * data () const`
- `const _CharT * data () const`
- `bool empty () const`
- `iterator end ()`

- [iterator](#) **end** ()
- [const_iterator](#) **end** () const
- [const_iterator](#) **end** () const
- [basic_string](#) & **erase** (size_type __pos=0, size_type __n=[npos](#))
- [iterator](#) **erase** ([iterator](#) __position)
- [iterator](#) **erase** ([iterator](#) __first, [iterator](#) __last)
- [basic_string](#) & **erase** (size_type __pos=0, size_type __n=[_Base::npos](#))
- [iterator](#) **erase** ([iterator](#) __position)
- [iterator](#) **erase** ([iterator](#) __first, [iterator](#) __last)
- size_type **find** (const [basic_string](#) &__str, size_type __pos=0) const
- size_type **find** (const [basic_string](#) &__str, size_type __pos=0) const
- size_type **find** (const _CharT *__s, size_type __pos, size_type __n) const
- size_type **find** (const _CharT *__s, size_type __pos=0) const
- size_type **find** (_CharT __c, size_type __pos=0) const
- size_type **find** (const _CharT *__s, size_type __pos, size_type __n) const
- size_type **find** (const _CharT *__s, size_type __pos=0) const
- size_type **find** (_CharT __c, size_type __pos=0) const
- size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const
- size_type **find_first_not_of** (const _CharT *__s, size_type __pos, size_type __n) const
- size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const
- size_type **find_first_not_of** (const _CharT *__s, size_type __pos, size_type __n) const
- size_type **find_first_not_of** (const [basic_string](#) &__str, size_type __pos=0) const
- size_type **find_first_not_of** (const [basic_string](#) &__str, size_type __pos=0) const
- size_type **find_first_not_of** (const _CharT *__s, size_type __pos=0) const
- size_type **find_first_of** (const _CharT *__s, size_type __pos, size_type __n) const
- size_type **find_first_of** (const _CharT *__s, size_type __pos=0) const
- size_type **find_first_of** (_CharT __c, size_type __pos=0) const
- size_type **find_first_of** (const [basic_string](#) &__str, size_type __pos=0) const
- size_type **find_first_of** (const _CharT *__s, size_type __pos, size_type __n) const
- size_type **find_first_of** (const _CharT *__s, size_type __pos=0) const
- size_type **find_first_of** (_CharT __c, size_type __pos=0) const
- size_type **find_first_of** (const [basic_string](#) &__str, size_type __pos=0) const
- size_type **find_last_not_of** (_CharT __c, size_type __pos=[_Base::npos](#)) const
- size_type **find_last_not_of** (const _CharT *__s, size_type __pos, size_type __n) const
- size_type **find_last_not_of** (const [basic_string](#) &__str, size_type __pos=[_Base::npos](#)) const
- size_type **find_last_not_of** (const [basic_string](#) &__str, size_type __pos=[_Base::npos](#)) const

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1055

- `size_type find_last_not_of` (const `basic_string` &__str, `size_type` __pos=`npos`) const
- `size_type find_last_not_of` (const `_CharT` *__s, `size_type` __pos=`npos`) const
- `size_type find_last_not_of` (`_CharT` __c, `size_type` __pos=`npos`) const
- `size_type find_last_not_of` (const `_CharT` *__s, `size_type` __pos, `size_type` __n) const
- `size_type find_last_not_of` (const `_CharT` *__s, `size_type` __pos=`_Base::npos`) const
- `size_type find_last_of` (`_CharT` __c, `size_type` __pos=`npos`) const
- `size_type find_last_of` (const `_CharT` *__s, `size_type` __pos, `size_type` __n) const
- `size_type find_last_of` (const `basic_string` &__str, `size_type` __pos=`_Base::npos`) const
- `size_type find_last_of` (const `_CharT` *__s, `size_type` __pos, `size_type` __n) const
- `size_type find_last_of` (const `_CharT` *__s, `size_type` __pos=`_Base::npos`) const

- `size_type find_last_of` (`_CharT` __c, `size_type` __pos=`_Base::npos`) const
- `size_type find_last_of` (const `basic_string` &__str, `size_type` __pos=`npos`) const
- `size_type find_last_of` (const `_CharT` *__s, `size_type` __pos=`npos`) const
- reference `front` ()
- const_reference `front` () const
- allocator_type `get_allocator` () const
- `basic_string` & `insert` (`size_type` __pos, `size_type` __n, `_CharT` __c)
- void `insert` (iterator __p, `_InputIterator` __beg, `_InputIterator` __end)
- void `insert` (iterator __p, `initializer_list`< `_CharT` > __l)
- `basic_string` & `insert` (`size_type` __pos1, const `basic_string` &__str)
- void `insert` (iterator __p, `size_type` __n, `_CharT` __c)
- `basic_string` & `insert` (`size_type` __pos, const `_CharT` *__s, `size_type` __n)
- `basic_string` & `insert` (`size_type` __pos, `size_type` __n, `_CharT` __c)
- `basic_string` & `insert` (`size_type` __pos1, const `basic_string` &__str, `size_type` __pos2, `size_type` __n)
- `basic_string` & `insert` (`size_type` __pos1, const `basic_string` &__str, `size_type` __pos2, `size_type` __n)
- iterator `insert` (iterator __p, `_CharT` __c)
- `basic_string` & `insert` (`size_type` __pos1, const `basic_string` &__str)
- `basic_string` & `insert` (`size_type` __pos, const `_CharT` *__s)
- iterator `insert` (iterator __p, `_CharT` __c)
- void `insert` (iterator __p, `size_type` __n, `_CharT` __c)
- `basic_string` & `insert` (`size_type` __pos, const `_CharT` *__s)
- void `insert` (iterator __p, `std::initializer_list`< `_CharT` > __l)
- `basic_string` & `insert` (`size_type` __pos, const `_CharT` *__s, `size_type` __n)
- template<typename `_InputIterator` >
void `insert` (iterator __p, `_InputIterator` __first, `_InputIterator` __last)

- `size_type length () const`
- `size_type max_size () const`
- `basic_string & operator+= (_CharT __c)`
- `basic_string & operator+= (std::initializer_list< _CharT > __l)`
- `basic_string & operator+= (const basic_string &__str)`
- `basic_string & operator+= (const _CharT *__s)`
- `basic_string & operator+= (_CharT __c)`
- `basic_string & operator+= (initializer_list< _CharT > __l)`
- `basic_string & operator+= (const basic_string &__str)`
- `basic_string & operator+= (const _CharT *__s)`
- `basic_string & operator= (std::initializer_list< _CharT > __l)`
- `basic_string & operator= (const basic_string &__str)`
- `basic_string & operator= (basic_string &&__str)`
- `basic_string & operator= (_CharT __c)`
- `basic_string & operator= (const _CharT *__s)`
- `const_reference operator[] (size_type __pos) const`
- `reference operator[] (size_type __pos)`
- `const_reference operator[] (size_type __pos) const`
- `reference operator[] (size_type __pos)`
- `void push_back (_CharT __c)`
- `void push_back (_CharT __c)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rend () const`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `reverse_iterator rend ()`
- `template<typename _InputIterator >`
`basic_string & replace (iterator __i1, iterator __i2, _InputIterator __j1, _-`
`InputIterator __j2)`
- `basic_string & replace (iterator __i1, iterator __i2, const basic_string &__str)`
- `basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)`
- `basic_string & replace (iterator __i1, iterator __i2, initializer_list< _CharT >`
`__l)`
- `basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _-`
`InputIterator __k2)`
- `basic_string & replace (size_type __pos, size_type __n, const basic_string &__-`
`str)`
- `basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s, size_type`
`__n)`

5.90 `_gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1057

- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `std::initializer_list`< `_CharT` > __l)
- `basic_string` & **replace** (`size_type` __pos, `size_type` __n1, `const` `_CharT` *__s)
- `basic_string` & **replace** (`size_type` __pos, `size_type` __n1, `const` `_CharT` *__s, `size_type` __n2)
- `basic_string` & **replace** (`size_type` __pos1, `size_type` __n1, `const` `basic_string` & __str)
- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `const` `basic_string` & __str)
- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `const` `_CharT` *__k1, `const` `_CharT` *__k2)
- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `const` `_CharT` *__s)
- `basic_string` & **replace** (`size_type` __pos1, `size_type` __n1, `const` `basic_string` & __str, `size_type` __pos2, `size_type` __n2)
- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `const` `_CharT` *__s, `size_type` __n)
- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `const` `_CharT` *__s)
- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `_CharT` *__k1, `_CharT` *__k2)
- `basic_string` & **replace** (`size_type` __pos, `size_type` __n1, `const` `_CharT` *__s, `size_type` __n2)
- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `iterator` __k1, `iterator` __k2)
- `basic_string` & **replace** (`size_type` __pos, `size_type` __n1, `const` `_CharT` *__s)
- `basic_string` & **replace** (`iterator` __i1, `iterator` __i2, `size_type` __n, `_CharT` __c)
- `basic_string` & **replace** (`size_type` __pos, `size_type` __n1, `size_type` __n2, `_CharT` __c)
- `basic_string` & **replace** (`size_type` __pos1, `size_type` __n1, `const` `basic_string` & __str, `size_type` __pos2, `size_type` __n2)
- `basic_string` & **replace** (`size_type` __pos, `size_type` __n1, `size_type` __n2, `_CharT` __c)
- `void` **reserve** (`size_type` __res_arg=0)
- `void` **resize** (`size_type` __n)
- `void` **resize** (`size_type` __n, `_CharT` __c)
- `void` **resize** (`size_type` __n, `_CharT` __c)
- `void` **resize** (`size_type` __n)
- `size_type` **rfind** (`const` `_CharT` *__s, `size_type` __pos, `size_type` __n) `const`
- `size_type` **rfind** (`const` `_CharT` *__s, `size_type` __pos=`_Base::npos`) `const`
- `size_type` **rfind** (`_CharT` __c, `size_type` __pos=`npos`) `const`
- `size_type` **rfind** (`const` `basic_string` & __str, `size_type` __pos=`_Base::npos`) `const`
- `size_type` **rfind** (`_CharT` __c, `size_type` __pos=`_Base::npos`) `const`
- `size_type` **rfind** (`const` `basic_string` & __str, `size_type` __pos=`npos`) `const`
- `size_type` **rfind** (`const` `_CharT` *__s, `size_type` __pos=`npos`) `const`
- `size_type` **rfind** (`const` `_CharT` *__s, `size_type` __pos, `size_type` __n) `const`

- void [shrink_to_fit](#) ()
- size_type [size](#) () const
- [basic_string](#) [substr](#) (size_type __pos=0, size_type __n=[npos](#)) const
- [basic_string](#) [substr](#) (size_type __pos=0, size_type __n=[_Base::npos](#)) const
- void [swap](#) ([basic_string](#)<_CharT, _Traits, _Allocator > &__x)
- void [swap](#) ([basic_string](#) &__s)

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Static Public Attributes

- static const size_type [npos](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) ([_Safe_sequence_base](#) &__x)

5.90.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>,
typename _Allocator = std::allocator<_CharT>> class [__gnu_debug::basic_string](#)<_CharT, _Traits, _Allocator >

Class [std::basic_string](#) with safety/checking/debug instrumentation.

Definition at line 42 of file debug/string.

5.90.2 Member Function Documentation

5.90.2.1 void [__gnu_debug::_Safe_sequence_base::_M_detach_all](#) () [protected, inherited]

Detach all iterators, leaving them singular.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1059

5.90.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
`[protected, inherited]`

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.90.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw ()` `[protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence<_Sequence>::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.90.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const`
`[inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.90.2.5 `void __gnu_debug::Safe_sequence<basic_string<_CharT, _Traits, _Allocator>>::_M_invalidate_if (_Predicate __pred)`
`[inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.90.2.6 `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ()`
`[protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.90.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (`
`_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.90.2.8 `void __gnu_debug::_Safe_sequence< basic_string< _CharT,`
`_Traits, _Allocator > >::_M_transfer_iter (const _Safe_iterator<`
`_Iterator, basic_string< _CharT, _Traits, _Allocator > > & __x)`
`[inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.90.2.9 `basic_string& std::basic_string< _CharT, _Traits, _Allocator`
`>::append (const basic_string< _CharT, _Traits, _Allocator > &`
`__str) [inherited]`

Append a string to this string.

Parameters

str The string to append.

Returns

Reference to this string.

5.90.2.10 `basic_string& std::basic_string< _CharT, _Traits, _Allocator`
`>::append (const basic_string< _CharT, _Traits, _Allocator > &`
`__str, size_type __pos, size_type __n) [inherited]`

Append a substring.

Parameters

str The string to append.

pos Index of the first character of *str* to append.

n The number of characters to append.

Returns

Reference to this string.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1061

Exceptions

[*std::out_of_range*](#) if *pos* is not a valid index.

This function appends *n* characters from *str* starting at *pos* to this string. If *n* is larger than the number of available characters in *str*, the remainder of *str* is appended.

5.90.2.11 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (const _CharT * __s, size_type __n) [inherited]`

Append a C substring.

Parameters

s The C string to append.

n The number of characters to append.

Returns

Reference to this string.

5.90.2.12 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (const _CharT * __s) [inline, inherited]`

Append a C string.

Parameters

s The C string to append.

Returns

Reference to this string.

Definition at line 995 of file `basic_string.h`.

5.90.2.13 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (size_type __n, _CharT __c) [inherited]`

Append multiple characters.

Parameters

n The number of characters to append.

c The character to use.

Returns

Reference to this string.

Appends *n* copies of *c* to this string.

5.90.2.14 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (initializer_list<_CharT> __l) [inline, inherited]`

Append an `initializer_list` of characters.

Parameters

l The `initializer_list` of characters to append.

Returns

Reference to this string.

Definition at line 1019 of file `basic_string.h`.

5.90.2.15 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (_InputIterator __first, _InputIterator __last) [inline, inherited]`

Append a range of characters.

Parameters

first Iterator referencing the first character to append.

last Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range `[first,last)` to this string.

Definition at line 1033 of file `basic_string.h`.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1063

5.90.2.16 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const basic_string<_CharT, _Traits, _Allocator> & __str) [inherited]`

Set value to contents of another string.

Parameters

str Source string to use.

Returns

Reference to this string.

5.90.2.17 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (basic_string<_CharT, _Traits, _Allocator> && __str) [inline, inherited]`

Set value to contents of another string.

Parameters

str Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1068 of file `basic_string.h`.

5.90.2.18 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos, size_type __n) [inline, inherited]`

Set value to a substring of a string.

Parameters

str The string to use.

pos Index of the first character of *str*.

n Number of characters to use.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) if *pos* is not a valid index.

This function sets this string to the substring of *str* consisting of *n* characters at *pos*. If *n* is larger than the number of available characters in *str*, the remainder of *str* is used.

Definition at line 1088 of file `basic_string.h`.

5.90.2.19 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const _CharT * __s, size_type __n) [inherited]`

Set value to a C substring.

Parameters

s The C string to use.

n Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first *n* characters of *s*. If *n* is larger than the number of available characters in *s*, the remainder of *s* is used.

5.90.2.20 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (const _CharT * __s) [inline, inherited]`

Set value to contents of a C string.

Parameters

s The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of *s*. The data is copied, so there is no dependence on *s* once the function returns.

Definition at line 1116 of file `basic_string.h`.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1065

5.90.2.21 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (size_type __n, _CharT __c) [inline, inherited]`

Set value to multiple characters.

Parameters

n Length of the resulting string.

c The character to use.

Returns

Reference to this string.

This function sets the value of this string to *n* copies of character *c*.

Definition at line 1132 of file `basic_string.h`.

5.90.2.22 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (_InputIterator __first, _InputIterator __last) [inline, inherited]`

Set value to a range of characters.

Parameters

first Iterator referencing the first character to append.

last Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

Definition at line 1145 of file `basic_string.h`.

5.90.2.23 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (initializer_list<_CharT> __l) [inline, inherited]`

Set value to an `initializer_list` of characters.

Parameters

l The `initializer_list` of characters to assign.

Returns

Reference to this string.

Definition at line 1155 of file basic_string.h.

5.90.2.24 `const_reference std::basic_string<_CharT,_Traits,_Allocator>::at (size_type __n) const [inline, inherited]`

Provides access to the data contained in the string.

Parameters

n The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 854 of file basic_string.h.

5.90.2.25 `reference std::basic_string<_CharT,_Traits,_Allocator>::at (size_type __n) [inline, inherited]`

Provides access to the data contained in the string.

Parameters

n The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 907 of file basic_string.h.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1067

5.90.2.26 `reference std::basic_string<_CharT, _Traits, _Allocator>::back ()` `[inline, inherited]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 883 of file `basic_string.h`.

5.90.2.27 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::back () const` `[inline, inherited]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 891 of file `basic_string.h`.

5.90.2.28 `iterator std::basic_string<_CharT, _Traits, _Allocator>::begin ()` `[inline, inherited]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 598 of file `basic_string.h`.

5.90.2.29 `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::begin () const` `[inline, inherited]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 609 of file `basic_string.h`.

5.90.2.30 `const _CharT* std::basic_string<_CharT, _Traits, _Allocator>::c_str () const` `[inline, inherited]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1764 of file `basic_string.h`.

5.90.2.31 `size_type std::basic_string<_CharT, _Traits, _Allocator>::capacity () const` `[inline, inherited]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 766 of file `basic_string.h`.

5.90.2.32 `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::cbegin () const` `[inline, inherited]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 673 of file basic_string.h.

5.90.2.33 `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::cend () const` `[inline, inherited]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 681 of file basic_string.h.

5.90.2.34 `void std::basic_string<_CharT, _Traits, _Allocator>::clear ()` `[inline, inherited]`

Erases the string, making it empty.

Definition at line 793 of file basic_string.h.

5.90.2.35 `int std::basic_string<_CharT, _Traits, _Allocator>::compare (const _CharT * __s) const` `[inherited]`

Compare to a C string.

Parameters

s C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *s*, 0 if their values are equivalent, or > 0 if this string is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of *size()* and the length of a string constructed from *s*. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1069

5.90.2.36 `int std::basic_string<_CharT, _Traits, _Allocator>::compare (const basic_string<_CharT, _Traits, _Allocator> & __str) const [inline, inherited]`

Compare to a string.

Parameters

str String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before *str*, 0 if their values are equivalent, or > 0 if this string is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of *size()* and *str.size()*. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2171 of file `basic_string.h`.

5.90.2.37 `int std::basic_string<_CharT, _Traits, _Allocator>::compare (size_type __pos, size_type __n, const basic_string<_CharT, _Traits, _Allocator> & __str) const [inherited]`

Compare substring to a string.

Parameters

pos Index of first character of substring.

n Number of characters in substring.

str String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the *n* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *str.size()*. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

5.90.2.38 `int std::basic_string<_CharT, _Traits, _Allocator >::compare (
 size_type __pos1, size_type __n1, const basic_string<_CharT,
 _Traits, _Allocator > & __str, size_type __pos2, size_type __n2)
 const [inherited]`

Compare substring to a substring.

Parameters

pos1 Index of first character of substring.

n1 Number of characters in substring.

str String to compare against.

pos2 Index of first character of substring of *str*.

n2 Number of characters in substring of *str*.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form the substring of *str* from the *n2* characters starting at *pos2*. Returns an integer < 0 if this substring is ordered before the substring of *str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

5.90.2.39 `int std::basic_string<_CharT, _Traits, _Allocator >::compare (
 size_type __pos, size_type __n1, const _CharT * __s, size_type
 __n2) const [inherited]`

Compare substring against a character array.

Parameters

pos1 Index of first character of substring.

n1 Number of characters in substring.

s character array to compare against.

n2 Number of characters of *s*.

Returns

Integer < 0, 0, or > 0.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1071

Form the substring of this string from the *n1* characters starting at *pos1*. Form a string from the first *n2* characters of *s*. Returns an integer < 0 if this substring is ordered before the string from *s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *n2*. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: *s* must have at least *n2* characters, `'\0'` has no special meaning.

5.90.2.40 `int std::basic_string<_CharT, _Traits, _Allocator>::compare (size_type __pos, size_type __n1, const _CharT * __s) const`
[*inherited*]

Compare substring to a C string.

Parameters

pos Index of first character of substring.

n1 Number of characters in substring.

s C string to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the *n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *s*. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

5.90.2.41 `size_type std::basic_string<_CharT, _Traits, _Allocator>::copy (_CharT * __s, size_type __n, size_type __pos = 0) const`
[*inherited*]

Copy substring into C string.

Parameters

s C string to copy value into.

n Number of characters to copy.

pos Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

[*std::out_of_range*](#) If *pos* > *size()*.

Copies up to *n* characters starting at *pos* into the C string *s*. If *pos* is greater than *size()*, *out_of_range* is thrown.

5.90.2.42 `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::crbegin() const [inline, inherited]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 690 of file `basic_string.h`.

5.90.2.43 `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::crend() const [inline, inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 699 of file `basic_string.h`.

5.90.2.44 `const _CharT* std::basic_string<_CharT, _Traits, _Allocator>::data() const [inline, inherited]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1774 of file `basic_string.h`.

5.90.2.45 `bool std::basic_string<_CharT, _Traits, _Allocator>::empty() const [inline, inherited]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 801 of file `basic_string.h`.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1073

5.90.2.46 `const_iterator std::basic_string<_CharT, _Traits, _Allocator>::end () const [inline, inherited]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 628 of file `basic_string.h`.

5.90.2.47 `iterator std::basic_string<_CharT, _Traits, _Allocator>::end () [inline, inherited]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 617 of file `basic_string.h`.

5.90.2.48 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::erase (size_type __pos = 0, size_type __n = npos) [inline, inherited]`

Remove characters.

Parameters

pos Index of first character to remove (default 0).

n Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Removes *n* characters from this string starting at *pos*. The length of the string is reduced by *n*. If there are $< n$ characters to remove, the remainder of the string is truncated. If *p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1345 of file `basic_string.h`.

5.90.2.49 `iterator std::basic_string<_CharT, _Traits, _Allocator>::erase (iterator __position) [inline, inherited]`

Remove one character.

Parameters

position Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1361 of file basic_string.h.

5.90.2.50 `iterator std::basic_string<_CharT,_Traits,_Allocator>::erase (iterator __first, iterator __last) [inherited]`

Remove a range of characters.

Parameters

first Iterator referencing the first character to remove.

last Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

5.90.2.51 `size_type std::basic_string<_CharT,_Traits,_Allocator>::find (const _CharT * __s, size_type __pos, size_type __n) const [inherited]`

Find position of a C substring.

Parameters

s C string to locate.

pos Index of character to search from.

n Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns npos.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1075

5.90.2.52 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = 0) const [inline, inherited]`

Find position of a string.

Parameters

str String to locate.

pos Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1809 of file `basic_string.h`.

5.90.2.53 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (_CharT __c, size_type __pos = 0) const [inherited]`

Find position of a character.

Parameters

c Character to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.90.2.54 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find (const _CharT* __s, size_type __pos = 0) const [inline, inherited]`

Find position of a C string.

Parameters

s C string to locate.

pos Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1823 of file `basic_string.h`.

5.90.2.55 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_first_not_of (const basic_string< _CharT, _Traits,
_Allocator > & __str, size_type __pos = 0) const [inline,
inherited]`

Find position of a character not in string.

Parameters

str String containing characters to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2033 of file `basic_string.h`.

5.90.2.56 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_first_not_of (const _CharT * __s, size_type __pos,
size_type __n) const [inherited]`

Find position of a character not in C substring.

Parameters

s C string containing characters to avoid.

pos Index of character to search from.

n Number of characters from *s* to consider.

Returns

Index of first occurrence.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1077

Starting from *pos*, searches forward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.90.2.57 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of(_CharT __c, size_type __pos = 0) const`
`[inherited]`

Find position of a different character.

Parameters

c Character to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.90.2.58 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of(const _CharT * __s, size_type __pos = 0) const`
`[inline, inherited]`

Find position of a character not in C string.

Parameters

s C string containing characters to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2062 of file `basic_string.h`.

5.90.2.59 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of(_CharT __c, size_type __pos = 0) const`
`[inline, inherited]`

Find position of a character.

Parameters

c Character to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for the character *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `find(c, pos)`.

Definition at line 1958 of file `basic_string.h`.

5.90.2.60 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_first_of (const _CharT * __s, size_type __pos = 0) const
[inline, inherited]`

Find position of a character of C string.

Parameters

s String containing characters to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1939 of file `basic_string.h`.

5.90.2.61 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_first_of (const basic_string< _CharT, _Traits, _Allocator >
& __str, size_type __pos = 0) const [inline, inherited]`

Find position of a character of string.

Parameters

str String containing characters to locate.

pos Index of character to search from (default 0).

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1079

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1911 of file `basic_string.h`.

5.90.2.62 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_of(const _CharT * __s, size_type __pos, size_type __n) const` **[inherited]**

Find position of a character of C substring.

Parameters

s String containing characters to locate.

pos Index of character to search from.

n Number of characters from *s* to search for.

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.90.2.63 `size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_not_of(const _CharT * __s, size_type __pos, size_type __n) const` **[inherited]**

Find last position of a character not in C substring.

Parameters

s C string containing characters to avoid.

pos Index of character to search back from.

n Number of characters from *s* to consider.

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.90.2.64 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_not_of (_CharT __c, size_type __pos = npos) const
[inherited]`

Find last position of a different character.

Parameters

c Character to avoid.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.90.2.65 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_not_of (const _CharT * __s, size_type __pos = npos)
const [inline, inherited]`

Find last position of a character not in C string.

Parameters

s C string containing characters to avoid.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2121 of file basic_string.h.

5.90.2.66 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_not_of (const basic_string< _CharT, _Traits,
_Allocator > & __str, size_type __pos = npos) const [inline,
inherited]`

Find last position of a character not in string.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1081

Parameters

str String containing characters to avoid.
pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2092 of file `basic_string.h`.

```
5.90.2.67 size_type std::basic_string<_CharT, _Traits, _Allocator  
>::find_last_of ( const basic_string<_CharT, _Traits, _Allocator >  
& __str, size_type __pos = npostr ) const [inline, inherited]
```

Find last position of a character of string.

Parameters

str String containing characters to locate.
pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1972 of file `basic_string.h`.

```
5.90.2.68 size_type std::basic_string<_CharT, _Traits, _Allocator  
>::find_last_of ( const _CharT * __s, size_type __pos = npostr )  
const [inline, inherited]
```

Find last position of a character of C string.

Parameters

s C string containing characters to locate.
pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2000 of file `basic_string.h`.

5.90.2.69 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_of (_CharT __c, size_type __pos = npos) const
[inline, inherited]`

Find last position of a character.

Parameters

c Character to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `rfind(c, pos)`.

Definition at line 2019 of file `basic_string.h`.

5.90.2.70 `size_type std::basic_string< _CharT, _Traits, _Allocator
>::find_last_of (const _CharT* __s, size_type __pos, size_type
__n) const [inherited]`

Find last position of a character of C substring.

Parameters

s C string containing characters to locate.

pos Index of character to search back from.

n Number of characters from *s* to search for.

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1083

5.90.2.71 `reference std::basic_string<_CharT, _Traits, _Allocator>::front () [inline, inherited]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 867 of file `basic_string.h`.

5.90.2.72 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::front () const [inline, inherited]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 875 of file `basic_string.h`.

5.90.2.73 `allocator_type std::basic_string<_CharT, _Traits, _Allocator>::get_allocator () const [inline, inherited]`

Return copy of allocator used to construct this string.

Definition at line 1781 of file `basic_string.h`.

5.90.2.74 `void std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, size_type __n, _CharT __c) [inline, inherited]`

Insert multiple characters.

Parameters

p Iterator referencing location in string to insert at.

n Number of characters to insert

c The character to insert.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Inserts *n* copies of character *c* starting at the position referenced by iterator *p*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1172 of file `basic_string.h`.

5.90.2.75 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos1, const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos2, size_type __n)` **[inline, inherited]**

Insert a substring.

Parameters

pos1 Iterator referencing location in string to insert at.

str The string to insert.

pos2 Start of characters in *str* to insert.

n Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.

std::out_of_range If *pos1* > `size()` or *pos2* > *str.size()*.

Starting at *pos1*, insert *n* character of *str* beginning with *pos2*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos1* is beyond the end of this string or *pos2* is beyond the end of *str*, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1240 of file `basic_string.h`.

5.90.2.76 `void std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, initializer_list<_CharT> __l)` **[inline, inherited]**

Insert an `initializer_list` of characters.

Parameters

p Iterator referencing location in string to insert at.

l The `initializer_list` of characters to insert.

Exceptions

std::length_error If new length exceeds `max_size()`.

Definition at line 1199 of file `basic_string.h`.

5.90.2.77 `iterator std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, _CharT __c) [inline, inherited]`

Insert one character.

Parameters

- p* Iterator referencing position in string to insert at.
- c* The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error If new length exceeds `max_size()`.

Inserts character *c* at position referenced by *p*. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If *p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1321 of file `basic_string.h`.

5.90.2.78 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos1, const basic_string<_CharT, _Traits, _Allocator> & __str) [inline, inherited]`

Insert value of a string.

Parameters

- pos1* Iterator referencing location in string to insert at.
- str* The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.

Inserts value of *str* starting at *pos1*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1218 of file `basic_string.h`.

5.90.2.79 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert(size_type __pos, const _CharT* __s) [inline, inherited]`

Insert a C string.

Parameters

pos Iterator referencing location in string to insert at.

s The C string to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1281 of file `basic_string.h`.

5.90.2.80 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert(size_type __pos, size_type __n, _CharT __c) [inline, inherited]`

Insert multiple characters.

Parameters

pos Index in string to insert at.

n Number of characters to insert

c The character to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1087

Inserts n copies of character c starting at index pos . If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If $pos > \text{length}()$, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1304 of file `basic_string.h`.

5.90.2.81 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (size_type __pos, const _CharT* __s, size_type __n)`
[`inherited`]

Insert a C substring.

Parameters

pos Iterator referencing location in string to insert at.

s The C string to insert.

n The number of characters to insert.

Returns

Reference to this string.

Exceptions

[`std::length_error`](#) If new length exceeds `max_size()`.

[`std::out_of_range`](#) If pos is beyond the end of this string.

Inserts the first n characters of s starting at pos . If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If pos is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

5.90.2.82 `void std::basic_string<_CharT, _Traits, _Allocator>::insert (iterator __p, _InputIterator __beg, _InputIterator __end)`
[`inline`, `inherited`]

Insert a range of characters.

Parameters

p Iterator referencing location in string to insert at.

beg Start of range.

end End of range.

Exceptions

[`std::length_error`](#) If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1188 of file basic_string.h.

5.90.2.83 `size_type std::basic_string<_CharT,_Traits,_Allocator>::length () const [inline, inherited]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 714 of file basic_string.h.

5.90.2.84 `size_type std::basic_string<_CharT,_Traits,_Allocator>::max_size () const [inline, inherited]`

Returns the size() of the largest possible string.

Definition at line 719 of file basic_string.h.

5.90.2.85 `basic_string& std::basic_string<_CharT,_Traits,_Allocator>::operator+=(const basic_string<_CharT,_Traits,_Allocator> & __str) [inline, inherited]`

Append a string to this string.

Parameters

str The string to append.

Returns

Reference to this string.

Definition at line 922 of file basic_string.h.

5.90.2.86 `basic_string& std::basic_string<_CharT,_Traits,_Allocator>::operator+=(const _CharT * __s) [inline, inherited]`

Append a C string.

Parameters

s The C string to append.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1089

Returns

Reference to this string.

Definition at line 931 of file `basic_string.h`.

5.90.2.87 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=(_CharT __c) [inline, inherited]`

Append a character.

Parameters

c The character to append.

Returns

Reference to this string.

Definition at line 940 of file `basic_string.h`.

5.90.2.88 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::operator+=(initializer_list<_CharT> __l) [inline, inherited]`

Append an `initializer_list` of characters.

Parameters

l The `initializer_list` of characters to be appended.

Returns

Reference to this string.

Definition at line 953 of file `basic_string.h`.

5.90.2.89 `const_reference std::basic_string<_CharT, _Traits, _Allocator>::operator[](size_type __pos) const [inline, inherited]`

Subscript access to the data contained in the string.

Parameters

pos The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 816 of file `basic_string.h`.

5.90.2.90 `reference std::basic_string<_CharT, _Traits, _Allocator>::operator[] (size_type __pos) [inline, inherited]`

Subscript access to the data contained in the string.

Parameters

pos The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 833 of file `basic_string.h`.

5.90.2.91 `void std::basic_string<_CharT, _Traits, _Allocator>::push_back (_CharT __c) [inline, inherited]`

Append a single character.

Parameters

c Character to append.

Definition at line 1041 of file `basic_string.h`.

5.90.2.92 `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::rbegin () const [inline, inherited]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 646 of file `basic_string.h`.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1091

5.90.2.93 `reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::rbegin()` `[inline, inherited]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 637 of file `basic_string.h`.

5.90.2.94 `const_reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::rend() const` `[inline, inherited]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 664 of file `basic_string.h`.

5.90.2.95 `reverse_iterator std::basic_string<_CharT, _Traits, _Allocator>::rend()` `[inline, inherited]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 655 of file `basic_string.h`.

5.90.2.96 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace(iterator __i1, iterator __i2, const _CharT * __s)` `[inline, inherited]`

Replace range of characters with C string.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1545 of file basic_string.h.

5.90.2.97 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, const _CharT * __s, size_type __n) [inline, inherited]`

Replace range of characters with C substring.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.
- n* Number of characters from *s* to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the first *n* characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1524 of file basic_string.h.

5.90.2.98 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2) [inline, inherited]`

Replace range of characters with range.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- k1* Iterator referencing start of range to insert.
- k2* Iterator referencing end of range to insert.

Returns

Reference to this string.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1093

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1589 of file `basic_string.h`.

5.90.2.99 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, size_type __n, _CharT __c) [inline, inherited]`

Replace range of characters with multiple characters.

Parameters

i1 Iterator referencing start of range to replace.

i2 Iterator referencing end of range to replace.

n Number of characters to insert.

c Character to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, *n* copies of *c* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1566 of file `basic_string.h`.

5.90.2.100 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (size_type __pos, size_type __n, const basic_string<_CharT, _Traits, _Allocator> & __str) [inline, inherited]`

Replace characters with value from another string.

Parameters

pos Index of first character to replace.

n Number of characters to be replaced.

str String to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*pos*,*pos*+*n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1400 of file `basic_string.h`.

5.90.2.101 `basic_string& std::basic_string<_CharT,_Traits,_Allocator>::replace (size_type __pos1, size_type __n1, const basic_string<_CharT,_Traits,_Allocator> & __str, size_type __pos2, size_type __n2) [inline, inherited]`

Replace characters with value from another string.

Parameters

pos1 Index of first character to replace.

n1 Number of characters to be replaced.

str String to insert.

pos2 Index of first character of *str* to use.

n2 Number of characters from *str* to use.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos1* > `size()` or *pos2* > `str.size()`.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*pos1*,*pos1* + *n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1422 of file `basic_string.h`.

5.90.2.102 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (iterator __i1, iterator __i2, const basic_string<_CharT, _Traits, _Allocator> & __str) [inline, inherited]`

Replace range of characters with string.

Parameters

i1 Iterator referencing start of range to replace.

i2 Iterator referencing end of range to replace.

str String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, the value of *str* is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1506 of file `basic_string.h`.

5.90.2.103 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (size_type __pos, size_type __n1, const _CharT* __s) [inline, inherited]`

Replace characters with value of a C string.

Parameters

pos Index of first character to replace.

n1 Number of characters to be replaced.

s C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos* > `size()`.

std::length_error If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of *s* are inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1465 of file `basic_string.h`.

5.90.2.104 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace(size_type __pos, size_type __n1, size_type __n2, _CharT __c) [inline, inherited]`

Replace characters with multiple characters.

Parameters

pos Index of first character to replace.
n1 Number of characters to be replaced.
n2 Number of characters to insert.
c Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos* > `size()`.
std::length_error If new length exceeds `max_size()`.

Removes the characters in the range `[pos, pos + n1)` from this string. In place, *n2* copies of *c* are inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1488 of file `basic_string.h`.

5.90.2.105 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace(iterator __i1, iterator __i2, initializer_list<_CharT> __l) [inline, inherited]`

Replace range of characters with `initializer_list`.

Parameters

i1 Iterator referencing start of range to replace.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1097

i2 Iterator referencing end of range to replace.

l The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1657 of file `basic_string.h`.

5.90.2.106 `basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) [inherited]`

Replace characters with value of a C substring.

Parameters

pos Index of first character to replace.

n1 Number of characters to be replaced.

s C string to insert.

n2 Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos1* > `size()`.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range `[pos,pos + n1)` from this string. In place, the first *n2* characters of *s* are inserted, or all of *s* if *n2* is too large. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

5.90.2.107 `void std::basic_string<_CharT, _Traits, _Allocator>::reserve (size_type __res_arg = 0) [inherited]`

Attempt to preallocate enough memory for specified number of characters.

Parameters

res_arg Number of characters required.

Exceptions

std::length_error If *res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

5.90.2.108 `void std::basic_string<_CharT, _Traits, _Allocator>::resize (size_type __n, _CharT __c) [inherited]`

Resizes the string to the specified number of characters.

Parameters

n Number of characters the string should contain.

c Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *c*.

5.90.2.109 `void std::basic_string<_CharT, _Traits, _Allocator>::resize (size_type __n) [inline, inherited]`

Resizes the string to the specified number of characters.

Parameters

n Number of characters the string should contain.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1099

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 746 of file `basic_string.h`.

5.90.2.110 `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (const basic_string<_CharT, _Traits, _Allocator> & __str, size_type __pos = npos) const [inline, inherited]`

Find last position of a string.

Parameters

str String to locate.

pos Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for value of *str* within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1853 of file `basic_string.h`.

5.90.2.111 `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (_CharT __c, size_type __pos = npos) const [inherited]`

Find last position of a character.

Parameters

c Character to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns `npos`.

5.90.2.112 `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (const _CharT * __s, size_type __pos = npos) const [inline, inherited]`

Find last position of a C string.

Parameters

s C string to locate.

pos Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1881 of file `basic_string.h`.

5.90.2.113 `size_type std::basic_string<_CharT, _Traits, _Allocator>::rfind (const _CharT * __s, size_type __pos, size_type __n) const [inherited]`

Find last position of a C substring.

Parameters

s C string to locate.

pos Index of character to search back from.

n Number of characters from *s* to search for.

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

5.90.2.114 `void std::basic_string<_CharT, _Traits, _Allocator>::shrink_to_fit () [inline, inherited]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 752 of file `basic_string.h`.

5.90 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference 1101

5.90.2.115 `size_type std::basic_string<_CharT, _Traits, _Allocator>::size ()`
`const [inline, inherited]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 708 of file `basic_string.h`.

5.90.2.116 `basic_string std::basic_string<_CharT, _Traits, _Allocator>::substr (size_type __pos = 0, size_type __n = npos) const`
`[inline, inherited]`

Get a substring.

Parameters

pos Index of first character (default 0).

n Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

[*std::out_of_range*](#) If *pos* > `size()`.

Construct and return a new string using the *n* characters starting at *pos*. If the string is too short, use the remainder of the characters. If *pos* is beyond the end of the string, `out_of_range` is thrown.

Definition at line 2153 of file `basic_string.h`.

5.90.2.117 `void std::basic_string<_CharT, _Traits, _Allocator>::swap`
`(basic_string<_CharT, _Traits, _Allocator> & __s)`
`[inherited]`

Swap contents with another string.

Parameters

s String to swap with.

Exchanges the contents of this string with that of *s* in constant time.

5.90.3 Member Data Documentation

5.90.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` `[inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.90.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.90.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` `[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

5.90.3.4 `const size_type std::basic_string<_CharT, _Traits, _Allocator>::npos` `[static, inherited]`

Value returned by various member functions when they fail.

Definition at line 278 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [debug/string](#)

5.91 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference

General reduction, using a binary operator.

Public Member Functions

- `__accumulate_binop_reduct` (`_BinOp` &__b)
- `template<typename _Result, typename _Addend> _Result operator()` (`const _Result` &__x, `const _Addend` &__y)

Public Attributes

- `_BinOp` & `__binop`

5.91.1 Detailed Description

`template<typename _BinOp> struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>`

General reduction, using a binary operator.

Definition at line 335 of file `for_each_selectors.h`.

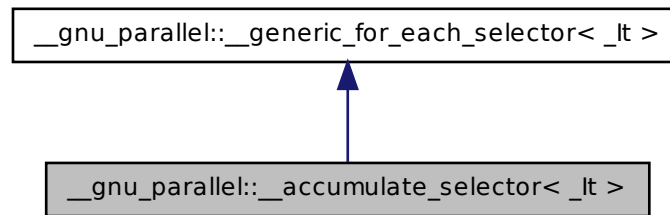
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.92 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

[std::accumulate\(\)](#) selector.

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

Public Attributes

- `_It M_finish_iterator`

5.92.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__accumulate_selector<_It>`

[std::accumulate\(\)](#) selector.

Definition at line 208 of file `for_each_selectors.h`.

5.92.2 Member Function Documentation

5.92.2.1 `template<typename _It> template<typename _Op> std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector<_It>::operator() (_Op __o, _It __i) [inline]`

Functor execution.

5.93 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference 1105

Parameters

- `__o` Operator (unused).
- `__i` iterator referencing object.

Returns

The current value.

Definition at line 216 of file `for_each_selectors.h`.

5.92.3 Member Data Documentation

5.92.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

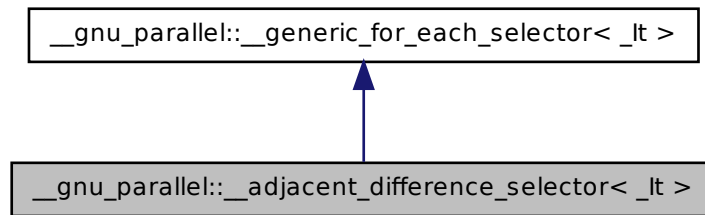
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.93 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference

Selector that returns the difference between two adjacent `__elements`.

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It` [`_M_finish_iterator`](#)

5.93.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__adjacent_difference_selector<_It>`

Selector that returns the difference between two adjacent `__elements`.

Definition at line 269 of file `for_each_selectors.h`.

5.93.2 Member Data Documentation

5.93.2.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator`
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

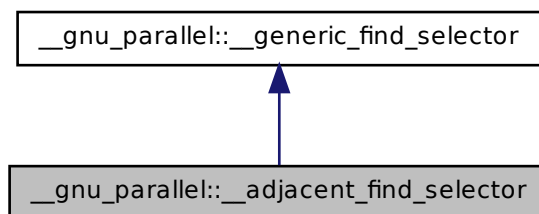
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.94 `__gnu_parallel::__adjacent_find_selector` Struct Reference

Test predicate on two adjacent elements.

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __-`
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

5.94.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

5.94.2 Member Function Documentation

5.94.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__adjacent_find_selector::M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

`__begin1` Begin iterator of first sequence.
`__end1` End iterator of first sequence.
`__begin2` Begin iterator of second sequence.
`__pred` Find predicate.

Definition at line 105 of file `find_selectors.h`.

References `std::make_pair()`.

5.94.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__adjacent_find_selector::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

`__i1` Iterator on first sequence.
`__i2` Iterator on second sequence (unused).
`__pred` Find predicate.

Definition at line 90 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

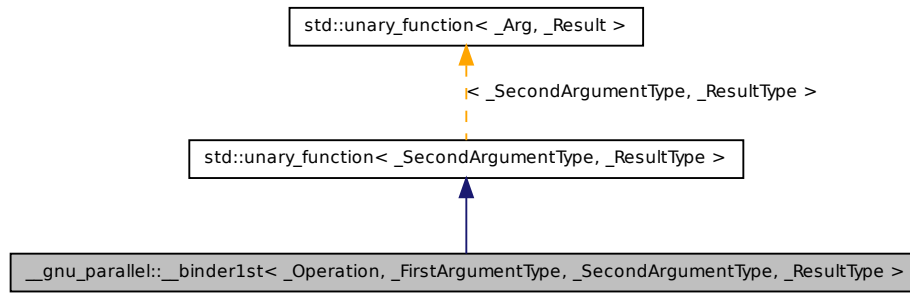
- [find_selectors.h](#)

5.95 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Similar to `std::binder1st`, but giving the argument types explicitly.

5.95 `__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>` Class Template Reference 1109

Inheritance diagram for `__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`:



Public Types

- typedef `_SecondArgumentType` [argument_type](#)
- typedef `_ResultType` [result_type](#)

Public Member Functions

- `__binder1st` (`const _Operation &__x, const _FirstArgumentType &__y`)
- `_ResultType operator()` (`_SecondArgumentType &__x`) `const`
- `_ResultType operator()` (`const _SecondArgumentType &__x`)

Protected Attributes

- `_Operation _M_op`
- `_FirstArgumentType _M_value`

5.95.1 Detailed Description

template<typename `_Operation`, typename `_FirstArgumentType`, typename `_SecondArgumentType`, typename `_ResultType`> class `__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`

Similar to `std::binder1st`, but giving the argument types explicitly.

Definition at line 192 of file parallel/base.h.

5.95.2 Member Typedef Documentation

5.95.2.1 `typedef _SecondArgumentType std::unary_function<
_SecondArgumentType , _ResultType >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.95.2.2 `typedef _ResultType std::unary_function< _SecondArgumentType ,
_ResultType >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file stl_function.h.

The documentation for this class was generated from the following file:

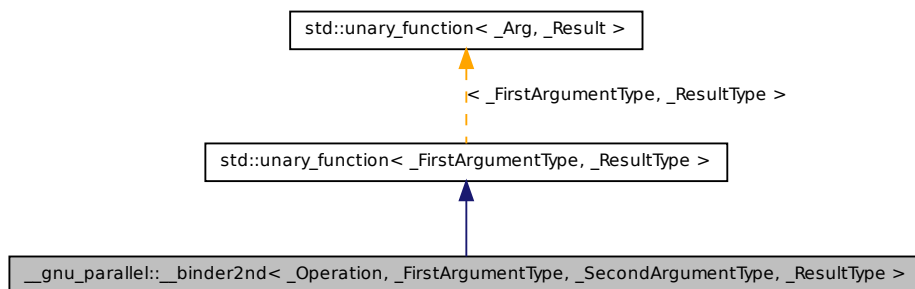
- [parallel/base.h](#)

5.96 `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Similar to [std::binder2nd](#), but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__binder2nd< _Operation, _`

`FirstArgumentType, _SecondArgumentType, _ResultType >:`



Public Types

- typedef `_FirstArgumentType` [argument_type](#)
- typedef `_ResultType` [result_type](#)

Public Member Functions

- `__binder2nd` (`const _Operation &__x, const _SecondArgumentType &__y`)
- `_ResultType operator()` (`_FirstArgumentType &__x`)
- `_ResultType operator()` (`const _FirstArgumentType &__x`) `const`

Protected Attributes

- `_Operation _M_op`
- `_SecondArgumentType _M_value`

5.96.1 Detailed Description

`template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType> class __gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>`

Similar to [std::binder2nd](#), but giving the argument types explicitly.

Definition at line 220 of file `parallel/base.h`.

5.96.2 Member Typedef Documentation

5.96.2.1 `typedef _FirstArgumentType std::unary_function<
_FirstArgumentType , _ResultType >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.96.2.2 `typedef _ResultType std::unary_function< _FirstArgumentType ,
_ResultType >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

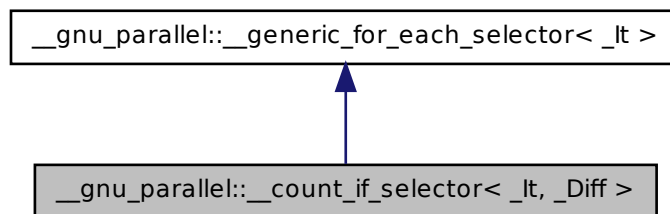
The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.97 __gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference

`std::count_if()` selector.

Inheritance diagram for `__gnu_parallel::__count_if_selector< _It, _Diff >`:



Public Member Functions

- `template<typename _Op>
_Diff operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.97.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_if_selector<_It, _Diff>`

`std::count_if()` selector.

Definition at line 194 of file `for_each_selectors.h`.

5.97.2 Member Function Documentation

5.97.2.1 `template<typename _It, typename _Diff> template<typename _Op
> _Diff __gnu_parallel::__count_if_selector<_It, _Diff>::operator()
(_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

5.97.3 Member Data Documentation

5.97.3.1 `template<typename _It> _It __gnu_parallel::__
generic_for_each_selector<_It>::_M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

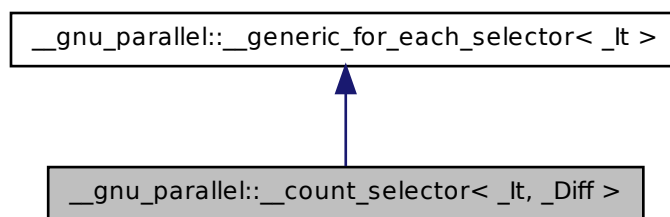
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.98 `__gnu_parallel::__count_selector< _It, _Diff >` Struct Template Reference

`std::count()` selector.

Inheritance diagram for `__gnu_parallel::__count_selector< _It, _Diff >`:



Public Member Functions

- `template<typename _ValueType >`
`_Diff operator\(\) (_ValueType &__v, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.98 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference

5.98.1 Detailed Description

`template<typename _It, typename _Diff> struct __gnu_parallel::__count_selector<_It, _Diff>`

`std::count()` selector.

Definition at line 180 of file `for_each_selectors.h`.

5.98.2 Member Function Documentation

5.98.2.1 `template<typename _It, typename _Diff> template<typename _ValueType > _Diff __gnu_parallel::__count_selector<_It, _Diff>::operator() (_ValueType & __v, _It __i) [inline]`

Functor execution.

Parameters

`__v` Current value.

`__i` iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 188 of file `for_each_selectors.h`.

5.98.3 Member Data Documentation

5.98.3.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

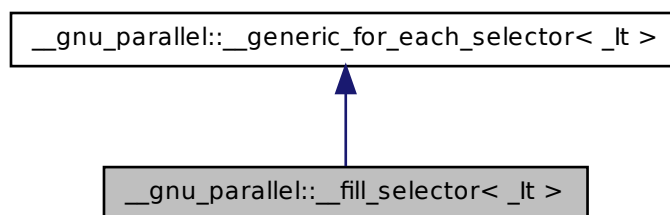
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.99 `__gnu_parallel::__fill_selector< _It >` Struct Template Reference

`std::fill()` selector.

Inheritance diagram for `__gnu_parallel::__fill_selector< _It >`:



Public Member Functions

- `template<typename _ValueType > bool operator() (_ValueType &__v, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.99.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__fill_selector< _It >`

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

5.99.2 Member Function Documentation

5.99.2.1 `template<typename _It> template<typename _ValueType> bool
__gnu_parallel::__fill_selector<_It>::operator() (_ValueType &
__v, _It __i) [inline]`

Functor execution.

Parameters

`__v` Current value.

`__i` iterator referencing object.

Definition at line 91 of file `for_each_selectors.h`.

5.99.3 Member Data Documentation

5.99.3.1 `template<typename _It> _It __gnu_parallel::__-
generic_for_each_selector<_It>::__M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

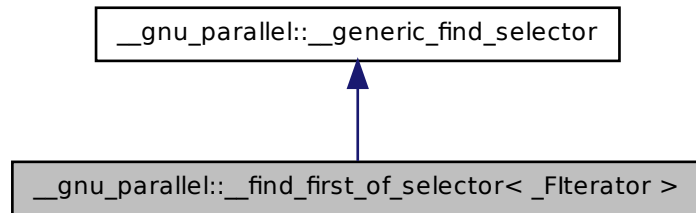
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.100 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference

Test predicate on several elements.

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



Public Member Functions

- **`__find_first_of_selector`** (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`std::pair<_RAIter1, _RAIter2> _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

Public Attributes

- `_FIterator _M_begin`
- `_FIterator _M_end`

5.100.1 Detailed Description

`template<typename _FIterator> struct __gnu_parallel::__find_first_of_selector<_FIterator>`

Test predicate on several elements.

Definition at line 153 of file `find_selectors.h`.

5.100.2 Member Function Documentation

5.100.2.1 `template<typename _FIterator > template<typename _RAIter1 ,
typename _RAIter2 , typename _Pred > std::pair<_RAIter1,
_RAIter2> __gnu_parallel::__find_first_of_selector< _FIterator
>::__M_sequential_algorithm (_RAIter1 __begin1, _RAIter1
__end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

__begin1 Begin iterator of first sequence.
__end1 End iterator of first sequence.
__begin2 Begin iterator of second sequence.
__pred Find predicate.

Definition at line 186 of file `find_selectors.h`.

References `std::make_pair()`.

5.100.2.2 `template<typename _FIterator > template<typename
_RAIter1 , typename _RAIter2 , typename _Pred > bool
__gnu_parallel::__find_first_of_selector< _FIterator >::operator() (
_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

__i1 Iterator on first sequence.
__i2 Iterator on second sequence (unused).
__pred Find predicate.

Definition at line 169 of file `find_selectors.h`.

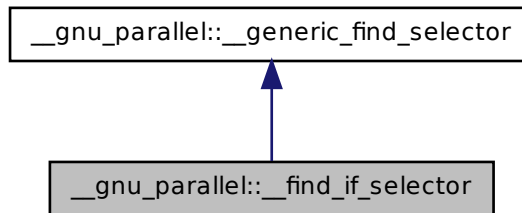
The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

5.101 `__gnu_parallel::__find_if_selector` Struct Reference

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Inheritance diagram for `__gnu_parallel::__find_if_selector`:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __-`
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

5.101.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file `find_selectors.h`.

5.101.2 Member Function Documentation

5.101.2.1 `template<typename _RAIter1, typename _RAIter2`
`, typename _Pred > std::pair<_RAIter1, _RAIter2>`
`__gnu_parallel::__find_if_selector::M_sequential_algorithm (`
`_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred`
`__pred) \[inline\]`

Corresponding sequential algorithm on a sequence.

Parameters

[__begin1](#) Begin iterator of first sequence.

5.102 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference 121

`__end1` End iterator of first sequence.

`__begin2` Begin iterator of second sequence.

`__pred` Find predicate.

Definition at line 72 of file `find_selectors.h`.

References `std::make_pair()`.

```
5.101.2.2  template<typename _RAIter1 , typename _RAIter2 , typename
            _Pred > bool __gnu_parallel::__find_if_selector::operator() (
            _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]
```

Test on one position.

Parameters

`__i1` _Iterator on first sequence.

`__i2` _Iterator on second sequence (unused).

`__pred` Find predicate.

Definition at line 60 of file `find_selectors.h`.

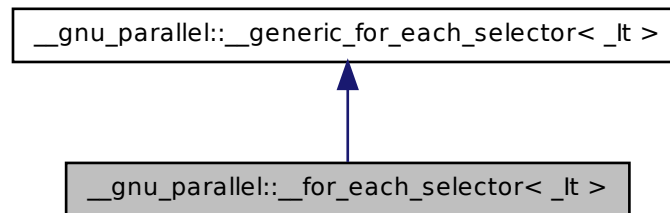
The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

5.102 `__gnu_parallel::__for_each_selector<_It>` Struct Template Reference

`std::for_each()` selector.

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It` `_M_finish_iterator`

5.102.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__for_each_selector<_It>`

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.

5.102.2 Member Function Documentation

5.102.2.1 `template<typename _It> template<typename _Op> bool`
`__gnu_parallel::__for_each_selector<_It>::operator() (_Op &`
`__o, _It __i) [inline]`

Functor execution.

5.103 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference 123

Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

5.102.3 Member Data Documentation

5.102.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

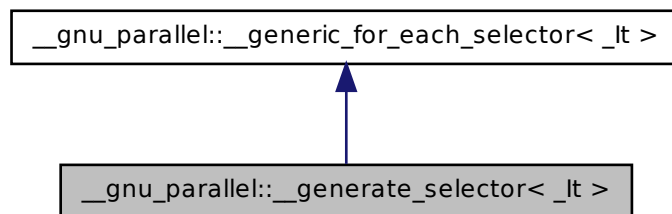
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.103 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

`std::generate()` selector.

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



Public Member Functions

- `template<typename _Op >`
`bool operator\(\) (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.103.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__generate_selector< _It >`

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

5.103.2 Member Function Documentation

5.103.2.1 `template<typename _It> template<typename _Op > bool`
`__gnu_parallel::__generate_selector< _It >::operator() (_Op &`
`__o, _It __i) [inline]`

Functor execution.

Parameters

- `__o` Operator.
- `__i` iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

5.103.3 Member Data Documentation

5.103.3.1 `template<typename _It > _It __gnu_parallel::__`
`generic_for_each_selector< _It >::__M_finish_iterator`
`\[inherited\]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

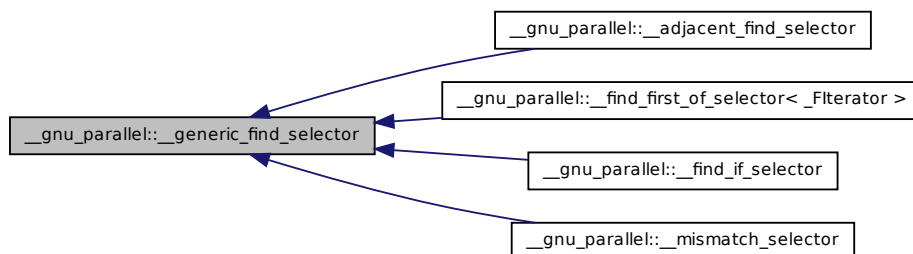
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.104 __gnu_parallel::__generic_find_selector Struct Reference

Base class of all [__gnu_parallel::__find_template](#) selectors.

Inheritance diagram for __gnu_parallel::__generic_find_selector:



5.104.1 Detailed Description

Base class of all [__gnu_parallel::__find_template](#) selectors.

Definition at line 43 of file [find_selectors.h](#).

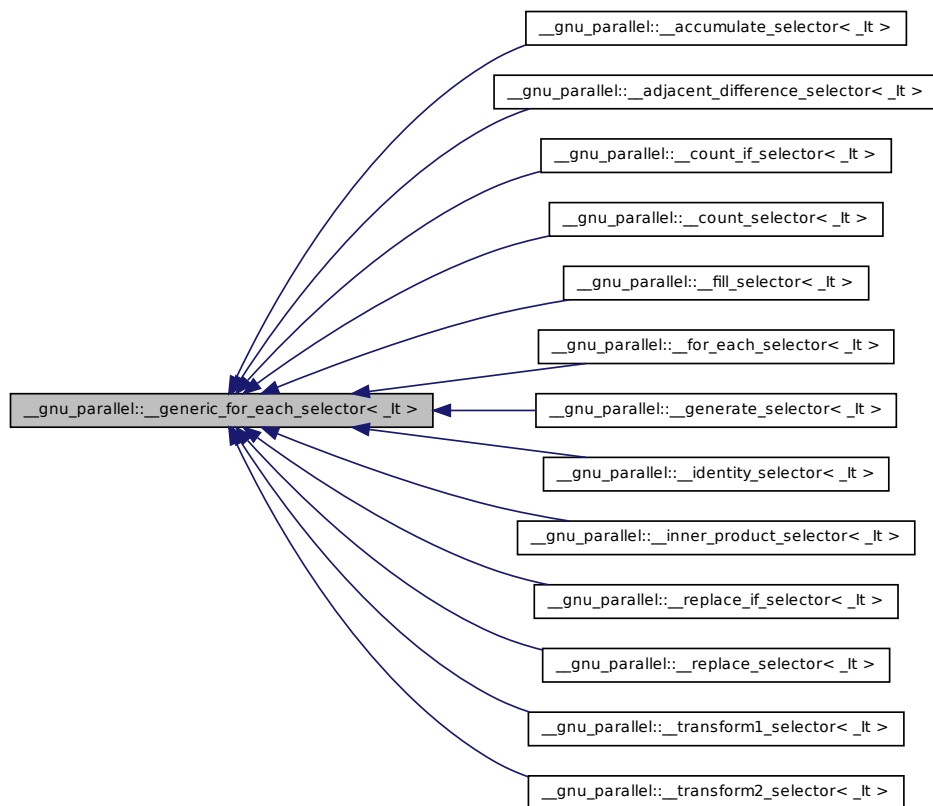
The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

5.105 __gnu_parallel::__generic_for_each_selector<_It > Struct Template Reference

Generic __selector for embarrassingly parallel functions.

Inheritance diagram for `__gnu_parallel::__generic_for_each_selector<_It>`:



Public Attributes

- `_It` [_M_finish_iterator](#)

5.105.1 Detailed Description

```
template<typename _It> struct __gnu_parallel::__generic_for_each_selector<
    _It>
```

Generic `__selector` for embarrassingly parallel functions.

5.106 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference 1127

Definition at line 42 of file `for_each_selectors.h`.

5.105.2 Member Data Documentation

5.105.2.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator`

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

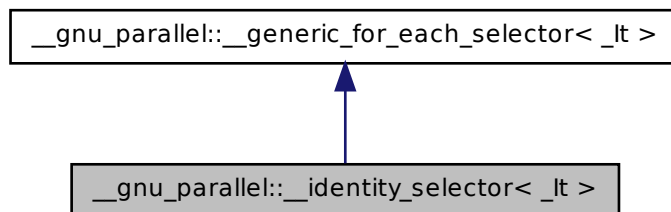
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.106 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

Selector that just returns the passed iterator.

Inheritance diagram for `__gnu_parallel::__identity_selector<_It>`:



Public Member Functions

- `template<typename _Op> _It operator() (_Op __o, _It __i)`

Public Attributes

- [_It _M_finish_iterator](#)

5.106.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__identity_selector< _It >`

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

5.106.2 Member Function Documentation

5.106.2.1 `template<typename _It> template<typename _Op > _It
__gnu_parallel::__identity_selector< _It >::operator() (_Op __o,
_It __i) [inline]`

Functor execution.

Parameters

- `__o` Operator (unused).
- `__i` iterator referencing object.

Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.

5.106.3 Member Data Documentation

5.106.3.1 `template<typename _It > _It __gnu_parallel::__-
generic_for_each_selector< _It >::__M_finish_iterator
[inherited]`

_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

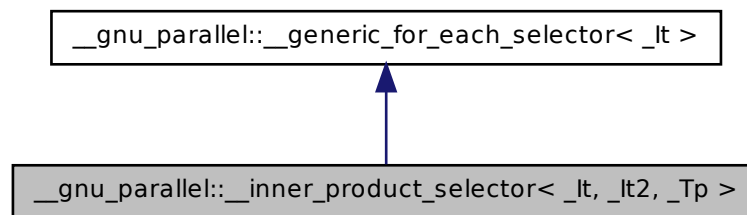
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.107 `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct` Template Reference

[std::inner_product\(\)](#) selector.

Inheritance diagram for `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`:



Public Member Functions

- [__inner_product_selector](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op >`
`_Tp operator\(\) (_Op __mult, _It __current)`

Public Attributes

- `_It __begin1_iterator`
- `_It2 __begin2_iterator`
- `_It __M_finish_iterator`

5.107.1 Detailed Description

`template<typename _It, typename _It2, typename _Tp> struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`

[std::inner_product\(\)](#) selector.

Definition at line 222 of file `for_each_selectors.h`.

5.107.2 Constructor & Destructor Documentation

5.107.2.1 `template<typename _It , typename _It2 , typename _Tp >
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp
>::__inner_product_selector (_It __b1, _It2 __b2) [inline,
explicit]`

Constructor.

Parameters

- b1* Begin iterator of first sequence.
- b2* Begin iterator of second sequence.

Definition at line 234 of file `for_each_selectors.h`.

5.107.3 Member Function Documentation

5.107.3.1 `template<typename _It , typename _It2 , typename _Tp >
template<typename _Op > _Tp __gnu_parallel::__inner_product_
selector< _It, _It2, _Tp >::operator() (_Op __mult, _It __current)
[inline]`

Functor execution.

Parameters

- __mult* Multiplication functor.
- __current* iterator referencing object.

Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

References `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_` - iterator, and `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_` - iterator.

5.107.4 Member Data Documentation

5.107.4.1 `template<typename _It , typename _It2 , typename _Tp >
_It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp
>::__begin1_iterator`

Begin iterator of first sequence.

5.108 `__gnu_parallel::__max_element_reduct<_Compare, _It> Struct Template Reference` 1131

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()`.

5.107.4.2 `template<typename _It, typename _It2, typename _Tp>
_It2 __gnu_parallel::__inner_product_selector<_It, _It2, _Tp
>::begin2_iterator`

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()`.

5.107.4.3 `template<typename _It> _It __gnu_parallel::__-
generic_for_each_selector<_It>::M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.108 `__gnu_parallel::__max_element_reduct<_Compare, _It> Struct Template Reference` -

Reduction for finding the maximum element, using a comparator.

Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

Public Attributes

- `_Compare & __comp`

5.108.1 Detailed Description

template<typename _Compare, typename _It> struct __gnu_parallel::__max_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.109 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference

Reduction for finding the maximum element, using a comparator.

Public Member Functions

- `__min_element_reduct (_Compare &__c)`
- `_It operator() (_It __x, _It __y)`

Public Attributes

- `_Compare & __comp`

5.109.1 Detailed Description

template<typename _Compare, typename _It> struct __gnu_parallel::__min_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.

Definition at line 307 of file `for_each_selectors.h`.

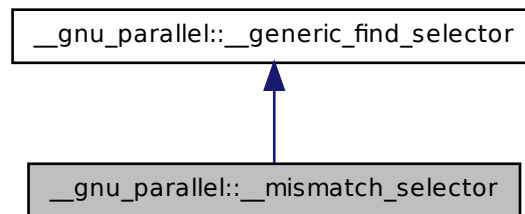
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.110 __gnu_parallel::__mismatch_selector Struct Reference

Test inverted predicate on a single element.

Inheritance diagram for __gnu_parallel::__mismatch_selector:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __-`
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

5.110.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

5.110.2 Member Function Documentation

5.110.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

`__begin1` Begin iterator of first sequence.
`__end1` End iterator of first sequence.
`__begin2` Begin iterator of second sequence.
`__pred` Find predicate.

Definition at line 143 of file `find_selectors.h`.

5.110.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__mismatch_selector::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

`__i1` Iterator on first sequence.
`__i2` Iterator on second sequence (unused).
`__pred` Find predicate.

Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

5.111 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for 3-way merging with `__sentinels` turned off.

5.112 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference` 1135

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

5.111.1 Detailed Description

`template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned off. Note that 3-way merging is always stable!

Definition at line 748 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.112 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference`

Switch for 3-way merging with `__sentinels` turned on.

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

5.112.1 Detailed Description

`template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 3-way merging with `__sentinels` turned on. Note that 3-way merging is always stable!

Definition at line 768 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.113 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for 4-way merging with `__sentinels` turned off.

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

5.113.1 Detailed Description

```
template<bool __sentinels, typename _RAIterIterator, typename _RAIter3,
typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__
multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _
RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned off. Note that 4-way merging is always stable!

Definition at line 791 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.114 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct` Template Reference

Switch for 4-way merging with `__sentinels` turned on.

5.115 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch<__sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare>` 1137

Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

5.114.1 Detailed Description

`template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

Switch for 4-way merging with `__sentinels` turned on. Note that 4-way merging is always stable!

Definition at line 811 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.115 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare>` Struct Template Reference

Switch for k-way merging with `__sentinels` turned on.

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

5.115.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterIterator, type-
name _RAIter3, typename _DifferenceTp, typename _Compare> struct __gnu_
parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable,
_RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with __sentinels turned on.

Definition at line 833 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.116 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

Switch for k-way merging with __sentinels turned off.

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

5.116.1 Detailed Description

```
template<bool __stable, typename _RAIterIterator, typename _RAIter3,
typename _DifferenceTp, typename _Compare> struct __gnu_parallel::__-
multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _-
RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with __sentinels turned off.

Definition at line 868 of file multiway_merge.h.

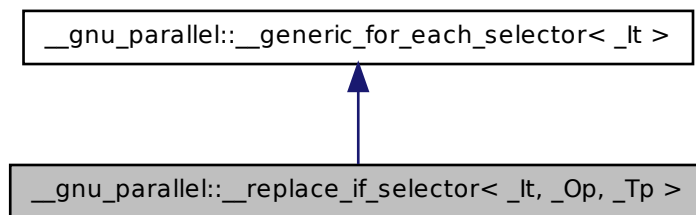
The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.117 `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>` Struct Template Reference

`std::replace()` selector.

Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`:



Public Member Functions

- `__replace_if_selector` (`const _Tp &__new_val`)
- `bool operator()` (`_Op &__o, _It __i`)

Public Attributes

- `const _Tp & __new_val`
- `_It _M_finish_iterator`

5.117.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp> struct __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>
```

`std::replace()` selector.

Definition at line 156 of file `for_each_selectors.h`.

5.117.2 Constructor & Destructor Documentation

5.117.2.1 `template<typename _It , typename _Op , typename _Tp
> __gnu_parallel::__replace_if_selector< _It, _Op, _Tp
>::__replace_if_selector(const _Tp & __new_val) [inline,
explicit]`

Constructor.

Parameters

`__new_val` Value to replace with.

Definition at line 164 of file `for_each_selectors.h`.

5.117.3 Member Function Documentation

5.117.3.1 `template<typename _It , typename _Op , typename _Tp > bool
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (
_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

`__o` Operator.

`__i` iterator referencing object.

Definition at line 170 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val`.

5.117.4 Member Data Documentation

5.117.4.1 `template<typename _It , typename _Op , typename _Tp > const
_Tp& __gnu_parallel::__replace_if_selector< _It, _Op, _Tp
>::__new_val`

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator()()`.

5.118 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

5.117.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator`
[`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

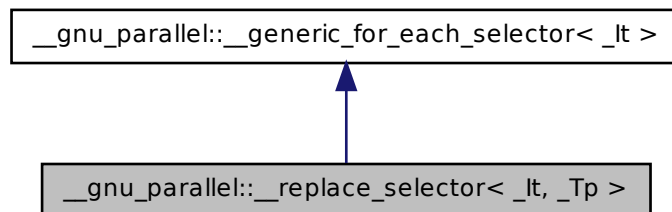
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.118 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

`std::replace()` selector.

Inheritance diagram for `__gnu_parallel::__replace_selector<_It, _Tp>`:



Public Member Functions

- [__replace_selector](#) (const `_Tp` & [__new_val](#))
- bool [operator\(\)](#) (`_Tp` & `__v`, `_It` `__i`)

Public Attributes

- const `_Tp` & [__new_val](#)
- `_It` [_M_finish_iterator](#)

5.118.1 Detailed Description

```
template<typename _It, typename _Tp> struct __gnu_parallel::__replace_
selector< _It, _Tp >
```

std::replace() selector.

Definition at line 132 of file for_each_selectors.h.

5.118.2 Constructor & Destructor Documentation

5.118.2.1 `template<typename _It, typename _Tp > __gnu_parallel::__replace_selector< _It, _Tp >::__replace_selector (const _Tp & __new_val) [inline, explicit]`

Constructor.

Parameters

`__new_val` Value to replace with.

Definition at line 140 of file for_each_selectors.h.

5.118.3 Member Function Documentation

5.118.3.1 `template<typename _It, typename _Tp > bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (_Tp & __v, _It __i) [inline]`

Functor execution.

Parameters

`__v` Current value.

`__i` iterator referencing object.

Definition at line 146 of file for_each_selectors.h.

References `__gnu_parallel::__replace_selector< _It, _Tp >::__new_val`.

5.118.4 Member Data Documentation

5.118.4.1 `template<typename _It, typename _Tp > const _Tp& __gnu_parallel::__replace_selector< _It, _Tp >::__new_val`

Value to replace with.

5.119 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

Definition at line 135 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_selector<_It, _Tp>::operator()`.

5.118.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator` [`inherited`]

`_Iterator` on last element processed; needed for some algorithms (e.g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

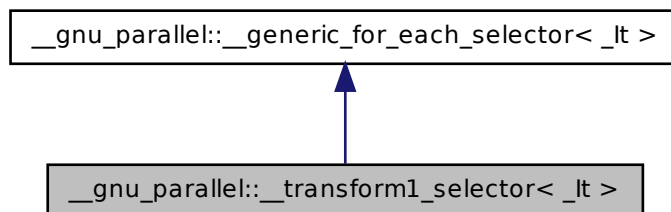
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.119 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

`std::transform()` `__selector`, one input sequence variant.

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- [_It _M_finish_iterator](#)

5.119.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform1_selector< _It >`

`std::transform()` __selector, one input sequence variant.

Definition at line 100 of file `for_each_selectors.h`.

5.119.2 Member Function Documentation

5.119.2.1 `template<typename _It> template<typename _Op > bool
__gnu_parallel::__transform1_selector< _It >::operator() (_Op &
__o, _It __i) [inline]`

Functor execution.

Parameters

__o Operator.

__i iterator referencing object.

Definition at line 107 of file `for_each_selectors.h`.

5.119.3 Member Data Documentation

5.119.3.1 `template<typename _It > _It __gnu_parallel::__
generic_for_each_selector< _It >::__M_finish_iterator
[inherited]`

__Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

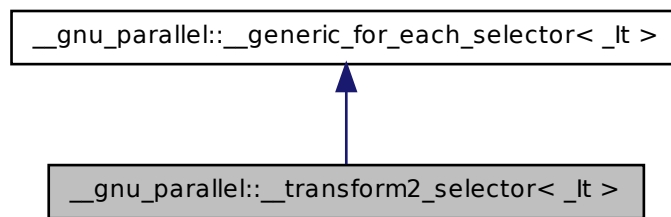
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.120 `__gnu_parallel::__transform2_selector<_It>` Struct Template Reference

`std::transform()` `__selector`, two input sequences variant.

Inheritance diagram for `__gnu_parallel::__transform2_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

5.120.1 Detailed Description

`template<typename _It> struct __gnu_parallel::__transform2_selector<_It>`

`std::transform()` `__selector`, two input sequences variant.

Definition at line 116 of file `for_each_selectors.h`.

5.120.2 Member Function Documentation

5.120.2.1 `template<typename _It> template<typename _Op > bool
__gnu_parallel::__transform2_selector<_It >::operator() (_Op &
__o, _It __i) [inline]`

Functor execution.

Parameters

`__o` Operator.

`__i` iterator referencing object.

Definition at line 123 of file `for_each_selectors.h`.

5.120.3 Member Data Documentation

5.120.3.1 `template<typename _It > _It __gnu_parallel::__
generic_for_each_selector<_It >::__M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

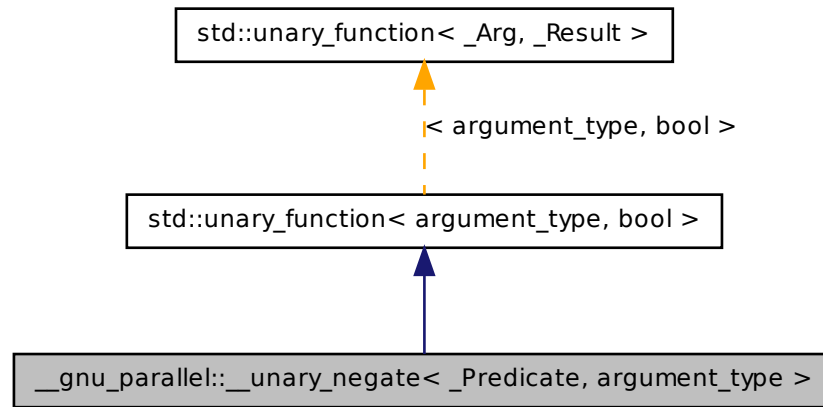
- [for_each_selectors.h](#)

5.121 `__gnu_parallel::__unary_negate< _Predicate, argument_type >` Class Template Reference

Similar to [std::unary_negate](#), but giving the argument types explicitly.

Inheritance diagram for `__gnu_parallel::__unary_negate< _Predicate, argument_type`

>:



Public Types

- typedef `argument_type` `argument_type`
- typedef `bool` `result_type`

Public Member Functions

- `__unary_negate` (`const _Predicate &__x`)
- `bool operator()` (`const argument_type &__x`)

Protected Attributes

- `_Predicate _M_pred`

5.121.1 Detailed Description

`template<typename _Predicate, typename argument_type> class __gnu_parallel::__unary_negate<_Predicate, argument_type>`

Similar to `std::unary_negate`, but giving the argument types explicitly.

Definition at line 173 of file parallel/base.h.

5.121.2 Member Typedef Documentation

5.121.2.1 `typedef argument_type std::unary_function< argument_type , bool >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.121.2.2 `typedef bool std::unary_function< argument_type , bool >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file stl_function.h.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.122 `__gnu_parallel::__DRandomShufflingGlobalData< _RAIter >` Struct Template Reference

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::iterator_traits< _RAIter > _TraitsType`
- `typedef _TraitsType::value_type _ValueType`

Public Member Functions

- `_DRandomShufflingGlobalData (_RAIter &__source)`

Public Attributes

- `_ThreadIndex * _M_bin_proc`

5.122 `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>` Struct Template Reference 1149

- `_DifferenceType** _M_dist`
- `int _M_num_bins`
- `int _M_num_bits`
- `_RAIter & _M_source`
- `_DifferenceType* _M_starts`
- `_ValueType** _M_temporaries`

5.122.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>`

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 52 of file `random_shuffle.h`.

5.122.2 Constructor & Destructor Documentation

5.122.2.1 `template<typename _RAIter> __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__DRandomShufflingGlobalData (_RAIter & __source)`
[`inline`]

Constructor.

Definition at line 83 of file `random_shuffle.h`.

5.122.3 Member Data Documentation

5.122.3.1 `template<typename _RAIter> _ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::__M_bin_proc`

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

5.122.3.2 `template<typename _RAIter> _DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist`

Two-dimensional array to hold the thread-bin distribution.

Dimensions (`_M_num_threads + 1`) `__x` (`_M_num_bins + 1`).

Definition at line 67 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.122.3.3 `template<typename _RAIter> int __gnu_ parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins`

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.122.3.4 `template<typename _RAIter> int __gnu_ parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits`

Number of bits needed to address the bins.

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.122.3.5 `template<typename _RAIter> _RAIter& __gnu_ parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source`

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

5.123 `__gnu_parallel::__DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference 1151

5.122.3.6 `template<typename _RAIter> _DifferenceType*
__gnu_parallel::__DRandomShufflingGlobalData<_RAIter
>::__M_starts`

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.122.3.7 `template<typename _RAIter> _ValueType**
__gnu_parallel::__DRandomShufflingGlobalData<_RAIter
>::__M_temporaries`

Temporary arrays for each thread.

Definition at line 62 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

The documentation for this struct was generated from the following file:

- [random_shuffle.h](#)

5.123 `__gnu_parallel::__DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference

Local data for a thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).

Public Attributes

- [_BinIndex __bins_end](#)
- [_BinIndex _M_bins_begin](#)
- [int _M_num_threads](#)
- [_DRandomShufflingGlobalData<_RAIter> * _M_sd](#)
- [uint32_t _M_seed](#)

5.123.1 Detailed Description

template<typename _RAIter, typename _RandomNumberGenerator> struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >

Local data for a thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).

Definition at line 91 of file random_shuffle.h.

5.123.2 Member Data Documentation

5.123.2.1 template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end

End index for bins taken care of by this thread.

Definition at line 100 of file random_shuffle.h.

Referenced by [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#).

5.123.2.2 template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__M_bins_begin

Begin index for bins taken care of by this thread.

Definition at line 97 of file random_shuffle.h.

Referenced by [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#).

5.123.2.3 template<typename _RAIter, typename _RandomNumberGenerator> int __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__M_num_threads

Number of threads participating in total.

Definition at line 94 of file random_shuffle.h.

Referenced by [__gnu_parallel::__parallel_random_shuffle_drs\(\)](#), and [__gnu_parallel::__parallel_random_shuffle_drs_pu\(\)](#).

5.123.2.4 `template<typename _RAIter, typename
_RandomNumberGenerator> _DRandomShufflingGlobalData<_
_RAIter>* __gnu_parallel::_DRSSorterPU< _RAIter,
_RandomNumberGenerator >::_M_sd`

Pointer to global data.

Definition at line 106 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

5.123.2.5 `template<typename _RAIter, typename
_RandomNumberGenerator> uint32_t __gnu_parallel::_
DRSSorterPU< _RAIter, _RandomNumberGenerator
>::_M_seed`

Random `_M_seed` for this thread.

Definition at line 103 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs_pu()`.

The documentation for this struct was generated from the following file:

- [random_shuffle.h](#)

5.124 `__gnu_parallel::_DummyReduct` Struct Reference

Reduction function doing nothing.

Public Member Functions

- `bool operator() (bool, bool) const`

5.124.1 Detailed Description

Reduction function doing nothing.

Definition at line 298 of file `for_each_selectors.h`.

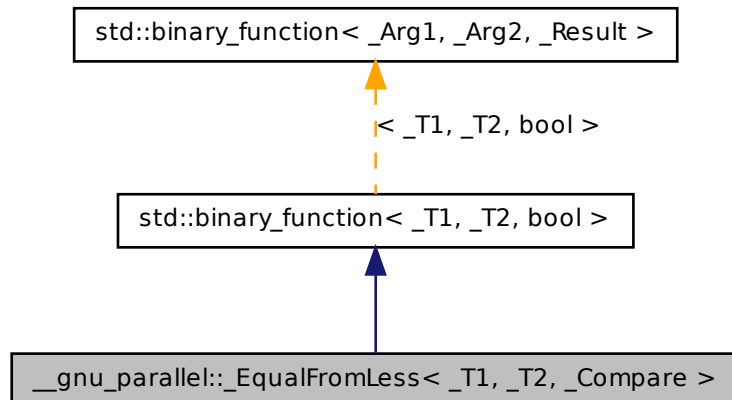
The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.125 `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >` Class Template Reference

Constructs predicate for equality from strict weak ordering predicate.

Inheritance diagram for `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >`:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_T2` [second_argument_type](#)

Public Member Functions

- `_EqualFromLess` (`_Compare &__comp`)
- `bool operator()` (`const _T1 &__a, const _T2 &__b`)

5.125.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >
```

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file `parallel/base.h`.

5.125.2 Member Typedef Documentation

5.125.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.125.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.125.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

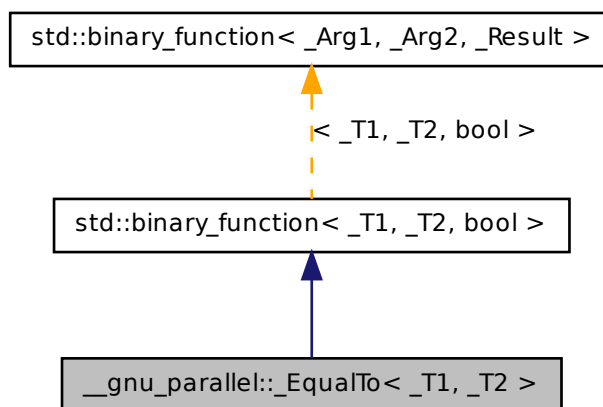
The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.126 `__gnu_parallel::_EqualTo< _T1, _T2 >` Struct Template Reference

Similar to `std::equal_to`, but allows two different types.

Inheritance diagram for `__gnu_parallel::_EqualTo<_T1, _T2>`:



Public Types

- `typedef _T1` [first_argument_type](#)
- `typedef bool` [result_type](#)
- `typedef _T2` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`

5.126.1 Detailed Description

```
template<typename _T1, typename _T2> struct __gnu_parallel::_EqualTo< _-
_T1, _T2 >
```

Similar to [std::equal_to](#), but allows two different types.

Definition at line 244 of file `parallel/base.h`.

5.126.2 Member Typedef Documentation

5.126.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.126.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.126.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

5.127 __gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

Public Member Functions

- [_GuardedIterator](#) (_RAIter __begin, _RAIter __end, _Compare &__comp)
- [operator _RAIter](#) ()
- `std::iterator_traits< _RAIter >::value_type & operator* ()`
- [_GuardedIterator](#)< _RAIter, _Compare > & [operator++](#) ()

Friends

- `bool operator< (_GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)`
- `bool operator<= (_GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)`

5.127.1 Detailed Description

`template<typename _RAIter, typename _Compare> class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >`

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons. The implicit supremum comes with a performance cost.

Deriving from _RAIter is not possible since _RAIter need not be a class.

Definition at line 66 of file multiway_merge.h.

5.127.2 Constructor & Destructor Documentation

5.127.2.1 `template<typename _RAIter, typename _Compare > __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::_GuardedIterator (_RAIter __begin, _RAIter __end, _Compare & __comp) [inline]`

Constructor. Sets iterator to beginning of sequence.

Parameters

- `__begin` Begin iterator of sequence.
- `__end` End iterator of sequence.
- `__comp` Comparator provided for associated overloaded compare operators.

Definition at line 84 of file multiway_merge.h.

5.127.3 Member Function Documentation

5.127.3.1 `template<typename _RAIter, typename _Compare > __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator _RAIter () [inline]`

Convert to wrapped iterator.

5.127 `__gnu_parallel::_GuardedIterator<_RAIter, _Compare>` Class Template Reference 1159

Returns

Wrapped iterator.

Definition at line 105 of file `multiway_merge.h`.

```
5.127.3.2  template<typename _RAIter , typename _Compare
            > std::iterator_traits<_RAIter>::value_type&
            __gnu_parallel::_GuardedIterator< _RAIter, _Compare
            >::operator*( ) [inline]
```

Dereference operator.

Returns

Referenced element.

Definition at line 100 of file `multiway_merge.h`.

```
5.127.3.3  template<typename _RAIter , typename _Compare
            > _GuardedIterator<_RAIter, _Compare>&
            __gnu_parallel::_GuardedIterator< _RAIter, _Compare
            >::operator++( ) [inline]
```

Pre-increment operator.

Returns

This.

Definition at line 91 of file `multiway_merge.h`.

5.127.4 Friends And Related Function Documentation

```
5.127.4.1  template<typename _RAIter , typename _Compare > bool
            operator< ( _GuardedIterator< _RAIter, _Compare > & __bi1,
            _GuardedIterator< _RAIter, _Compare > & __bi2 ) [friend]
```

Compare two elements referenced by guarded iterators.

Parameters

`__bi1` First iterator.

`__bi2` Second iterator.

Returns

`true` if less.

Definition at line 113 of file `multiway_merge.h`.

5.127.4.2 `template<typename _RAIter, typename _Compare> bool
operator<= (_GuardedIterator< _RAIter, _Compare> & __bi1,
_GuardedIterator< _RAIter, _Compare> & __bi2) [friend]`

Compare two elements referenced by guarded iterators.

Parameters

`__bi1` First iterator.

`__bi2` Second iterator.

Returns

`True` if less equal.

Definition at line 128 of file `multiway_merge.h`.

The documentation for this class was generated from the following file:

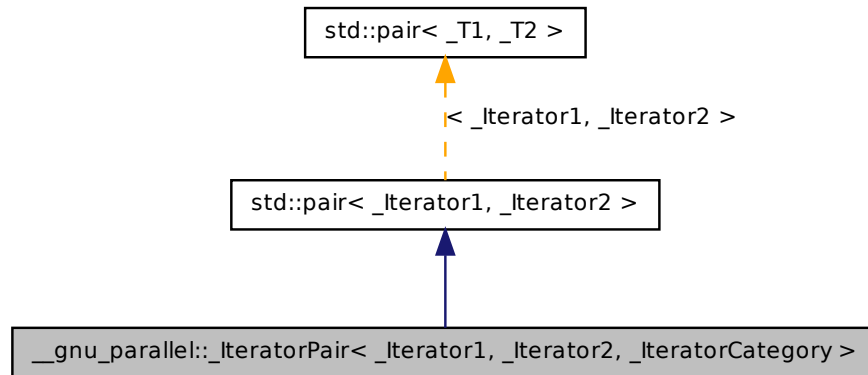
- [multiway_merge.h](#)

5.128 `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory>` Class Template Reference

A pair of iterators. The usual iterator operations are applied to both child iterators.

Inheritance diagram for `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _`

IteratorCategory >:



Public Types

- typedef `std::iterator_traits<_Iterator1>` **_TraitsType**
- typedef `_TraitsType::difference_type` **difference_type**
- typedef `_Iterator1` **first_type**
- typedef `_IteratorCategory` **iterator_category**
- typedef `_IteratorPair` * **pointer**
- typedef `_IteratorPair` & **reference**
- typedef `_Iterator2` **second_type**
- typedef void **value_type**

Public Member Functions

- **_IteratorPair** (`const _Iterator1 &__first, const _Iterator2 &__second`)
- **operator _Iterator2** () const
- `_IteratorPair` **operator+** (`difference_type __delta`) const
- const `_IteratorPair` **operator++** (int)
- `_IteratorPair` & **operator++** ()
- `difference_type` **operator-** (const `_IteratorPair` &__other) const
- const `_IteratorPair` **operator--** (int)
- `_IteratorPair` & **operator--** ()

- `_IteratorPair` & `operator=` (const `_IteratorPair` &__other)
- void `swap` (pair &__p)

Public Attributes

- `_Iterator1` `first`
- `_Iterator2` `second`

5.128.1 Detailed Description

```
template<typename _Iterator1,    typename _Iterator2,    typename _-
IteratorCategory> class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2,
_IteratorCategory >
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file `iterator.h`.

5.128.2 Member Typedef Documentation

5.128.2.1 `typedef _Iterator1 std::pair< _Iterator1 , _Iterator2 >::first_type`
[`inherited`]

`first_type` is the first bound type

Definition at line 86 of file `stl_pair.h`.

5.128.2.2 `typedef _Iterator2 std::pair< _Iterator1 , _Iterator2 >::second_type`
[`inherited`]

`second_type` is the second bound type

Definition at line 87 of file `stl_pair.h`.

5.128.3 Member Data Documentation

5.128.3.1 `_Iterator1 std::pair< _Iterator1 , _Iterator2 >::first` [`inherited`]

`first` is a copy of the first object

Definition at line 89 of file `stl_pair.h`.

5.128.3.2 `_Iterator2 std::pair<_Iterator1, _Iterator2>::second` [inherited]

`second` is a copy of the second object

Definition at line 90 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

5.129 `__gnu_parallel::IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Public Types

- typedef `std::iterator_traits<_Iterator1>::difference_type` **difference_type**
- typedef `_IteratorCategory` **iterator_category**
- typedef `_IteratorTriple` * **pointer**
- typedef `_IteratorTriple` & **reference**
- typedef void **value_type**

Public Member Functions

- `_IteratorTriple` (`const _Iterator1 &__first, const _Iterator2 &__second, const _Iterator3 &__third`)
- `operator _Iterator3` () const
- `_IteratorTriple operator+` (`difference_type __delta`) const
- `const _IteratorTriple operator++` (int)
- `_IteratorTriple & operator++` ()
- `difference_type operator-` (`const _IteratorTriple &__other`) const
- `_IteratorTriple & operator--` ()
- `const _IteratorTriple operator--` (int)
- `_IteratorTriple & operator=` (`const _IteratorTriple &__other`)

Public Attributes

- `_Iterator1 _M_first`
- `_Iterator2 _M_second`
- `_Iterator3 _M_third`

5.129.1 Detailed Description

`template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory> class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >`

A triple of iterators. The usual iterator operations are applied to all three child iterators. Definition at line 120 of file `iterator.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

5.130 `__gnu_parallel::_Job< _DifferenceTp > Struct` Template Reference

One `__job` for a certain thread.

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Attributes

- `volatile _DifferenceType _M_first`
- `volatile _DifferenceType _M_last`
- `volatile _DifferenceType _M_load`

5.130.1 Detailed Description

`template<typename _DifferenceTp> struct __gnu_parallel::_Job< _DifferenceTp >`

One `__job` for a certain thread.

Definition at line 54 of file `workstealing.h`.

5.130.2 Member Data Documentation

5.130.2.1 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job<_DifferenceTp>::_M_first`

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

5.130.2.2 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job<_DifferenceTp>::_M_last`

Last element.

Changed by owning thread only.

Definition at line 67 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

5.130.2.3 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job<_DifferenceTp>::_M_load`

Number of elements, i.e. `_M_last - _M_first + 1`.

Changed by owning thread only.

Definition at line 72 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

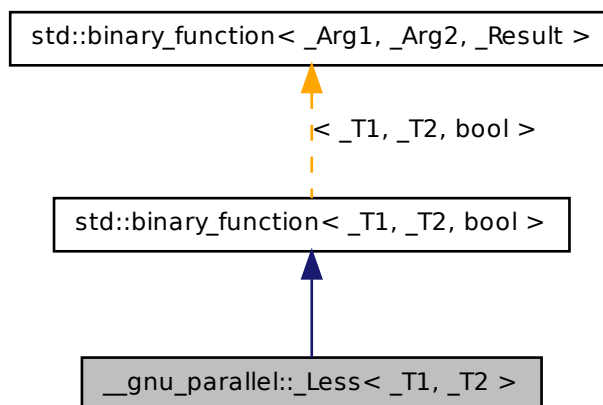
The documentation for this struct was generated from the following file:

- [workstealing.h](#)

5.131 `__gnu_parallel::_Less<_T1, _T2>` Struct Template Reference

Similar to [std::less](#), but allows two different types.

Inheritance diagram for `__gnu_parallel::_Less<_T1, _T2>`:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_T2` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`
- `bool operator() (const _T2 &__t2, const _T1 &__t1) const`

5.131.1 Detailed Description

`template<typename _T1, typename _T2> struct __gnu_parallel::_Less<_T1, _T2>`

Similar to [std::less](#), but allows two different types.

Definition at line 252 of file `parallel/base.h`.

5.131.2 Member Typedef Documentation

5.131.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.131.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.131.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

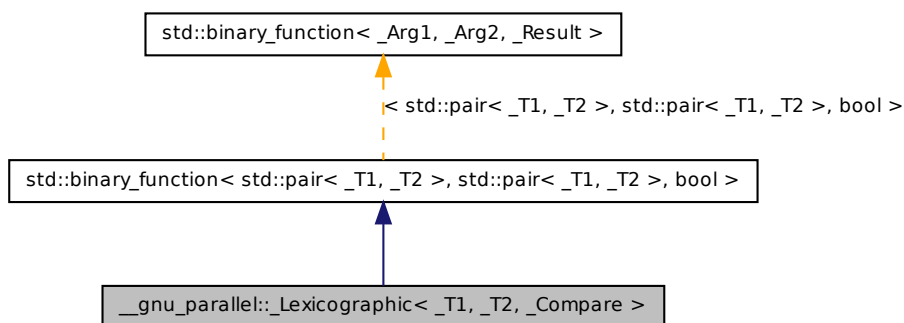
The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

5.132 __gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference

Compare __a pair of types lexicographically, ascending.

Inheritance diagram for `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`:



Public Types

- typedef `std::pair<_T1, _T2>` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair<_T1, _T2>` `second_argument_type`

Public Member Functions

- `_Lexicographic` (`_Compare` &`__comp`)
- `bool operator()` (`const std::pair<_T1, _T2>` &`__p1`, `const std::pair<_T1, _T2>` &`__p2`) `const`

5.132.1 Detailed Description

`template<typename _T1, typename _T2, typename _Compare> class __gnu_parallel::_Lexicographic<_T1, _T2, _Compare>`

Compare `__`a pair of types lexicographically, ascending.

Definition at line 55 of file `multiseq_selection.h`.

5.132.2 Member Typedef Documentation

5.132.2.1 `typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::first_argument_type`
[*inherited*]

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.132.2.2 `typedef bool std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::result_type` [*inherited*]

type of the return type

Definition at line 118 of file stl_function.h.

5.132.2.3 `typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::second_argument_type`
[*inherited*]

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this class was generated from the following file:

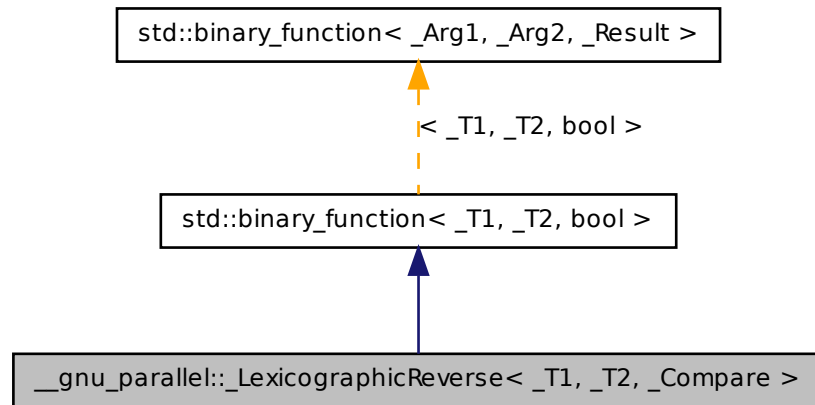
- [multiseq_selection.h](#)

5.133 __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare > Class Template Reference

Compare __a pair of types lexicographically, descending.

Inheritance diagram for __gnu_parallel::_LexicographicReverse< _T1, _T2, _-

Compare >:



Public Types

- typedef [_T1](#) [first_argument_type](#)
- typedef bool [result_type](#)
- typedef [_T2](#) [second_argument_type](#)

Public Member Functions

- [_LexicographicReverse](#) ([_Compare](#) &[__comp](#))
- bool **operator()** (const [std::pair](#)< [_T1](#), [_T2](#) > &[__p1](#), const [std::pair](#)< [_T1](#), [_T2](#) > &[__p2](#)) const

5.133.1 Detailed Description

template<typename [_T1](#), typename [_T2](#), typename [_Compare](#)> class [__gnu_parallel::_LexicographicReverse](#)< [_T1](#), [_T2](#), [_Compare](#) >

Compare [__](#)a pair of types lexicographically, descending.

Definition at line 82 of file [multiseq_selection.h](#).

5.133.2 Member Typedef Documentation

5.133.2.1 `typedef _T1 std::binary_function< _T1 , _T2 , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.133.2.2 `typedef bool std::binary_function< _T1 , _T2 , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.133.2.3 `typedef _T2 std::binary_function< _T1 , _T2 , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

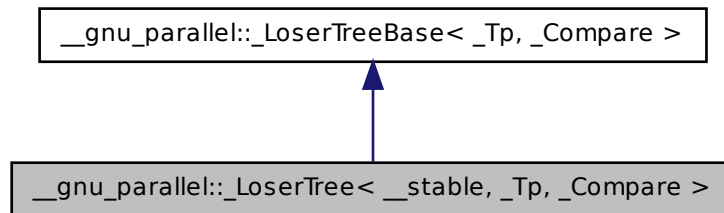
The documentation for this class was generated from the following file:

- [multiseq_selection.h](#)

5.134 __gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference

Stable [_LoserTree](#) variant.

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTree` (unsigned int __k, _Compare __comp)
- void `__delete_min_insert` (_Tp __key, bool __sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` * `_M_losers`
- unsigned int `_M_offset`

5.134.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare> class `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`

Stable `_LoserTree` variant. Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

5.134 __gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference 1173

Unstable variant is done using partial specialisation below.

Definition at line 165 of file losertree.h.

5.134.2 Member Function Documentation

5.134.2.1 `template<bool __stable, typename _Tp, typename _Compare
> void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare
>::__delete_min_insert (_Tp __key, bool __sup) [inline]`

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 217 of file losertree.h.

References `std::swap()`.

5.134.2.2 `template<typename _Tp, typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::__get_min_source () [inline, inherited]`

Returns

the index of the sequence with the smallest element.

Definition at line 151 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

5.134.2.3 `template<typename _Tp, typename _Compare > void
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start
(const _Tp & __key, int __source, bool __sup) [inline,
inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

`__key` the element to insert

`__source` __index of the __source __sequence

`__sup` flag that determines whether the value to insert is an explicit __supremum.

Definition at line 130 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

5.134.3 Member Data Documentation

5.134.3.1 `template<typename _Tp, typename _Compare> _Compare
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp
[protected, inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

5.134.3.2 `template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_M_first_insert [protected, inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.134.3.3 `template<typename _Tp, typename _Compare> unsigned int
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k
[protected, inherited]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.134.3.4 `template<typename _Tp, typename _Compare> _Loser*
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers
[protected, inherited]`

`_LoserTree` __elements.

5.135 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference 1175

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

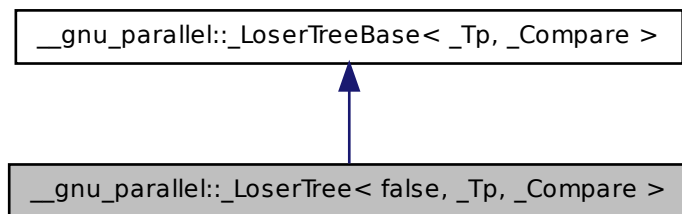
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.135 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Unstable [_LoserTree](#) variant.

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



Public Member Functions

- `_LoserTree` (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

Protected Attributes

- [_Compare](#) [_M_comp](#)
- [bool](#) [_M_first_insert](#)
- [unsigned int](#) [_M_ik](#)
- [unsigned int](#) [_M_k](#)
- [unsigned int](#) [_M_log_k](#)
- [_Loser](#) * [_M_losers](#)
- [unsigned int](#) [_M_offset](#)

5.135.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTree< false, _Tp, _Compare >`

Unstable [_LoserTree](#) variant. Stability (non-stable here) is selected with partial specialization.

Definition at line 255 of file losertree.h.

5.135.2 Member Function Documentation

5.135.2.1 `template<typename _Tp , typename _Compare > void
__gnu_parallel::_LoserTree< false, _Tp, _Compare
>::__delete_min_insert (_Tp __key, bool __sup) [inline]`

Delete the `_M_key` smallest element and insert the element `__key` instead.

Parameters

`__key` the `_M_key` to insert

`__sup` true iff `__key` is an explicitly marked supremum

Definition at line 317 of file losertree.h.

References `std::swap()`.

5.135.2.2 `template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::__get_min_source () [inline, inherited]`

Returns

the index of the sequence with the smallest element.

5.135 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference 1177

Definition at line 151 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

5.135.2.3 `template<typename _Tp, typename _Compare > unsigned int
__gnu_parallel::_LoserTree< false, _Tp, _Compare >::_init_winner
(unsigned int __root) [inline]`

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

Parameters

`__root` __index of the "game" to start.

Definition at line 277 of file losertree.h.

5.135.2.4 `template<typename _Tp, typename _Compare > void
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start
(const _Tp & __key, int __source, bool __sup) [inline,
inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

`__key` the element to insert

`__source` __index of the __source __sequence

`__sup` flag that determines whether the value to insert is an explicit __supremum.

Definition at line 130 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

5.135.3 Member Data Documentation

5.135.3.1 `template<typename _Tp , typename _Compare > _Compare
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp
[protected, inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

5.135.3.2 `template<typename _Tp , typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_M_first_insert [protected, inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`,
and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.135.3.3 `template<typename _Tp , typename _Compare > unsigned int
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k
[protected, inherited]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.135.3.4 `template<typename _Tp , typename _Compare > _Loser*
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers
[protected, inherited]`

`_LoserTree` __elements.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase()`.

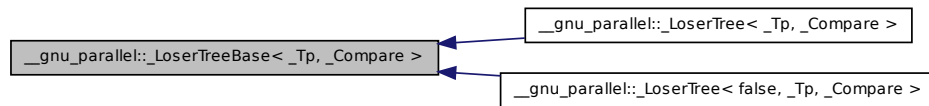
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.136 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference

Guarded loser/tournament tree.

Inheritance diagram for `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`:



Classes

- struct [_Loser](#)
Internal representation of a [_LoserTree](#) element.

Public Member Functions

- [_LoserTreeBase](#) (unsigned int __k, _Compare __comp)
- [~_LoserTreeBase](#) ()
- int [__get_min_source](#) ()
- void [__insert_start](#) (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare [_M_comp](#)
- bool [_M_first_insert](#)
- unsigned int [_M_ik](#)
- unsigned int [_M_k](#)
- unsigned int [_M_log_k](#)
- [_Loser](#) * [_M_losers](#)
- unsigned int [_M_offset](#)

5.136.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >
```

Guarded loser/tournament tree. The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

Parameters

`_Tp` the element type

`_Compare` the comparator to use, defaults to `std::less<_Tp>`

Definition at line 55 of file `losertree.h`.

5.136.2 Constructor & Destructor Documentation

```
5.136.2.1 template<typename _Tp , typename _Compare >
           __gnu_parallel::_LoserTreeBase< _Tp, _Compare
           >::_LoserTreeBase ( unsigned int __k, _Compare __comp )
           [inline]
```

The constructor.

Parameters

`__k` The number of sequences to merge.

`__comp` The comparator to use.

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::_rd_log2()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

```
5.136.2.2 template<typename _Tp , typename _Compare >
           __gnu_parallel::_LoserTreeBase< _Tp, _Compare
           >::~~_LoserTreeBase ( ) [inline]
```

The destructor.

Definition at line 118 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

5.136.3 Member Function Documentation

5.136.3.1 `template<typename _Tp, typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::__get_min_source() [inline]`

Returns

the index of the sequence with the smallest element.

Definition at line 151 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::__M_source`.

5.136.3.2 `template<typename _Tp, typename _Compare > void
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start
(const _Tp & __key, int __source, bool __sup) [inline]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

`__key` the element to insert

`__source` index of the `__source` sequence

`__sup` flag that determines whether the value to insert is an explicit `__supremum`.

Definition at line 130 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::__M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::__M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::__M_sup`.

5.136.4 Member Data Documentation

5.136.4.1 `template<typename _Tp, typename _Compare > _Compare
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__M_comp
[protected]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

5.136.4.2 `template<typename _Tp , typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_M_first_insert [protected]`

State flag that determines whether the [_LoserTree](#) is empty.

Only used for building the [_LoserTree](#).

Definition at line 85 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.136.4.3 `template<typename _Tp , typename _Compare > unsigned int
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k
[protected]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

5.136.4.4 `template<typename _Tp , typename _Compare > _Loser*
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers
[protected]`

[_LoserTree](#) __elements.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.137 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser` Struct Reference

Internal representation of a [_LoserTree](#) element.

Public Attributes

- [_Tp _M_key](#)
- [int _M_source](#)
- [bool _M_sup](#)

5.137.1 Detailed Description

`template<typename _Tp, typename _Compare> struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser`

Internal representation of a [_LoserTree](#) element.

Definition at line 59 of file `losertree.h`.

5.137.2 Member Data Documentation

5.137.2.1 `template<typename _Tp, typename _Compare >
_Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_Loser::_M_key`

`_M_key` of the element in the [_LoserTree](#).

Definition at line 66 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`.

5.137.2.2 `template<typename _Tp, typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_Loser::_M_source`

`__index` of the `__source` `__sequence`.

Definition at line 64 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`.

5.137.2.3 `template<typename _Tp, typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare
>::_Loser::_M_sup`

flag, true iff this is a "maximum" `__sentinel`.

Definition at line 62 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start()`.

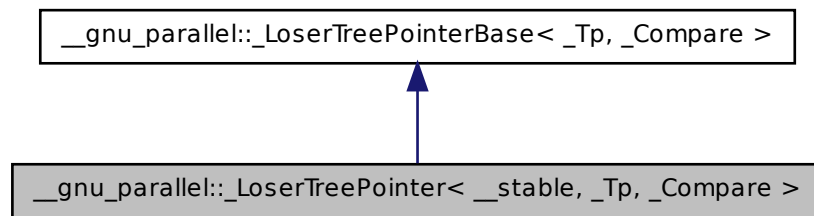
The documentation for this struct was generated from the following file:

- [losertree.h](#)

5.138 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Stable [LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointer` (unsigned int __k, _Compare __comp=[std::less](#)< _Tp >())
- void `__delete_min_insert` (const _Tp &__key, bool __sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`

5.139 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference 1185

- unsigned int `_M_k`
- [_Loser](#) * `_M_losers`
- unsigned int `_M_offset`

5.138.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`

Stable [_LoserTree](#) implementation. The unstable variant is implemented using partial instantiation below.

Definition at line 401 of file `losertree.h`.

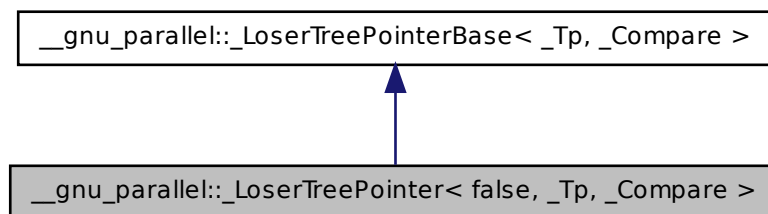
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.139 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Unstable [_LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointer` (unsigned int `__k`, `_Compare` `__comp=std::less< _Tp >()`)

- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- [_Loser](#) * **_M_losers**
- unsigned int **_M_offset**

5.139.1 Detailed Description

template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >

Unstable [_LoserTree](#) implementation. The stable variant is above.

Definition at line 482 of file losertree.h.

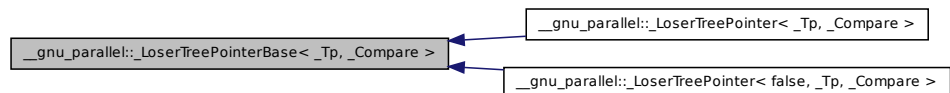
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.140 **__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference**

Base class of [_Loser](#) Tree implementation using pointers.

Inheritance diagram for **__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >**:



Classes

- struct [_Loser](#)
Internal representation of [_LoserTree](#) __elements.

Public Member Functions

- `_LoserTreePointerBase` (unsigned int __k, _Compare __comp=[std::less](#)< _Tp >())
- int `__get_min_source` ()
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- [_Loser](#) * `_M_losers`
- unsigned int `_M_offset`

5.140.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`

Base class of [_Loser](#) Tree implementation using pointers.

Definition at line 349 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.141 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser` Struct Reference

Internal representation of [_LoserTree](#) __elements.

Public Attributes

- `const _Tp * _M_keyp`
- `int _M_source`
- `bool _M_sup`

5.141.1 Detailed Description

`template<typename _Tp, typename _Compare> struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`

Internal representation of [_LoserTree](#) __elements.

Definition at line 353 of file `losertree.h`.

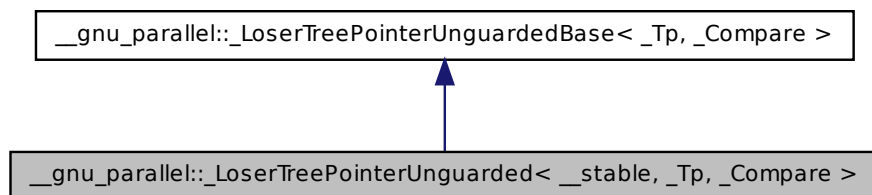
The documentation for this struct was generated from the following file:

- [losertree.h](#)

5.142 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Stable unguarded [_LoserTree](#) variant storing pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointerUnguarded` (unsigned int __k, const _Tp &__sentinel, _Compare __comp=`std::less< _Tp >()`)
- void `__delete_min_insert` (const _Tp &__key, bool __sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare _M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

5.142.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`

Stable unguarded [_LoserTree](#) variant storing pointers. Unstable variant is implemented below using partial specialization.

Definition at line 868 of file `losertree.h`.

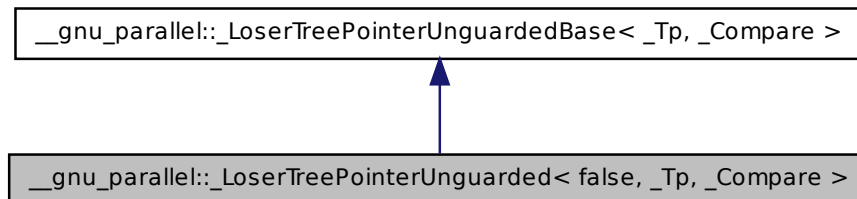
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.143 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

Unstable unguarded [_LoserTree](#) variant storing pointers.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



Public Member Functions

- **_LoserTreePointerUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=`std::less< _Tp >()`)
- void **_delete_min_insert** (const _Tp &__key, bool __sup)
- int **_get_min_source** ()
- void **_init** ()
- unsigned int **_init_winner** (unsigned int __root)
- void **_insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare` **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- `_Loser *` **_M_losers**
- unsigned int **_M_offset**

5.143.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`

Unstable unguarded [_LoserTree](#) variant storing pointers. Stable variant is above.

5.144 `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >` Class Template Reference 1191

Definition at line 953 of file `losertree.h`.

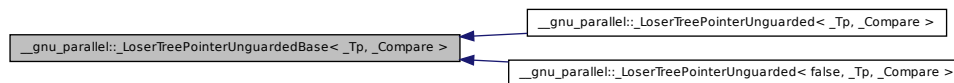
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.144 `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >` Class Template Reference

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`:



Public Member Functions

- `_LoserTreePointerUnguardedBase` (unsigned int __k, const _Tp &__sentinel, _Compare __comp=`std::less< _Tp >()`)
- `int __get_min_source ()`
- `void __insert_start` (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

5.144.1 Detailed Description

```
template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure. No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 805 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

5.145 __gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference

Traits for determining whether the loser tree should use pointers or copies.

Static Public Attributes

- static const bool [_M_use_pointer](#)

5.145.1 Detailed Description

```
template<typename _Tp> struct __gnu_parallel::_LoserTreeTraits< _Tp >
```

Traits for determining whether the loser tree should use pointers or copies. The field "[_M_use_pointer](#)" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };
```

```
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

5.146 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference 1193

Parameters

`_Tp` type to give the loser tree traits for.

Definition at line 727 of file `multiway_merge.h`.

5.145.2 Member Data Documentation

5.145.2.1 `template<typename _Tp > const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer` [static]

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

Definition at line 735 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

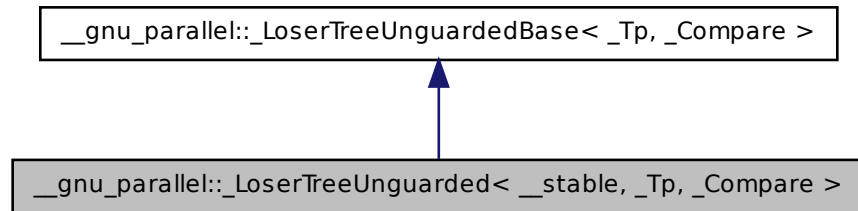
- [multiway_merge.h](#)

5.146 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Stable implementation of unguarded [_LoserTree](#).

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _`

Compare >:



Public Member Functions

- **_LoserTreeUnguarded** (unsigned int __k, const _Tp __sentinel, _Compare __comp=[std::less](#)< _Tp >())
- void **__delete_min_insert** (_Tp __key, bool)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser * **_M_losers**
- unsigned int **_M_offset**

5.146.1 Detailed Description

`template<bool __stable, typename _Tp, typename _Compare> class __gnu_parallel::__LoserTreeUnguarded< __stable, _Tp, _Compare >`

Stable implementation of unguarded [_LoserTree](#). Unstable variant is selected below with partial specialization.

5.147 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference 1195

Definition at line 627 of file `losertree.h`.

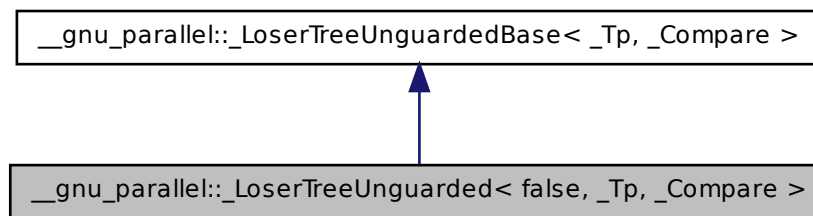
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.147 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference

Non-Stable implementation of unguarded [_LoserTree](#).

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:



Public Member Functions

- `_LoserTreeUnguarded` (unsigned int `__k`, const `_Tp` `__sentinel`, `_Compare` `__comp=std::less< _Tp >()`)
- void `__delete_min_insert` (`_Tp` `__key`, bool)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool)

Protected Attributes

- `_Compare` `_M_comp`

- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser * _M_losers`
- unsigned int `_M_offset`

5.147.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`

Non-Stable implementation of ungarded [_LoserTree](#). Stable implementation is above.

Definition at line 713 of file `losertree.h`.

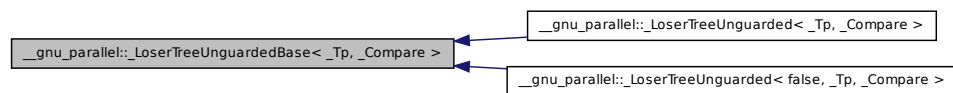
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.148 `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >` Class Template Reference

Base class for ungarded [_LoserTree](#) implementation.

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`:



Public Member Functions

- `_LoserTreeUnguardedBase` (unsigned int `__k`, const `_Tp` `__sentinel`, `_Compare` `__comp=std::less< _Tp >()`)
- int `__get_min_source` ()
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool)

Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `_Loser * _M_losers`
- `unsigned int _M_offset`

5.148.1 Detailed Description

`template<typename _Tp, typename _Compare> class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`

Base class for unguarded [_LoserTree](#) implementation. The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

Definition at line 564 of file `losertree.h`.

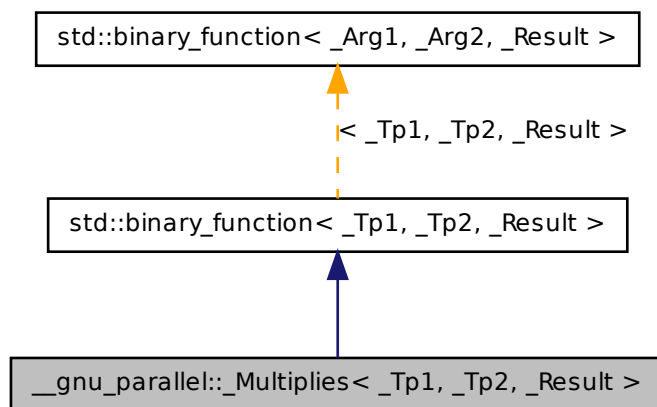
The documentation for this class was generated from the following file:

- [losertree.h](#)

5.149 `__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >` Struct Template Reference

Similar to `std::multiplies`, but allows two different types.

Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result >`:



Public Types

- typedef `_Tp1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Tp2` [second_argument_type](#)

Public Member Functions

- `_Result operator() (const _Tp1 &__x, const _Tp2 &__y) const`

5.149.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__ -
(*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0))> struct __gnu_parallel::_
Multiplies<_Tp1, _Tp2, _Result >
```

Similar to [std::multiplies](#), but allows two different types.

Definition at line 288 of file `parallel/base.h`.

5.149.2 Member Typedef Documentation

5.149.2.1 `typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.149.2.2 `typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result
>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.149.2.3 `typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

5.150 `__gnu_parallel::_Nothing` Struct Reference

Functor doing nothing.

Public Member Functions

- `template<typename _It >
void operator\(\) (_It)`

5.150.1 Detailed Description

Functor doing nothing. For some `__reduction` tasks (this is not a function object, but is passed as `__selector` `__dummy` parameter.

Definition at line 288 of file `for_each_selectors.h`.

5.150.2 Member Function Documentation

5.150.2.1 `template<typename _It > void __gnu_parallel::_Nothing::operator()
(_It) [inline]`

Functor execution.

Parameters

`__i` iterator referencing object.

Definition at line 294 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

5.151 `__gnu_parallel::_Piece<_DifferenceTp > Struct` Template Reference

Subsequence description.

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Attributes

- `_DifferenceType _M_begin`
- `_DifferenceType _M_end`

5.151.1 Detailed Description

`template<typename _DifferenceTp> struct __gnu_parallel::_Piece< _-
DifferenceTp >`

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

5.152 `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >` Struct Template Reference

5.151.2 Member Data Documentation

5.151.2.1 `template<typename _DifferenceTp > _DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_begin`

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

5.151.2.2 `template<typename _DifferenceTp > _DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end`

End of subsequence.

Definition at line 54 of file `multiway_mergesort.h`.

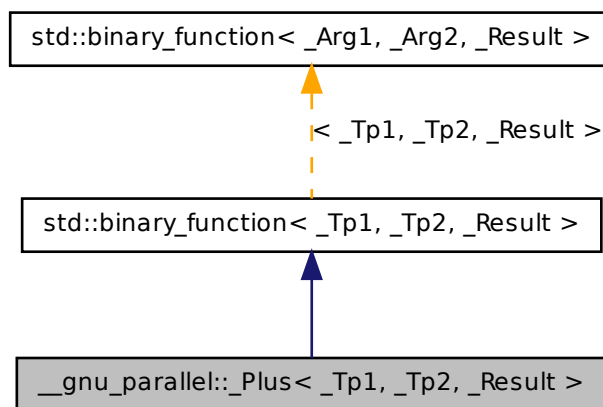
The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.152 `__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >` Struct Template Reference

Similar to [std::plus](#), but allows two different types.

Inheritance diagram for `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result >`:



Public Types

- typedef `_Tp1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Tp2` [second_argument_type](#)

Public Member Functions

- `_Result operator() (const _Tp1 &__x, const _Tp2 &__y) const`

5.152.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__ -
(*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))> struct __gnu_parallel::_
Plus<_Tp1, _Tp2, _Result >
```

Similar to [std::plus](#), but allows two different types.

Definition at line 272 of file `parallel/base.h`.

5.152.2 Member Typedef Documentation

5.152.2.1 `typedef _Tp1 std::binary_function<_Tp1, _Tp2, _Result>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.152.2.2 `typedef _Result std::binary_function<_Tp1, _Tp2, _Result>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.152.2.3 `typedef _Tp2 std::binary_function<_Tp1, _Tp2, _Result>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

5.153 `__gnu_parallel::__PMWMSSortingData<_RAIter>` Struct Template Reference

Data accessed by all threads.

Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::iterator_traits<_RAIter> _TraitsType`
- `typedef _TraitsType::value_type _ValueType`

Public Attributes

- `_ThreadIndex _M_num_threads`
- `_DifferenceType * _M_offsets`

- `std::vector<_Piece<_DifferenceType>>> *_M_pieces`
- `_ValueType *_M_samples`
- `_RAIter _M_source`
- `_DifferenceType *_M_starts`
- `_ValueType **_M_temporary`

5.153.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::PMWMSortingData<_RAIter>`

Data accessed by all threads. PMWMS = parallel multiway mergesort

Definition at line 61 of file `multiway_mergesort.h`.

5.153.2 Member Data Documentation

5.153.2.1 `template<typename _RAIter> _ThreadIndex
__gnu_parallel::PMWMSortingData<_RAIter>::
_M_num_threads`

Number of threads involved.

Definition at line 68 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

5.153.2.2 `template<typename _RAIter> _DifferenceType*
__gnu_parallel::PMWMSortingData<_RAIter>::
_M_offsets`

Offsets to add to the found positions.

Definition at line 83 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

5.153.2.3 `template<typename _RAIter> std::vector<_Piece<_
_DifferenceType>>> *_M_pieces`

Pieces of data to merge [thread][__sequence].

Definition at line 86 of file `multiway_mergesort.h`.

5.153 __gnu_parallel::PMWMSSortingData<_RAIter> Struct Template Reference 1205

Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

5.153.2.4 template<typename _RAIter> _ValueType* __gnu_parallel::PMWMSSortingData<_RAIter>::_M_samples

Samples.

Definition at line 80 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), and __gnu_parallel::parallel_sort_mwms().

5.153.2.5 template<typename _RAIter> _RAIter __gnu_parallel::PMWMSSortingData<_RAIter>::_M_source

Input __begin.

Definition at line 71 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

5.153.2.6 template<typename _RAIter> _DifferenceType* __gnu_parallel::PMWMSSortingData<_RAIter>::_M_starts

Start indices, per thread.

Definition at line 74 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

5.153.2.7 template<typename _RAIter> _ValueType** __gnu_parallel::PMWMSSortingData<_RAIter>::_M_temporary

Storage in which to sort.

Definition at line 77 of file multiway_mergesort.h.

Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.154 `__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >` Class Template Reference

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Public Types

- `typedef _DifferenceTp _DifferenceType`
- `typedef _PseudoSequenceIterator< _Tp, uint64_t > iterator`

Public Member Functions

- `_PseudoSequence (const _Tp &__val, _DifferenceType __count)`
- `iterator begin () const`
- `iterator end () const`

5.154.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >
```

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Parameters

- `_Tp` Sequence `_M` value type.
- `_DifferenceTp` Sequence difference type.

Definition at line 359 of file `parallel/base.h`.

5.154.2 Constructor & Destructor Documentation

```
5.154.2.1 template<typename _Tp, typename _DifferenceTp>
        __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp
        >::_PseudoSequence ( const _Tp & __val, _DifferenceType __count
        ) [inline]
```

Constructor.

5.155 `__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >` Class Template Reference 1207

Parameters

- `_M_val` Element of the sequence.
- `__count` Number of (virtual) copies.

Definition at line 371 of file `parallel/base.h`.

5.154.3 Member Function Documentation

5.154.3.1 `template<typename _Tp, typename _DifferenceTp> iterator
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::begin () const [inline]`

Begin iterator.

Definition at line 376 of file `parallel/base.h`.

5.154.3.2 `template<typename _Tp, typename _DifferenceTp> iterator
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::end () const [inline]`

End iterator.

Definition at line 381 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.155 `__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >` Class Template Reference

`_Iterator` associated with [__gnu_parallel::_PseudoSequence](#). If features the usual random-access iterator functionality.

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Member Functions

- `_PseudoSequenceIterator` (`const _Tp &__val, _DifferenceType __pos`)

- `bool operator!= (const _PseudoSequenceIterator &__i2)`
- `const _Tp & operator* () const`
- `_PseudoSequenceIterator & operator++ ()`
- `_PseudoSequenceIterator operator++ (int)`
- `_DifferenceType operator- (const _PseudoSequenceIterator &__i2)`
- `bool operator== (const _PseudoSequenceIterator &__i2)`
- `const _Tp & operator[] (_DifferenceType) const`

5.155.1 Detailed Description

`template<typename _Tp, typename _DifferenceTp> class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >`

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. If features the usual random-access iterator functionality.

Parameters

- `_Tp` Sequence `_M_value` type.
- `_DifferenceTp` Sequence difference type.

Definition at line 306 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

5.156 `__gnu_parallel::_QSBThreadLocal`< `_RAIter` > Struct Template Reference

Information local to one thread in the parallel quicksort run.

Public Types

- `typedef _TraitsType::difference_type _DifferenceType`
- `typedef std::pair< _RAIter, _RAIter > _Piece`
- `typedef std::iterator_traits< _RAIter > _TraitsType`

Public Member Functions

- `_QSBThreadLocal (int __queue_size)`

Public Attributes

- volatile _DifferenceType * [_M_elements_leftover](#)
- [_Piece _M_global](#)
- [_Piece _M_initial](#)
- [_RestrictedBoundedConcurrentQueue<_Piece> _M_leftover_parts](#)
- [_ThreadIndex _M_num_threads](#)

5.156.1 Detailed Description

`template<typename _RAIter> struct __gnu_parallel::_QSBThreadLocal<_RAIter>`

Information local to one thread in the parallel quicksort run.

Definition at line 62 of file `balanced_quicksort.h`.

5.156.2 Member Typedef Documentation

5.156.2.1 `template<typename _RAIter> typedef std::pair<_RAIter, _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_Piece`

Continuous part of the sequence, described by an iterator pair.

Definition at line 69 of file `balanced_quicksort.h`.

5.156.3 Constructor & Destructor Documentation

5.156.3.1 `template<typename _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_QSBThreadLocal (int __queue_size) [inline]`

Constructor.

Parameters

`__queue_size` size of the work-stealing queue.

Definition at line 88 of file `balanced_quicksort.h`.

5.156.4 Member Data Documentation

5.156.4.1 `template<typename _RAIter> volatile _DifferenceType*
__gnu_parallel::__QSBThreadLocal< _RAIter
>::__M_elements_leftover`

Pointer to a counter of elements left over to sort.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.156.4.2 `template<typename _RAIter> _Piece __gnu_parallel::__-
QSBThreadLocal< _RAIter >::__M_global`

The complete sequence to sort.

Definition at line 84 of file `balanced_quicksort.h`.

5.156.4.3 `template<typename _RAIter> _Piece __gnu_parallel::__-
QSBThreadLocal< _RAIter >::__M_initial`

Initial piece to work on.

Definition at line 72 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.156.4.4 `template<typename _RAIter> _-
RestrictedBoundedConcurrentQueue< _Piece>
__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_leftover_parts`

Work-stealing queue.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.156.4.5 `template<typename _RAIter> _ThreadIndex
__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_num_threads`

Number of threads involved in this algorithm.

Definition at line 78 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced_quicksort.h](#)

5.157 `__gnu_parallel::_RandomNumber` Class Reference

Random number generator, based on the Mersenne twister.

Public Member Functions

- [_RandomNumber](#) ()
- [_RandomNumber](#) (uint32_t __seed, uint64_t _M_supremum=0x100000000ULL)
- unsigned long [__genrand_bits](#) (int __bits)
- uint32_t [operator\(\)](#) ()
- uint32_t [operator\(\)](#) (uint64_t local_supremum)

5.157.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

5.157.2 Constructor & Destructor Documentation

5.157.2.1 `__gnu_parallel::_RandomNumber::_RandomNumber ()` [`inline`]

Default constructor. Seed with 0.

Definition at line 74 of file `random_number.h`.

5.157.2.2 `__gnu_parallel::_RandomNumber::_RandomNumber (uint32_t __seed, uint64_t _M_supremum = 0x100000000ULL)` [`inline`]

Constructor.

Parameters

__seed Random __seed.

_M_supremum Generate integer random numbers in the interval [0, *_M_supremum*).

Definition at line 85 of file random_number.h.

5.157.3 Member Function Documentation

5.157.3.1 `unsigned long __gnu_parallel::_RandomNumber::_genrand_bits (int __bits) [inline]`

Generate a number of random bits, run-time parameter.

Parameters

bits Number of bits to generate.

Definition at line 109 of file random_number.h.

5.157.3.2 `uint32_t __gnu_parallel::_RandomNumber::operator() () [inline]`

Generate unsigned random 32-bit integer.

Definition at line 94 of file random_number.h.

5.157.3.3 `uint32_t __gnu_parallel::_RandomNumber::operator() (uint64_t local_supremum) [inline]`

Generate unsigned random 32-bit integer in the interval [0, *local_supremum*).

Definition at line 100 of file random_number.h.

The documentation for this class was generated from the following file:

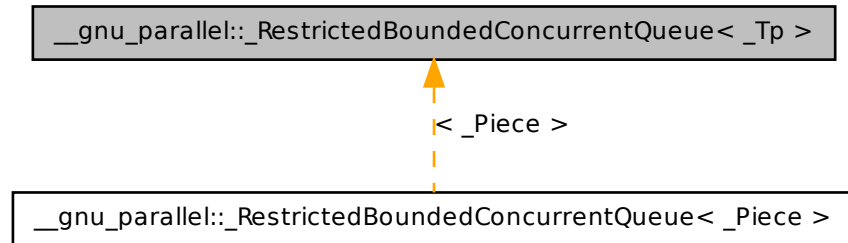
- [random_number.h](#)

5.158 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference

Double-ended queue of bounded size, allowing lock-free atomic access. [push_front\(\)](#) and [pop_front\(\)](#) must not be called concurrently to each other, while [pop_back\(\)](#) can be called concurrently at all times. [empty\(\)](#), [size\(\)](#), and [top\(\)](#) are intentionally not provided. Calling them would not make sense in a concurrent setting.

5.158 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference 1213

Inheritance diagram for `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`:



Public Member Functions

- [`_RestrictedBoundedConcurrentQueue`](#) ([`_SequenceIndex`](#) __max_size)
- [`~_RestrictedBoundedConcurrentQueue`](#) ()
- bool [`pop_back`](#) ([`_Tp`](#) &__t)
- bool [`pop_front`](#) ([`_Tp`](#) &__t)
- void [`push_front`](#) (const [`_Tp`](#) &__t)

5.158.1 Detailed Description

template<typename `_Tp`> **class** **`__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>`**

Double-ended queue of bounded size, allowing lock-free atomic access. [`push_front\(\)`](#) and [`pop_front\(\)`](#) must not be called concurrently to each other, while [`pop_back\(\)`](#) can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Parameters

`_Tp` Contained element type.

Definition at line 52 of file `queue.h`.

5.158.2 Constructor & Destructor Documentation

5.158.2.1 `template<typename _Tp> __gnu_parallel::_-
RestrictedBoundedConcurrentQueue< _Tp
>::_RestrictedBoundedConcurrentQueue (_SequenceIndex
__max_size) [inline]`

Constructor. Not to be called concurrent, of course.

Parameters

__M_max_size Maximal number of elements to be contained.

Definition at line 68 of file queue.h.

5.158.2.2 `template<typename _Tp> __gnu_parallel::_-
RestrictedBoundedConcurrentQueue< _Tp
>::~~RestrictedBoundedConcurrentQueue () [inline]`

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file queue.h.

5.158.3 Member Function Documentation

5.158.3.1 `template<typename _Tp> bool __gnu_parallel::_-
RestrictedBoundedConcurrentQueue< _Tp >::pop_back (_Tp &
__t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with [pop_front\(\)](#).

Definition at line 127 of file queue.h.

5.158.3.2 `template<typename _Tp> bool __gnu_parallel::_-
RestrictedBoundedConcurrentQueue< _Tp >::pop_front (_Tp &
__t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with [pop_front\(\)](#).

Definition at line 100 of file queue.h.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.159 `__gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference **1215**

5.158.3.3 `template<typename _Tp> void __gnu_parallel::__RestrictedBoundedConcurrentQueue< _Tp >::push_front (const _Tp & __t) [inline]`

Pushes one element into the queue at the front end. Must not be called concurrently with [pop_front\(\)](#).

Definition at line 83 of file [queue.h](#).

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this class was generated from the following file:

- [queue.h](#)

5.159 `__gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Stable sorting functor.

Public Member Functions

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

5.159.1 Detailed Description

`template<bool __stable, class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::__SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

Stable sorting functor. Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1004 of file [multiway_merge.h](#).

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.160 `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference`

Non-`__stable` sorting functor.

Public Member Functions

- `void operator() (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)`

5.160.1 Detailed Description

`template<class _RAIter, class _StrictWeakOrdering> struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

Non-`__stable` sorting functor. Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1017 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

5.161 `__gnu_parallel::_Settings Struct Reference`

`class _Settings` /// Run-time settings for the parallel mode including all tunable parameters.

Static Public Member Functions

- `static _GLIBCXX_CONST const _Settings & get () throw ()`
- `static void set (_Settings &) throw ()`

Public Attributes

- `_SequenceIndex accumulate_minimal_n`
- `unsigned int adjacent_difference_minimal_n`
- `_AlgorithmStrategy algorithm_strategy`

- unsigned int [cache_line_size](#)
- [_SequenceIndex](#) count_minimal_n
- [_SequenceIndex](#) fill_minimal_n
- [_FindAlgorithm](#) **find_algorithm**
- double [find_increasing_factor](#)
- [_SequenceIndex](#) find_initial_block_size
- [_SequenceIndex](#) find_maximum_block_size
- float [find_scale_factor](#)
- [_SequenceIndex](#) find_sequential_search_size
- [_SequenceIndex](#) for_each_minimal_n
- [_SequenceIndex](#) generate_minimal_n
- unsigned long long [L1_cache_size](#)
- unsigned long long [L2_cache_size](#)
- [_SequenceIndex](#) max_element_minimal_n
- [_SequenceIndex](#) merge_minimal_n
- unsigned int [merge_oversampling](#)
- [_SplittingAlgorithm](#) **merge_splitting**
- [_SequenceIndex](#) min_element_minimal_n
- [_MultiwayMergeAlgorithm](#) **multiway_merge_algorithm**
- int [multiway_merge_minimal_k](#)
- [_SequenceIndex](#) multiway_merge_minimal_n
- unsigned int [multiway_merge_oversampling](#)
- [_SplittingAlgorithm](#) **multiway_merge_splitting**
- [_SequenceIndex](#) nth_element_minimal_n
- [_SequenceIndex](#) partial_sort_minimal_n
- [_PartialSumAlgorithm](#) **partial_sum_algorithm**
- float [partial_sum_dilation](#)
- unsigned int [partial_sum_minimal_n](#)
- double [partition_chunk_share](#)
- [_SequenceIndex](#) partition_chunk_size
- [_SequenceIndex](#) partition_minimal_n
- [_SequenceIndex](#) qsb_steals
- unsigned int [random_shuffle_minimal_n](#)
- [_SequenceIndex](#) replace_minimal_n
- [_SequenceIndex](#) search_minimal_n
- [_SequenceIndex](#) set_difference_minimal_n
- [_SequenceIndex](#) set_intersection_minimal_n
- [_SequenceIndex](#) set_symmetric_difference_minimal_n
- [_SequenceIndex](#) set_union_minimal_n
- [_SortAlgorithm](#) **sort_algorithm**
- [_SequenceIndex](#) sort_minimal_n
- unsigned int [sort_mwms_oversampling](#)
- unsigned int [sort_qs_num_samples_preset](#)

- [_SequenceIndex](#) `sort_qsb_base_case_maximal_n`
- [_SplittingAlgorithm](#) `sort_splitting`
- unsigned int `TLB_size`
- [_SequenceIndex](#) `transform_minimal_n`
- [_SequenceIndex](#) `unique_copy_minimal_n`
- [_SequenceIndex](#) `workstealing_chunk_size`

5.161.1 Detailed Description

class [_Settings](#) /// Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file settings.h.

5.161.2 Member Function Documentation

5.161.2.1 static [_GLIBCXX_CONST](#) const [_Settings&](#) [__gnu_parallel::_Settings::get](#) () throw () [**static**]

Get the global settings.

Referenced by [__gnu_parallel::__find_template\(\)](#), [__gnu_parallel::__for_each_template_random_access_workstealing\(\)](#), [__gnu_parallel::__parallel_nth_element\(\)](#), [__gnu_parallel::__parallel_partial_sum\(\)](#), [__gnu_parallel::__parallel_partial_sum_linear\(\)](#), [__gnu_parallel::__parallel_partition\(\)](#), [__gnu_parallel::__parallel_sort\(\)](#), [__gnu_parallel::__parallel_sort_qs_conquer\(\)](#), [__gnu_parallel::__qsb_local_sort_with_helping\(\)](#), [__gnu_parallel::multiway_merge_sampling_splitting\(\)](#), [__gnu_parallel::parallel_multiway_merge\(\)](#), [__gnu_parallel::parallel_sort_mwms\(\)](#), and [__gnu_parallel::parallel_sort_mwms_pu\(\)](#).

5.161.2.2 static void [__gnu_parallel::_Settings::set](#) ([_Settings &](#)) throw () [**static**]

Set the global settings.

5.161.3 Member Data Documentation

5.161.3.1 [_SequenceIndex](#) [__gnu_parallel::_Settings::accumulate_minimal_n](#)

Minimal input size for accumulate.

Definition at line 139 of file settings.h.

5.161.3.2 unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n

Minimal input size for adjacent_difference.

Definition at line 142 of file settings.h.

5.161.3.3 unsigned int __gnu_parallel::_Settings::cache_line_size

Overestimation of cache line size. Used to avoid false /// sharing, i.e. elements of different threads are at least this /// amount apart.

Definition at line 265 of file settings.h.

Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

5.161.3.4 _SequenceIndex __gnu_parallel::_Settings::count_minimal_n

Minimal input size for count and count_if.

Definition at line 145 of file settings.h.

5.161.3.5 _SequenceIndex __gnu_parallel::_Settings::fill_minimal_n

Minimal input size for fill.

Definition at line 148 of file settings.h.

5.161.3.6 double __gnu_parallel::_Settings::find_increasing_factor

Block size increase factor for find.

Definition at line 151 of file settings.h.

5.161.3.7 _SequenceIndex __gnu_parallel::_Settings::find_initial_block_size

Initial block size for find.

Definition at line 154 of file settings.h.

Referenced by __gnu_parallel::__find_template().

5.161.3.8 _SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size

Maximal block size for find.

Definition at line 157 of file settings.h.

5.161.3.9 float __gnu_parallel::_Settings::find_scale_factor

Block size scale-down factor with respect to current position.

Definition at line 276 of file settings.h.

Referenced by __gnu_parallel::_find_template().

5.161.3.10 _SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size

Start with looking for this many elements sequentially, for find.

Definition at line 160 of file settings.h.

Referenced by __gnu_parallel::_find_template().

5.161.3.11 _SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n

Minimal input size for for_each.

Definition at line 163 of file settings.h.

5.161.3.12 _SequenceIndex __gnu_parallel::_Settings::generate_minimal_n

Minimal input size for generate.

Definition at line 166 of file settings.h.

5.161.3.13 unsigned long long __gnu_parallel::_Settings::L1_cache_size

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file settings.h.

5.161.3.14 unsigned long long __gnu_parallel::_Settings::L2_cache_size

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file settings.h.

Referenced by __gnu_parallel::_parallel_random_shuffle_drs(), and __gnu_parallel::_sequential_random_shuffle().

**5.161.3.15 _SequenceIndex __gnu_parallel::_Settings::max_element -
 minimal_n**

Minimal input size for max_element.

Definition at line 169 of file settings.h.

5.161.3.16 _SequenceIndex __gnu_parallel::_Settings::merge_minimal_n

Minimal input size for merge.

Definition at line 172 of file settings.h.

5.161.3.17 unsigned int __gnu_parallel::_Settings::merge_oversampling

Oversampling factor for merge.

Definition at line 175 of file settings.h.

Referenced by __gnu_parallel::multiway_merge_sampling_splitting(), and __gnu_parallel::parallel_multiway_merge().

**5.161.3.18 _SequenceIndex __gnu_parallel::_Settings::min_element_minimal -
 n**

Minimal input size for min_element.

Definition at line 178 of file settings.h.

5.161.3.19 int __gnu_parallel::_Settings::multiway_merge_minimal_k

Oversampling factor for multiway_merge.

Definition at line 184 of file settings.h.

**5.161.3.20 _SequenceIndex __gnu_parallel::_Settings::multiway_merge -
 minimal_n**

Minimal input size for multiway_merge.

Definition at line 181 of file settings.h.

5.161.3.21 unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling

Oversampling factor for multiway_merge.

Definition at line 187 of file settings.h.

5.161.3.22 _SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n

Minimal input size for nth_element.

Definition at line 190 of file settings.h.

Referenced by __gnu_parallel::_parallel_nth_element().

5.161.3.23 _SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n

Minimal input size for partial_sort.

Definition at line 203 of file settings.h.

5.161.3.24 float __gnu_parallel::_Settings::partial_sum_dilation

Ratio for partial_sum. Assume "sum and write result" to be /// this factor slower than just "sum".

Definition at line 207 of file settings.h.

Referenced by __gnu_parallel::_parallel_partial_sum_linear().

5.161.3.25 unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n

Minimal input size for partial_sum.

Definition at line 210 of file settings.h.

5.161.3.26 double __gnu_parallel::_Settings::partition_chunk_share

Chunk size for partition, relative to input size. If > 0.0, /// this value overrides partition_chunk_size.

Definition at line 197 of file settings.h.

Referenced by __gnu_parallel::_parallel_partition().

5.161.3.27 _SequenceIndex __gnu_parallel::_Settings::partition_chunk_size

Chunk size for partition.

Definition at line 193 of file settings.h.

Referenced by __gnu_parallel::__parallel_partition().

5.161.3.28 _SequenceIndex __gnu_parallel::_Settings::partition_minimal_n

Minimal input size for partition.

Definition at line 200 of file settings.h.

Referenced by __gnu_parallel::__parallel_nth_element().

5.161.3.29 _SequenceIndex __gnu_parallel::_Settings::qsb_steals

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

5.161.3.30 unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n

Minimal input size for random_shuffle.

Definition at line 213 of file settings.h.

5.161.3.31 _SequenceIndex __gnu_parallel::_Settings::replace_minimal_n

Minimal input size for replace and replace_if.

Definition at line 216 of file settings.h.

5.161.3.32 _SequenceIndex __gnu_parallel::_Settings::search_minimal_n

Minimal input size for search and search_n.

Definition at line 273 of file settings.h.

5.161.3.33 _SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n

Minimal input size for set_difference.

Definition at line 219 of file settings.h.

5.161.3.34 `_SequenceIndex __gnu_parallel::_Settings::set_intersection_minimal_n`

Minimal input size for `set_intersection`.

Definition at line 222 of file `settings.h`.

5.161.3.35 `_SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Definition at line 225 of file `settings.h`.

5.161.3.36 `_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

5.161.3.37 `_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

5.161.3.38 `unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for parallel `std::sort` (MWMS).

Definition at line 234 of file `settings.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

5.161.3.39 `unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file `settings.h`.

5.162 __gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference 1225

5.161.3.40 _SequenceIndex __gnu_parallel::Settings::sort_qsb_base_case_maximal_n

Maximal subsequence __length to switch to unbalanced __base case. /// Applies to std::sort with dynamically load-balanced quicksort.

Definition at line 241 of file settings.h.

Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

5.161.3.41 unsigned int __gnu_parallel::Settings::TLB_size

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file settings.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__sequential_random_shuffle().

5.161.3.42 _SequenceIndex __gnu_parallel::Settings::transform_minimal_n

Minimal input size for parallel std::transform.

Definition at line 244 of file settings.h.

5.161.3.43 _SequenceIndex __gnu_parallel::Settings::unique_copy_minimal_n

Minimal input size for unique_copy.

Definition at line 247 of file settings.h.

The documentation for this struct was generated from the following file:

- [settings.h](#)

5.162 __gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

Split consistently.

5.162.1 Detailed Description

```
template<bool __exact, typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< __exact, _-
RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

Definition at line 122 of file multiway_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.163 __gnu_parallel::_SplitConsistently< false, _-RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

Split by sampling.

Public Member Functions

- void **operator()** (const [_ThreadIndex](#) __iam, [_PMWMSortingData](#)< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

5.163.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::_SplitConsistently< false,
_RAIter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

Definition at line 187 of file multiway_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.164 `__gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference 1227

5.164 `__gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

Split by exact splitting.

Public Member Functions

- void **operator()** (const [_ThreadIndex](#) __iam, [_PMWMSSortingData](#)< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

5.164.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _-
SortingPlacesIterator> struct __gnu_parallel::SplitConsistently< true, _-
RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

Definition at line 128 of file `multiway_mergesort.h`.

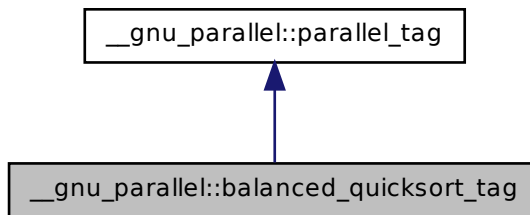
The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

5.165 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Forces parallel sorting using balanced quicksort at compile time.

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



Public Member Functions

- `balanced_quicksort_tag` (`_ThreadIndex` __num_threads)
- `_ThreadIndex` `__get_num_threads` ()
- `void` `set_num_threads` (`_ThreadIndex` __num_threads)

5.165.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file tags.h.

5.165.2 Member Function Documentation

5.165.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.165.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]

Set the desired number of threads.

Parameters

__num_threads Desired number of threads.

Definition at line 73 of file tags.h.

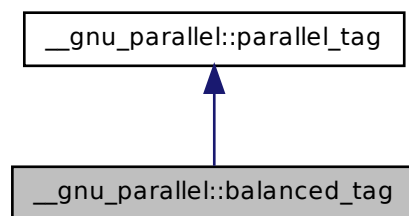
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.166 __gnu_parallel::balanced_tag Struct Reference

Recommends parallel execution using dynamic load-balancing at compile time.

Inheritance diagram for __gnu_parallel::balanced_tag:



Public Member Functions

- [_ThreadIndex](#) __get_num_threads ()
- void [set_num_threads](#) ([_ThreadIndex](#) __num_threads)

5.166.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file tags.h.

5.166.2 Member Function Documentation

5.166.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.166.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` [inline, inherited]

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

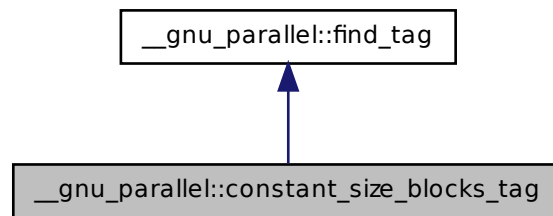
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.167 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

Selects the constant block size variant for `std::find()`.

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



5.167.1 Detailed Description

Selects the constant block size variant for `std::find()`.

See also

[_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS](#)

Definition at line 178 of file `tags.h`.

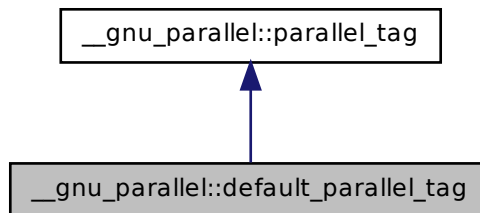
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.168 `__gnu_parallel::default_parallel_tag` Struct Reference

Recommends parallel execution using the default parallel algorithm.

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



Public Member Functions

- `default_parallel_tag` (`_ThreadIndex` __num_threads)
- `_ThreadIndex` `__get_num_threads` ()
- `void` `set_num_threads` (`_ThreadIndex` __num_threads)

5.168.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file tags.h.

5.168.2 Member Function Documentation

5.168.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.168.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]

Set the desired number of threads.

Parameters

__num_threads Desired number of threads.

Definition at line 73 of file tags.h.

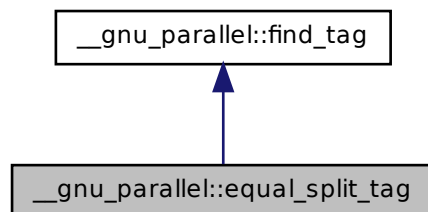
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.169 __gnu_parallel::equal_split_tag Struct Reference

Selects the equal splitting variant for std::find().

Inheritance diagram for __gnu_parallel::equal_split_tag:



5.169.1 Detailed Description

Selects the equal splitting variant for std::find().

See also

[_GLIBCXX_FIND_EQUAL_SPLIT](#)

Definition at line 182 of file tags.h.

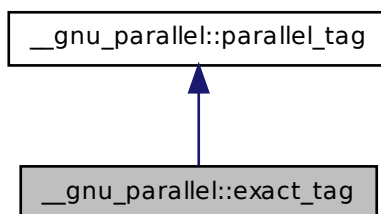
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.170 __gnu_parallel::exact_tag Struct Reference

Forces parallel merging with exact splitting, at compile time.

Inheritance diagram for __gnu_parallel::exact_tag:



Public Member Functions

- `exact_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

5.170.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file tags.h.

5.170.2 Member Function Documentation

5.170.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

5.170.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline, inherited]

Set the desired number of threads.

Parameters

__num_threads Desired number of threads.

Definition at line 73 of file tags.h.

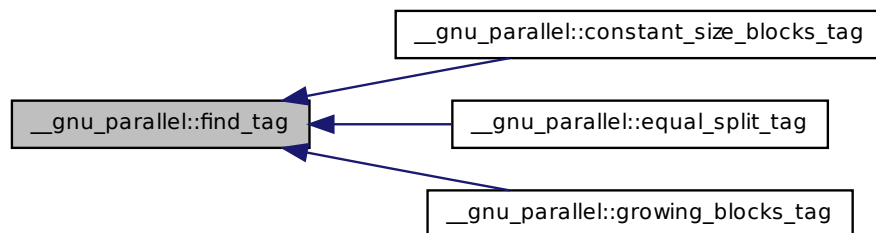
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.171 __gnu_parallel::find_tag Struct Reference

Base class for for std::find() variants.

Inheritance diagram for `__gnu_parallel::find_tag`:



5.171.1 Detailed Description

Base class for `std::find()` variants.

Definition at line 104 of file `tags.h`.

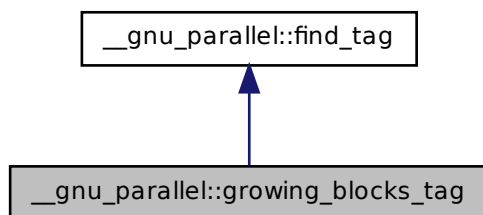
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.172 `__gnu_parallel::growing_blocks_tag` Struct Reference

Selects the growing block size variant for `std::find()`.

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



5.172.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

[_GLIBCXX_FIND_GROWING_BLOCKS](#)

Definition at line 174 of file `tags.h`.

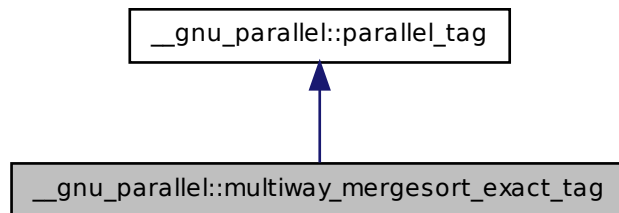
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.173 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:



Public Member Functions

- `multiway_mergesort_exact_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

5.173.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file tags.h.

5.173.2 Member Function Documentation

5.173.2.1 [_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.174 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference 1239

5.173.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

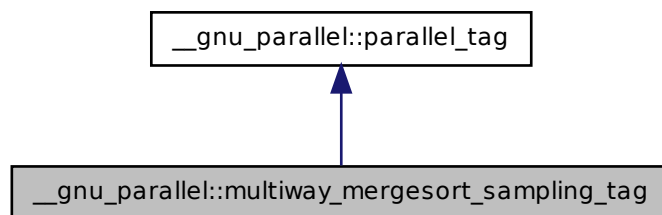
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.174 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:



Public Member Functions

- `multiway_mergesort_sampling_tag (_ThreadIndex __num_threads)`
- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

5.174.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file tags.h.

5.174.2 Member Function Documentation

5.174.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.174.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` [inline, inherited]

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

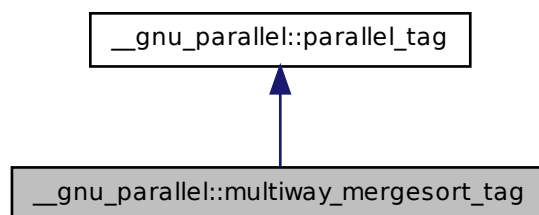
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.175 `__gnu_parallel::multiway_mergesort_tag` Struct Reference

Forces parallel sorting using multiway mergesort at compile time.

Inheritance diagram for __gnu_parallel::multiway_mergesort_tag:



Public Member Functions

- `multiway_mergesort_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

5.175.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file tags.h.

5.175.2 Member Function Documentation

5.175.2.1 [_ThreadIndex](#) `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.175.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

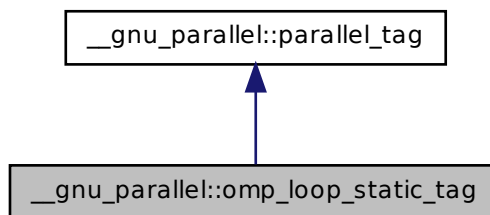
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.176 `__gnu_parallel::omp_loop_static_tag` Struct Reference

Recommends parallel execution using OpenMP static load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::omp_loop_static_tag`:



Public Member Functions

- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

5.176.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file tags.h.

5.176.2 Member Function Documentation

5.176.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.176.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` [inline, inherited]

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file tags.h.

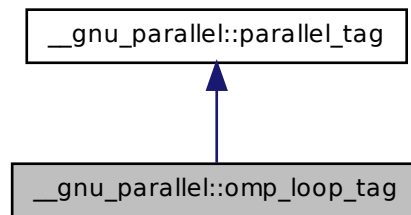
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.177 `__gnu_parallel::omp_loop_tag` Struct Reference

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::omp_loop_tag`:



Public Member Functions

- [_ThreadIndex __get_num_threads\(\)](#)
- void [set_num_threads\(_ThreadIndex __num_threads\)](#)

5.177.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

5.177.2 Member Function Documentation

5.177.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.177.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

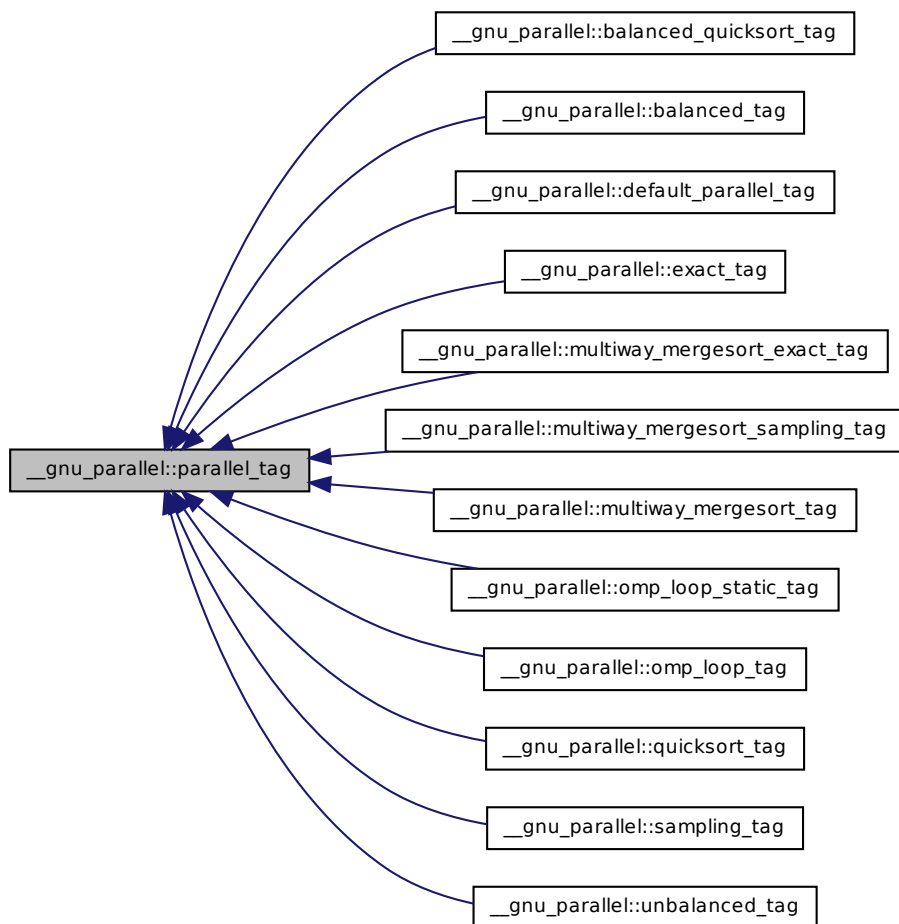
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.178 `__gnu_parallel::parallel_tag` Struct Reference

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Inheritance diagram for `__gnu_parallel::parallel_tag`:



Public Member Functions

- [parallel_tag](#) ()
- [parallel_tag](#) ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) [__get_num_threads](#) ()
- void [set_num_threads](#) ([_ThreadIndex](#) __num_threads)

5.178.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

5.178.2 Constructor & Destructor Documentation

5.178.2.1 `__gnu_parallel::parallel_tag::parallel_tag () [inline]`

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

5.178.2.2 `__gnu_parallel::parallel_tag::parallel_tag (_ThreadIndex __num_threads) [inline]`

Default constructor. Recommend number of threads to use.

Parameters

`__num_threads` Desired number of threads.

Definition at line 58 of file tags.h.

5.178.3 Member Function Documentation

5.178.3.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.178.3.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline]`

Set the desired number of threads.

Parameters

__num_threads Desired number of threads.

Definition at line 73 of file tags.h.

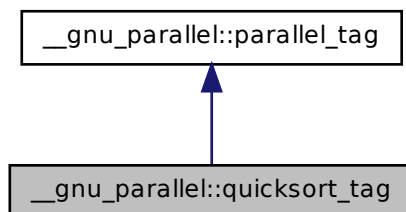
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.179 `__gnu_parallel::quicksort_tag` Struct Reference

Forces parallel sorting using unbalanced quicksort at compile time.

Inheritance diagram for `__gnu_parallel::quicksort_tag`:

**Public Member Functions**

- `quicksort_tag` ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

5.179.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file tags.h.

5.179.2 Member Function Documentation

5.179.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.179.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` [inline, inherited]

Set the desired number of threads.

Parameters

__num_threads Desired number of threads.

Definition at line 73 of file `tags.h`.

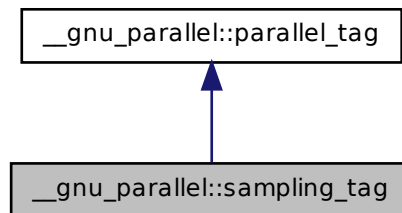
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.180 `__gnu_parallel::sampling_tag` Struct Reference

Forces parallel merging with exact splitting, at compile time.

Inheritance diagram for `__gnu_parallel::sampling_tag`:



Public Member Functions

- `sampling_tag` (`_ThreadIndex` __num_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` __num_threads)

5.180.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

5.180.2 Member Function Documentation

5.180.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`().

5.180.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

5.181 `__gnu_parallel::sequential_tag` Struct Reference

Forces sequential execution at compile time.

5.181.1 Detailed Description

Forces sequential execution at compile time.

Definition at line 42 of file `tags.h`.

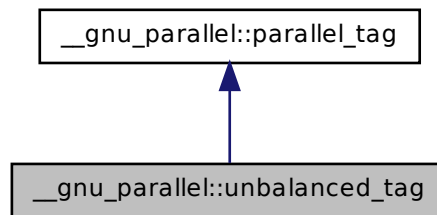
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.182 `__gnu_parallel::unbalanced_tag` Struct Reference

Recommends parallel execution using static load-balancing at compile time.

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:



Public Member Functions

- [_ThreadIndex __get_num_threads \(\)](#)
- void [set_num_threads \(_ThreadIndex __num_threads\)](#)

5.182.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file `tags.h`.

5.182.2 Member Function Documentation

5.182.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` [inline, inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

5.182.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex
__num_threads) [inline, inherited]`

Set the desired number of threads.

Parameters

`__num_threads` Desired number of threads.

Definition at line 73 of file `tags.h`.

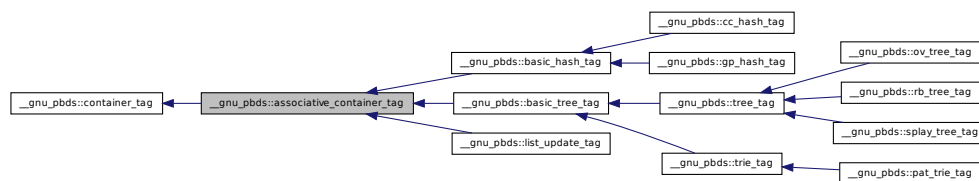
The documentation for this struct was generated from the following file:

- [tags.h](#)

5.183 `__gnu_pbds::associative_container_tag` Struct Reference

Basic associative-container.

Inheritance diagram for `__gnu_pbds::associative_container_tag`:



5.183.1 Detailed Description

Basic associative-container.

Definition at line 103 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.184 `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >` Class Template Reference

An abstract basic hash-based associative container.

Inheritance diagram for `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

5.184.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn,
typename Resize_Policy, bool Store_Hash, typename Tag, typename Policy_TL,
typename Allocator> class __gnu_pbds::basic_hash_table< Key, Mapped,
Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_TL, Allocator >
```

An abstract basic hash-based associative container.

Definition at line 144 of file assoc_container.hpp.

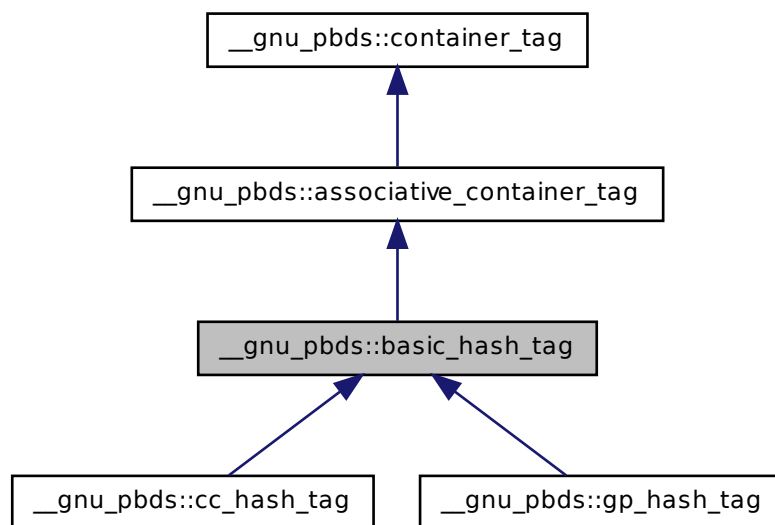
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.185 __gnu_pbds::basic_hash_tag Struct Reference

Basic hash.

Inheritance diagram for __gnu_pbds::basic_hash_tag:



5.185.1 Detailed Description

Basic hash.

Definition at line 106 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.186 `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >` Class Template Reference

An abstract basic tree-like (tree, trie) associative container.

Inheritance diagram for `__gnu_pbds::basic_tree< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**

- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef Node_Update **node_update**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

5.186.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename Allocator> class `__gnu_pbds::basic_tree`< Key, Mapped, Tag, Node_Update, Policy_Tl, Allocator >

An abstract basic tree-like (tree, trie) associative container.

Definition at line 477 of file `assoc_container.hpp`.

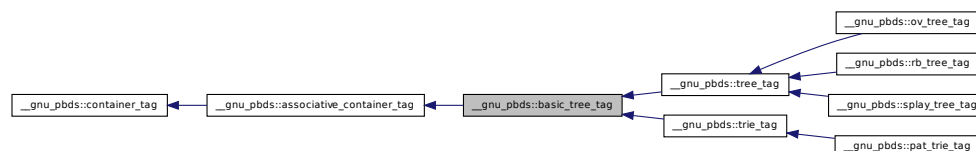
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.187 `__gnu_pbds::basic_tree_tag` Struct Reference

Basic tree.

Inheritance diagram for `__gnu_pbds::basic_tree_tag`:



5.187.1 Detailed Description

Basic tree.

Definition at line 115 of file tag_and_trait.hpp.

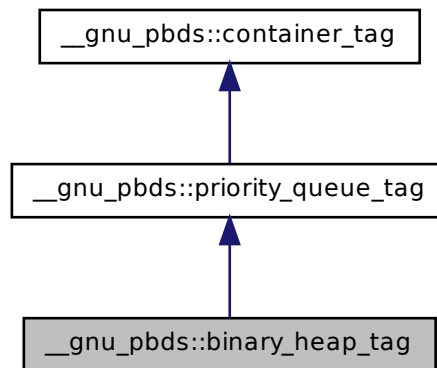
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.188 __gnu_pbds::binary_heap_tag Struct Reference

Binary-heap (array-based).

Inheritance diagram for __gnu_pbds::binary_heap_tag:



5.188.1 Detailed Description

Binary-heap (array-based).

Definition at line 151 of file tag_and_trait.hpp.

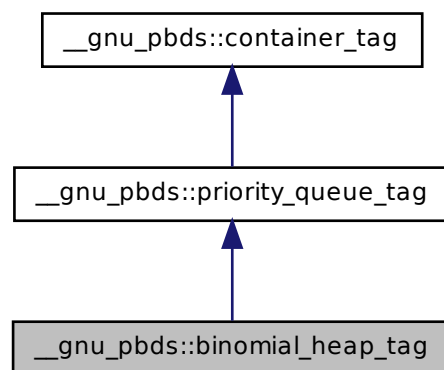
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.189 __gnu_pbds::binomial_heap_tag Struct Reference

Binomial-heap.

Inheritance diagram for __gnu_pbds::binomial_heap_tag:



5.189.1 Detailed Description

Binomial-heap.

Definition at line 145 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.190 `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >` Class Template Reference

A concrete collision-chaining hash-based associative container.

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef [cc_hash_tag](#) **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**

5.190 `__gnu_pbds::cc_hash_table`< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Allocator > Class Template

Reference

1261

- `typedef Resize_Policy` **resize_policy**
- `typedef allocator_type::size_type` **size_type**
- `typedef allocator_type::template rebind< value_type >::other` **value_rebind**
- `typedef base_type::value_type` **value_type**

Public Member Functions

- **cc_hash_table** (const hash_fn &h)
- **cc_hash_table** (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- `template<typename It >`
cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- **cc_hash_table** (const [cc_hash_table](#) &other)
- `template<typename It >`
cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- `template<typename It >`
cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- **cc_hash_table** (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- `template<typename It >`
cc_hash_table (It first, It last, const hash_fn &h)
- `template<typename It >`
cc_hash_table (It first, It last)
- **cc_hash_table** (const hash_fn &h, const eq_fn &e)
- [cc_hash_table](#) & **operator=** (const [cc_hash_table](#) &other)
- void **swap** ([cc_hash_table](#) &other)

5.190.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = type-
name detail::default_hash_fn<Key>::type, typename Eq_Fn = typename
detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_
comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash,
typename Allocator = std::allocator<char>>> class __gnu_pbds::cc_hash_table<
Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, Al-
locator >
```

A concrete collision-chaining hash-based associative container.

Definition at line 180 of file `assoc_container.hpp`.

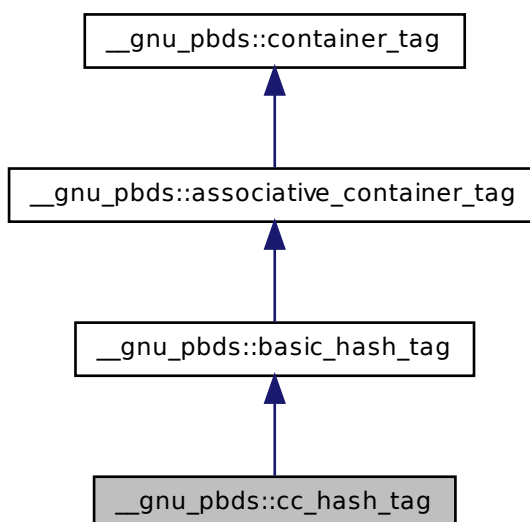
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.191 __gnu_pbds::cc_hash_tag Struct Reference

Collision-chaining hash.

Inheritance diagram for __gnu_pbds::cc_hash_tag:



5.191.1 Detailed Description

Collision-chaining hash.

Definition at line 109 of file `tag_and_trait.hpp`.

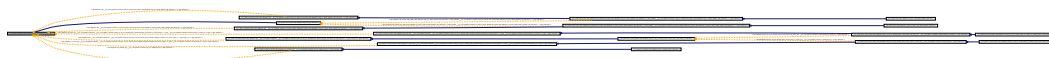
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.192 `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >` Class Template Reference

An abstract basic associative container.

Inheritance diagram for `__gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

5.192.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Policy_Tl, typename Allocator> class __gnu_pbds::container_base< Key, Mapped, Tag, Policy_Tl, Allocator >

An abstract basic associative container.

Definition at line 77 of file `assoc_container.hpp`.

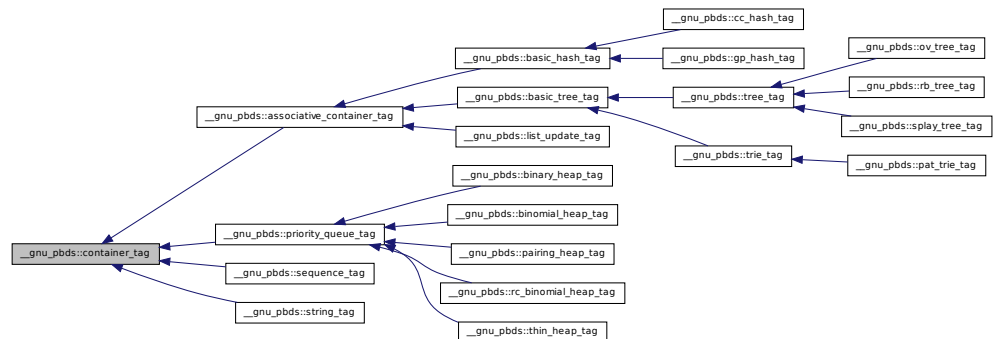
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.193 __gnu_pbds::container_tag Struct Reference

Base data structure tag.

Inheritance diagram for __gnu_pbds::container_tag:



5.193.1 Detailed Description

Base data structure tag.

Definition at line 93 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.194 `__gnu_pbds::container_traits< Cntnr >` Struct Template Reference

[container_traits](#)

Inherits `container_traits_base< Cntnr::container_category >`.

Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `container_traits_base< container_category >` **base_type**
- typedef `Cntnr::container_category` **container_category**
- typedef `Cntnr` **container_type**
- typedef `base_type::invalidation_guarantee` **invalidation_guarantee**

5.194.1 Detailed Description

`template<typename Cntnr> struct __gnu_pbds::container_traits< Cntnr >`

[container_traits](#)

Definition at line 346 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.195 `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >` Struct Template Reference

Public Types

- typedef `mapped_type_allocator::const_pointer` **const_mapped_pointer**
- typedef `mapped_type_allocator::const_reference` **const_mapped_reference**
- typedef `value_type_allocator::const_pointer` **const_pointer**
- typedef `value_type_allocator::const_reference` **const_reference**
- typedef `mapped_type_allocator::pointer` **mapped_pointer**
- typedef `mapped_type_allocator::reference` **mapped_reference**
- typedef `mapped_type_allocator::value_type` **mapped_type**

- typedef Allocator::template rebind< Mapped >::other **mapped_type_allocator**
- typedef value_type_allocator::pointer **pointer**
- typedef value_type_allocator::reference **reference**
- typedef value_type_allocator::value_type **value_type**
- typedef Allocator::template rebind< [std::pair](#)< const Key, Mapped > >::other **value_type_allocator**

5.195.1 Detailed Description

template<typename Key, typename Mapped, typename Allocator> struct `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, false >`

Specialization of `value_type_base` for the case where the hash value is not stored alongside each value.

Definition at line 61 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic_types.hpp](#)

5.196 `__gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >` Struct Template Reference

Public Types

- typedef mapped_type_allocator::const_pointer **const_mapped_pointer**
- typedef mapped_type_allocator::const_reference **const_mapped_reference**
- typedef value_type_allocator::const_pointer **const_pointer**
- typedef value_type_allocator::const_reference **const_reference**
- typedef mapped_type_allocator::pointer **mapped_pointer**
- typedef mapped_type_allocator::reference **mapped_reference**
- typedef mapped_type_allocator::value_type **mapped_type**
- typedef Allocator::template rebind< Mapped >::other **mapped_type_allocator**
- typedef value_type_allocator::pointer **pointer**
- typedef value_type_allocator::reference **reference**
- typedef value_type_allocator::value_type **value_type**
- typedef Allocator::template rebind< [std::pair](#)< const Key, Mapped > >::other **value_type_allocator**

5.196.1 Detailed Description

`template<typename Key, typename Mapped, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, Mapped, Allocator, true >`

Specialization of `value_type_base` for the case where the hash value is stored alongside each value.

Definition at line 88 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic_types.hpp](#)

5.197 `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >` Struct Template Reference

Public Types

- `typedef mapped_type_allocator::const_pointer const_mapped_pointer`
- `typedef mapped_type_allocator::const_reference const_mapped_reference`
- `typedef value_type_allocator::const_pointer const_pointer`
- `typedef value_type_allocator::const_reference const_reference`
- `typedef mapped_type_allocator::pointer mapped_pointer`
- `typedef mapped_type_allocator::reference mapped_reference`
- `typedef mapped_type_allocator::value_type mapped_type`
- `typedef Allocator::template rebind< null_mapped_type >::other mapped_type_allocator`
- `typedef value_type_allocator::pointer pointer`
- `typedef value_type_allocator::reference reference`
- `typedef Key value_type`
- `typedef Allocator::template rebind< value_type >::other value_type_allocator`

Static Public Attributes

- static [null_mapped_type](#) `s_null_mapped`

5.197.1 Detailed Description

```
template<typename Key, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, false >
```

Specialization of `value_type_base` for the case where the hash value is not stored alongside each value.

Definition at line 122 of file `basic_types.hpp`.

The documentation for this struct was generated from the following file:

- [basic_types.hpp](#)

5.198 `__gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >` Struct Template Reference

Public Types

- typedef `mapped_type_allocator::const_pointer` **const_mapped_pointer**
- typedef `mapped_type_allocator::const_reference` **const_mapped_reference**
- typedef `value_type_allocator::const_pointer` **const_pointer**
- typedef `value_type_allocator::const_reference` **const_reference**
- typedef `mapped_type_allocator::pointer` **mapped_pointer**
- typedef `mapped_type_allocator::reference` **mapped_reference**
- typedef `mapped_type_allocator::value_type` **mapped_type**
- typedef `Allocator::template rebind< null_mapped_type >::other` **mapped_type_allocator**
- typedef `value_type_allocator::pointer` **pointer**
- typedef `value_type_allocator::reference` **reference**
- typedef `Key` **value_type**
- typedef `Allocator::template rebind< value_type >::other` **value_type_allocator**

Static Public Attributes

- static [null_mapped_type](#) **s_null_mapped**

5.199 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class
Template Reference 1269
5.198.1 Detailed Description

```
template<typename Key, typename Allocator> struct __gnu_pbds::detail::value_type_base< Key, null_mapped_type, Allocator, true >
```

Specialization of value_type_base for the case where the hash value is stored alongside each value.

Definition at line 164 of file basic_types.hpp.

The documentation for this struct was generated from the following file:

- [basic_types.hpp](#)

5.199 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator > Class
Template Reference

A concrete general-probing hash-based associative container.

Inheritance diagram for __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >:



Public Types

- typedef Allocator **allocator_type**
- typedef Comb_Probe_Fn **comb_probe_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef [gp_hash_tag](#) **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**

- typedef Hash_Fn **hash_fn**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef Probe_Fn **probe_fn**
- typedef value_rebind::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

Public Member Functions

- **gp_hash_table** (const hash_fn &h)
- **gp_hash_table** (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h)
- **gp_hash_table** (const [gp_hash_table](#) &other)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- template<typename It >
gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- **gp_hash_table** (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- template<typename It >
gp_hash_table (It first, It last)

- **gp_hash_table** (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- **gp_hash_table** (const hash_fn &h, const eq_fn &e)
- **gp_hash_table** & **operator=** (const [gp_hash_table](#) &other)
- void **swap** ([gp_hash_table](#) &other)

5.199.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = type-
name detail::default_hash_fn<Key>::type, typename Eq_Fn = type-
name detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename
detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy
= typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_
Hash = detail::default_store_hash, typename Allocator = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_
Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, Allocator >
```

A concrete general-probing hash-based associative container.

Definition at line 318 of file `assoc_container.hpp`.

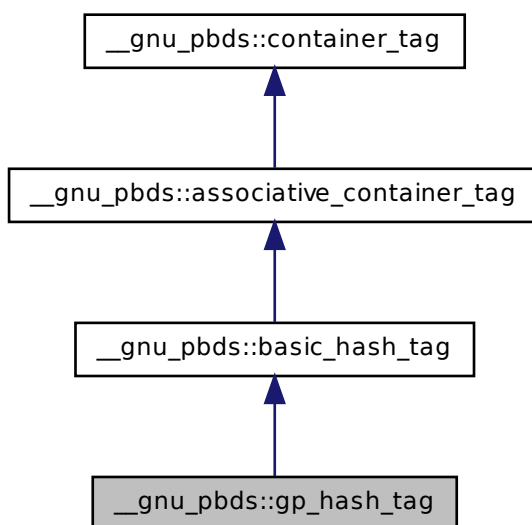
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.200 __gnu_pbds::gp_hash_tag Struct Reference

General-probing hash.

Inheritance diagram for `__gnu_pbds::gp_hash_tag`:



5.200.1 Detailed Description

General-probing hash.

Definition at line 112 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.201 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >` Class Template Reference

A list-update based associative container.

5.201 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >` Class Template Reference 1273

Inheritance diagram for `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >`:



Public Types

- typedef Allocator **allocator**
- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef [list_update_tag](#) **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef Update_Policy **update_policy**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

Public Member Functions

- `template<typename It >`
`list_update` (It first, It last)
- `list_update` (const [list_update](#) &other)
- `list_update` & `operator=` (const [list_update](#) &other)
- `void swap` ([list_update](#) &other)

5.201.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename
detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_
update_policy::type, class Allocator = std::allocator<char>> class __gnu_
pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, Allocator >
```

A list-update based associative container.

Definition at line 654 of file `assoc_container.hpp`.

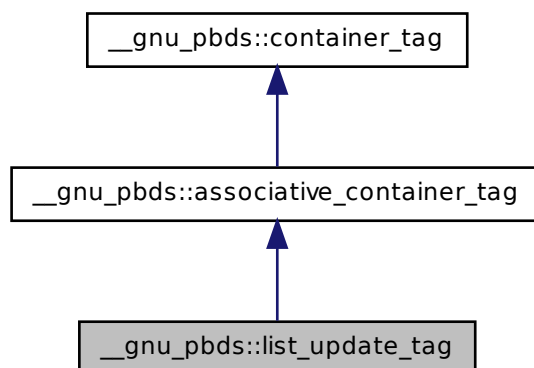
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.202 `__gnu_pbds::list_update_tag` Struct Reference

List-update.

Inheritance diagram for __gnu_pbds::list_update_tag:



5.202.1 Detailed Description

List-update.

Definition at line 136 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.203 __gnu_pbds::null_mapped_type Struct Reference

A mapped-policy indicating that an associative container is a set.

5.203.1 Detailed Description

A mapped-policy indicating that an associative container is a set.

Definition at line 89 of file `tag_and_trait.hpp`.

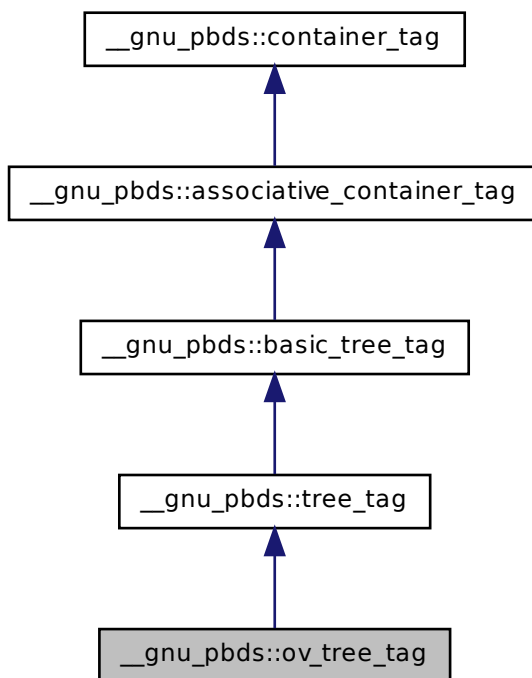
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.204 __gnu_pbds::ov_tree_tag Struct Reference

Ordered-vector tree.

Inheritance diagram for __gnu_pbds::ov_tree_tag:



5.204.1 Detailed Description

Ordered-vector tree.

Definition at line 127 of file `tag_and_trait.hpp`.

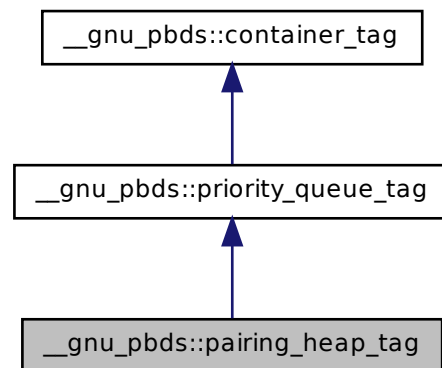
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.205 __gnu_pbds::pairing_heap_tag Struct Reference

Pairing-heap.

Inheritance diagram for __gnu_pbds::pairing_heap_tag:



5.205.1 Detailed Description

Pairing-heap.

Definition at line 142 of file `tag_and_trait.hpp`.

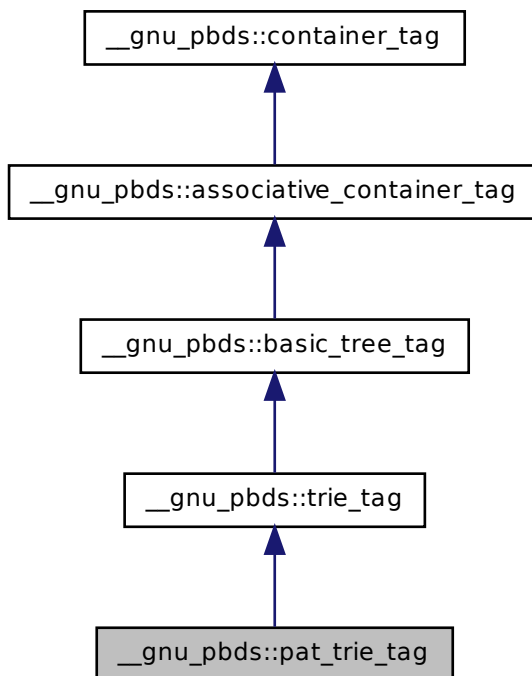
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.206 __gnu_pbds::pat_trie_tag Struct Reference

PATRICIA trie.

Inheritance diagram for `__gnu_pbds::pat_trie_tag`:



5.206.1 Detailed Description

PATRICIA trie.

Definition at line 133 of file `tag_and_trait.hpp`.

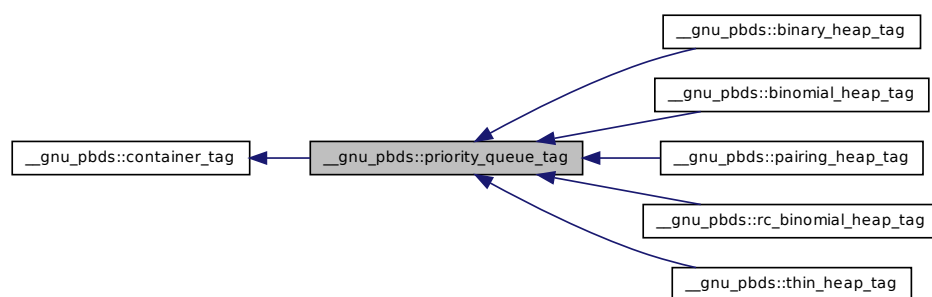
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.207 __gnu_pbds::priority_queue_tag Struct Reference

Basic priority-queue.

Inheritance diagram for __gnu_pbds::priority_queue_tag:



5.207.1 Detailed Description

Basic priority-queue.

Definition at line 139 of file `tag_and_trait.hpp`.

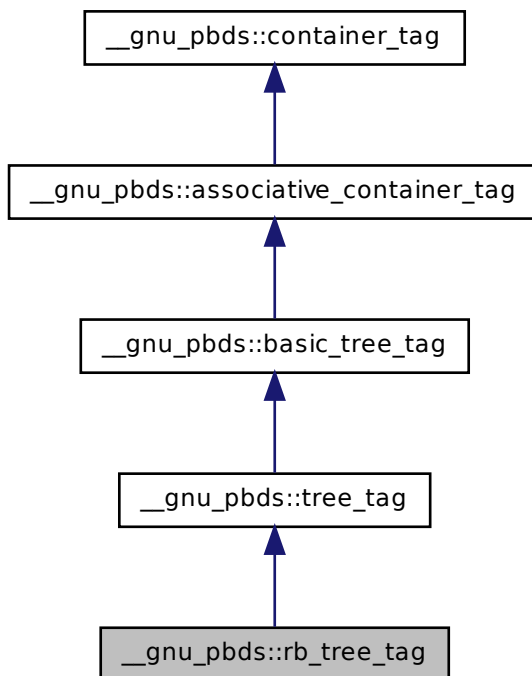
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.208 __gnu_pbds::rb_tree_tag Struct Reference

Red-black tree.

Inheritance diagram for `__gnu_pbds::rb_tree_tag`:



5.208.1 Detailed Description

Red-black tree.

Definition at line 121 of file `tag_and_trait.hpp`.

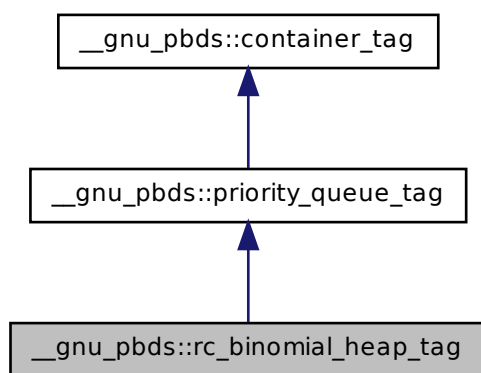
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.209 __gnu_pbds::rc_binomial_heap_tag Struct Reference

Redundant-counter binomial-heap.

Inheritance diagram for __gnu_pbds::rc_binomial_heap_tag:



5.209.1 Detailed Description

Redundant-counter binomial-heap.

Definition at line 148 of file `tag_and_trait.hpp`.

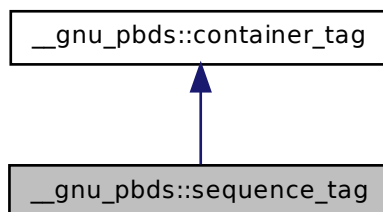
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.210 __gnu_pbds::sequence_tag Struct Reference

Basic sequence.

Inheritance diagram for `__gnu_pbds::sequence_tag`:



5.210.1 Detailed Description

Basic sequence.

Definition at line 100 of file `tag_and_trait.hpp`.

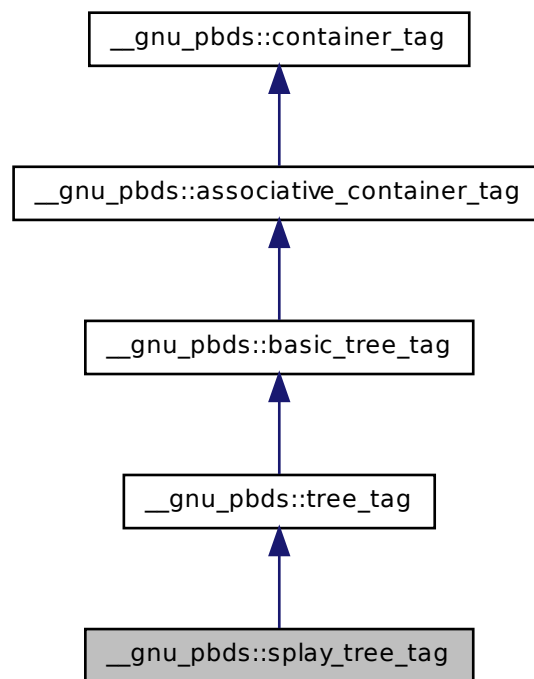
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.211 `__gnu_pbds::splay_tree_tag` Struct Reference

Splay tree.

Inheritance diagram for __gnu_pbds::splay_tree_tag:



5.211.1 Detailed Description

Splay tree.

Definition at line 124 of file `tag_and_trait.hpp`.

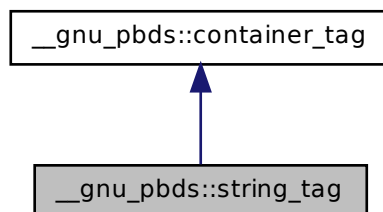
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.212 __gnu_pbds::string_tag Struct Reference

Basic string container, inclusive of strings, ropes, etc.

Inheritance diagram for `__gnu_pbds::string_tag`:



5.212.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

Definition at line 97 of file `tag_and_trait.hpp`.

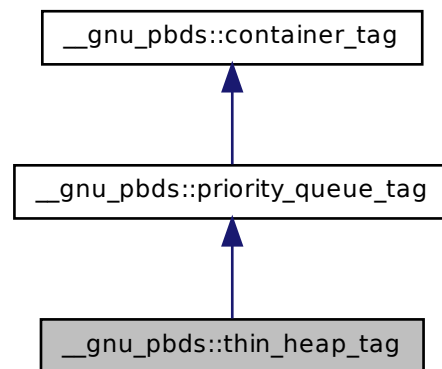
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.213 `__gnu_pbds::thin_heap_tag` Struct Reference

Thin heap.

Inheritance diagram for __gnu_pbds::thin_heap_tag:



5.213.1 Detailed Description

Thin heap.

Definition at line 154 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.214 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator > Class Template Reference

A concrete basic tree-based associative container.

Inheritance diagram for __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >:



Public Types

- typedef Allocator **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**
- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Tag, Allocator >::node_update **node_update**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

Public Member Functions

- **tree** (const cmp_fn &c)
- template<typename It >
 tree (It first, It last, const cmp_fn &c)
- **tree** (const [tree](#) &other)
- template<typename It >
 tree (It first, It last)
- [tree](#) & **operator=** (const [tree](#) &other)
- void **swap** ([tree](#) &other)

5.214.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn =
std::less<Key>, typename Tag = rb_tree_tag, template< typename Const_
Node_Iterator, typename Node_Iterator, typename Cmp_Fn_, typename
Allocator_ > class Node_Update = __gnu_pbds::null_tree_node_update, type-
name Allocator = std::allocator<char>>> class __gnu_pbds::tree< Key, Mapped,
Cmp_Fn, Tag, Node_Update, Allocator >
```

A concrete basic tree-based associative container.

Definition at line 510 of file `assoc_container.hpp`.

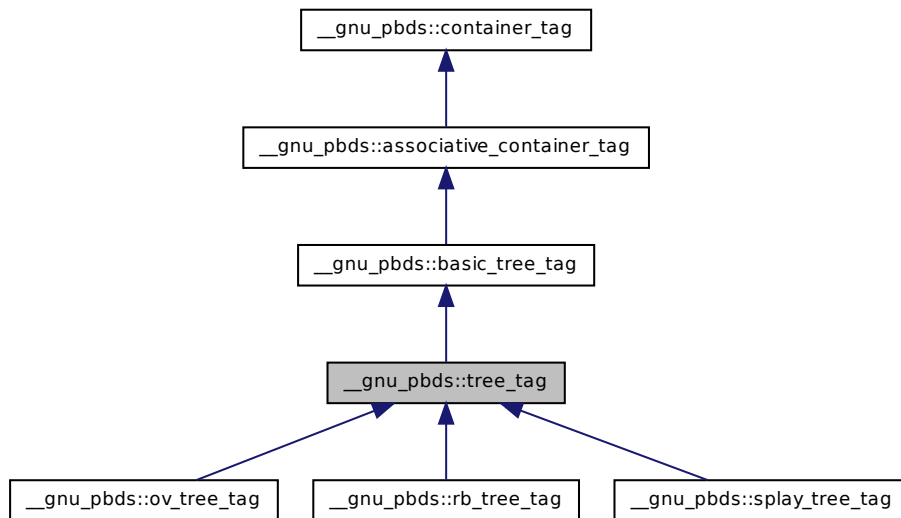
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.215 __gnu_pbds::tree_tag Struct Reference

tree.

Inheritance diagram for __gnu_pbds::tree_tag:



5.215.1 Detailed Description

tree.

Definition at line 118 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.216 `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >` Class Template Reference

A concrete basic trie-based associative container.

Inheritance diagram for `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`:



Public Types

- typedef Allocator **allocator_type**
- typedef base_type::const_iterator **const_iterator**
- typedef key_rebind::const_pointer **const_key_pointer**
- typedef key_rebind::const_reference **const_key_reference**
- typedef mapped_rebind::const_pointer **const_mapped_pointer**
- typedef mapped_rebind::const_reference **const_mapped_reference**
- typedef base_type::const_point_iterator **const_point_iterator**
- typedef value_rebind::const_pointer **const_pointer**
- typedef value_rebind::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef E_Access_Traits **e_access_traits**
- typedef base_type::iterator **iterator**
- typedef key_rebind::pointer **key_pointer**
- typedef allocator_type::template rebind< key_type >::other **key_rebind**
- typedef key_rebind::reference **key_reference**
- typedef allocator_type::template rebind< Key >::other::value_type **key_type**
- typedef mapped_rebind::pointer **mapped_pointer**

- typedef allocator_type::template rebind< mapped_type >::other **mapped_rebind**
- typedef mapped_rebind::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef detail::trie_traits< Key, Mapped, E_Access_Traits, Node_Update, Tag, Allocator >::node_update **node_update**
- typedef base_type::point_iterator **point_iterator**
- typedef value_rebind::pointer **pointer**
- typedef value_rebind::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef allocator_type::template rebind< value_type >::other **value_rebind**
- typedef base_type::value_type **value_type**

Public Member Functions

- **trie** (const e_access_traits &t)
- template<typename It >
 trie (It first, It last, const e_access_traits &t)
- **trie** (const [trie](#) &other)
- template<typename It >
 trie (It first, It last)
- [trie](#) & **operator=** (const [trie](#) &other)
- void **swap** ([trie](#) &other)

5.216.1 Detailed Description

template<typename Key, typename Mapped, typename E_Access_Traits = typename detail::default_trie_e_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Const_Node_Iterator, typename Node_Iterator, typename E_Access_Traits_, typename Allocator_ > class Node_Update = null_trie_node_update, typename Allocator = std::allocator<char>> class __gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >

A concrete basic trie-based associative container.

Definition at line 586 of file `assoc_container.hpp`.

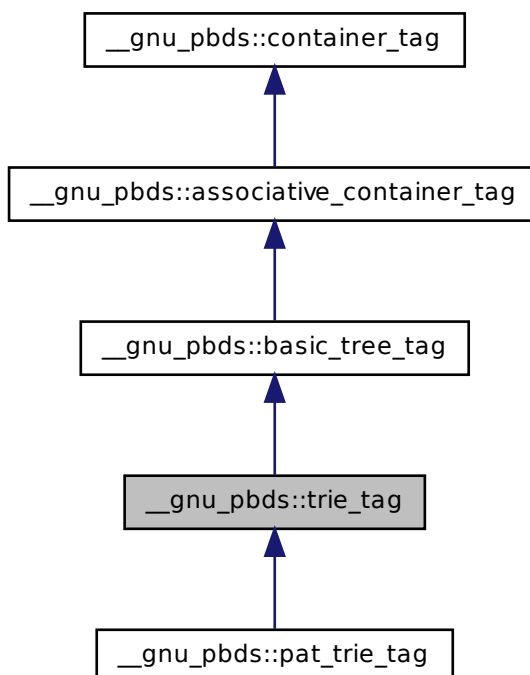
The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

5.217 `__gnu_pbds::trie_tag` Struct Reference

trie.

Inheritance diagram for `__gnu_pbds::trie_tag`:



5.217.1 Detailed Description

trie.

Definition at line 130 of file `tag_and_trait.hpp`.

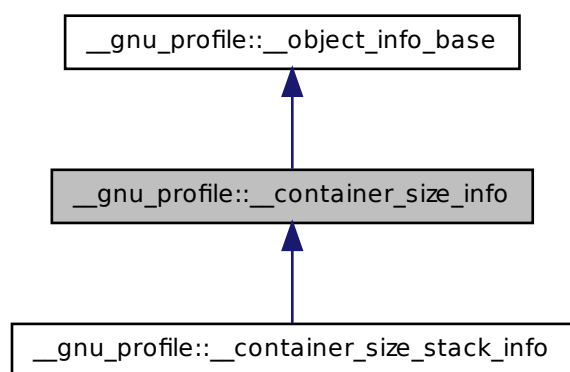
The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

5.218 __gnu_profile::__container_size_info Class Reference

A container size instrumentation line in the object table.

Inheritance diagram for __gnu_profile::__container_size_info:



Public Member Functions

- **__container_size_info** (const [__container_size_info](#) &__o)
- **__container_size_info** (__stack_t __stack, std::size_t __num)
- [std::string](#) **__advice** () const
- void **__destruct** (std::size_t __num, std::size_t __inum)
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const [__container_size_info](#) &__o)
- void **__resize** (std::size_t __from, std::size_t __to)
- float **__resize_cost** (std::size_t __from, std::size_t)
- **__stack_t** **__stack** () const
- void **__write** (FILE *__f) const

Protected Attributes

- **__stack_t** **_M_stack**

- `bool _M_valid`

5.218.1 Detailed Description

A container size instrumentation line in the object table.

Definition at line 49 of file `profiler_container_size.h`.

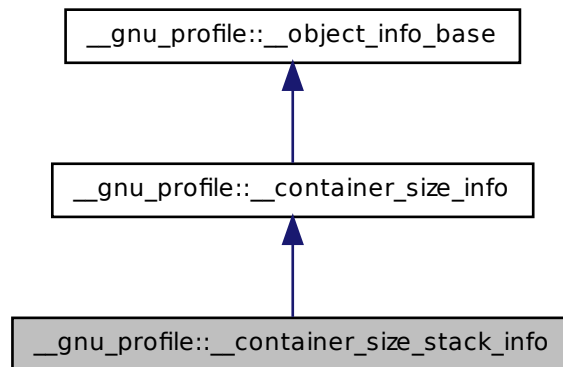
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

5.219 `__gnu_profile::__container_size_stack_info` Class Reference

A container size instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__container_size_stack_info`:



Public Member Functions

- `__container_size_stack_info` (const `__container_size_info` &`_o`)
- `std::string __advice` () const

- void `__destruct` (std::size_t __num, std::size_t __inum)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const [__container_size_info](#) &__o)
- void `__resize` (std::size_t __from, std::size_t __to)
- float `__resize_cost` (std::size_t __from, std::size_t)
- `__stack_t` `__stack` () const
- void `__write` (FILE *__f) const

Protected Attributes

- `__stack_t` `_M_stack`
- bool `_M_valid`

5.219.1 Detailed Description

A container size instrumentation line in the stack table.

Definition at line 161 of file `profiler_container_size.h`.

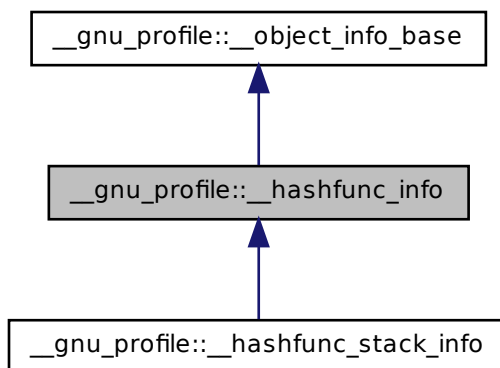
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

5.220 `__gnu_profile::__hashfunc_info` Class Reference

A hash performance instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__hashfunc_info`:



Public Member Functions

- `__hashfunc_info` (const [__hashfunc_info](#) &__o)
- `__hashfunc_info` (__stack_t __stack)
- [std::string](#) `__advice` () const
- void `__destruct` (std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const [__hashfunc_info](#) &__o)
- `__stack_t` `__stack` () const
- void `__write` (FILE *__f) const

Protected Attributes

- `__stack_t` `_M_stack`
- bool `_M_valid`

5.220.1 Detailed Description

A hash performance instrumentation line in the object table.

Definition at line 47 of file `profiler_hash_func.h`.

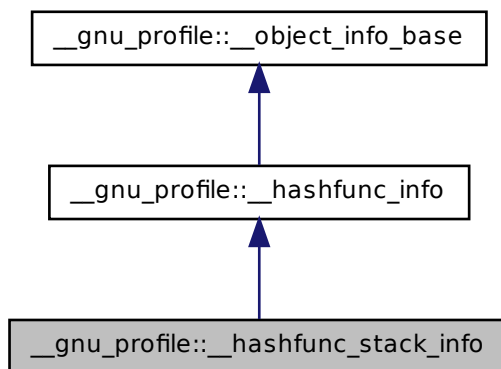
The documentation for this class was generated from the following file:

- `profiler_hash_func.h`

5.221 `__gnu_profile::__hashfunc_stack_info` Class Reference

A hash performance instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__hashfunc_stack_info`:



Public Member Functions

- `__hashfunc_stack_info` (const `__hashfunc_info` &`__o`)
- `std::string __advice` () const
- void `__destruct` (std::size_t `__chain`, std::size_t `__accesses`, std::size_t `__hops`)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__hashfunc_info` &`__o`)
- `__stack_t __stack` () const
- void `__write` (FILE *`__f`) const

Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

5.221.1 Detailed Description

A hash performance instrumentation line in the stack table.

Definition at line 102 of file `profiler_hash_func.h`.

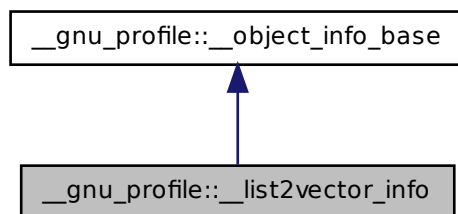
The documentation for this class was generated from the following file:

- `profiler_hash_func.h`

5.222 `__gnu_profile::__list2vector_info` Class Reference

A list-to-vector instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__list2vector_info`:



Public Member Functions

- `__list2vector_info (__stack_t __stack)`
- `__list2vector_info (const __list2vector_info &__o)`
- `std::string __advice () const`
- `bool __is_valid () const`

- `bool __is_valid ()`
- `std::size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const __list2vector_info &__o)`
- `void __opr_insert (std::size_t __shift, std::size_t __size)`
- `void __opr_iterate (std::size_t __num)`
- `std::size_t __resize ()`
- `void __resize (std::size_t __from, std::size_t)`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `std::size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

Protected Attributes

- `__stack_t _M_stack`

5.222.1 Detailed Description

A list-to-vector instrumentation line in the object table.

Definition at line 49 of file `profiler_list_to_vector.h`.

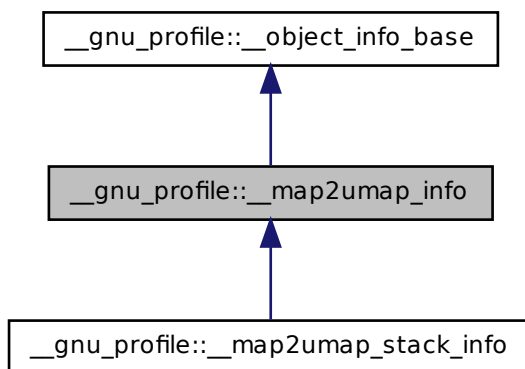
The documentation for this class was generated from the following file:

- [profiler_list_to_vector.h](#)

5.223 `__gnu_profile::__map2umap_info` Class Reference

A map-to-unordered_map instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__map2umap_info`:



Public Member Functions

- `__map2umap_info` (`__stack_t __stack`)
- `__map2umap_info` (`const __map2umap_info &__o`)
- `std::string __advice` () `const`
- `bool __is_valid` () `const`
- `float __magnitude` () `const`
- `void __merge` (`const __map2umap_info &__o`)
- `void __record_erase` (`std::size_t __size`, `std::size_t __count`)
- `void __record_find` (`std::size_t __size`)
- `void __record_insert` (`std::size_t __size`, `std::size_t __count`)
- `void __record_invalidate` ()
- `void __record_iterate` (`std::size_t __count`)
- `__stack_t __stack` () `const`
- `void __write` (`FILE *__f`) `const`

Protected Attributes

- `__stack_t _M_stack`

5.223.1 Detailed Description

A map-to-unordered_map instrumentation line in the object table.

Definition at line 73 of file profiler_map_to_unordered_map.h.

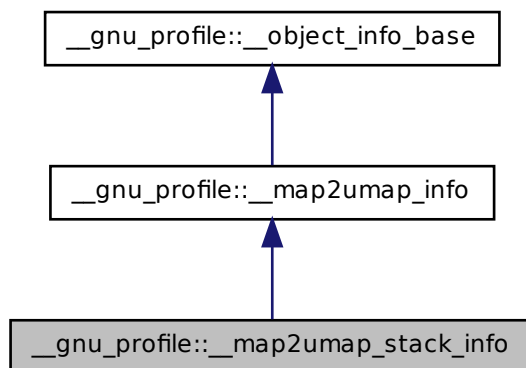
The documentation for this class was generated from the following file:

- [profiler_map_to_unordered_map.h](#)

5.224 __gnu_profile::__map2umap_stack_info Class Reference

A map-to-unordered_map instrumentation line in the stack table.

Inheritance diagram for __gnu_profile::__map2umap_stack_info:



Public Member Functions

- **__map2umap_stack_info** (const [__map2umap_info](#) &__o)
- [std::string](#) **__advice** () const
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const [__map2umap_info](#) &__o)

- void **__record_erase** (std::size_t __size, std::size_t __count)
- void **__record_find** (std::size_t __size)
- void **__record_insert** (std::size_t __size, std::size_t __count)
- void **__record_invalidate** ()
- void **__record_iterate** (std::size_t __count)
- __stack_t **__stack** () const
- void **__write** (FILE *__f) const

Protected Attributes

- __stack_t **_M_stack**

5.224.1 Detailed Description

A map-to-unordered_map instrumentation line in the stack table.

Definition at line 177 of file profiler_map_to_unordered_map.h.

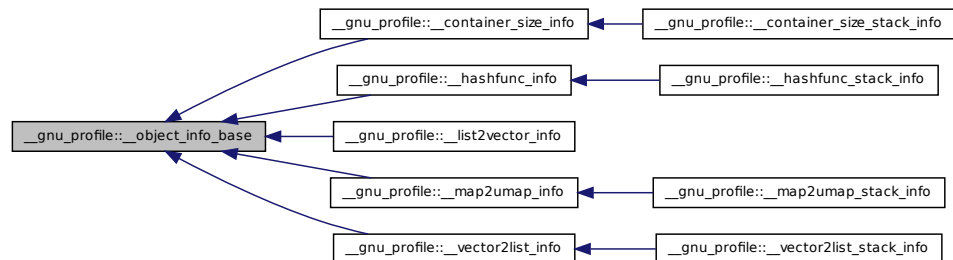
The documentation for this class was generated from the following file:

- [profiler_map_to_unordered_map.h](#)

5.225 __gnu_profile::__object_info_base Class Reference

Base class for a line in the object table.

Inheritance diagram for __gnu_profile::__object_info_base:



Public Member Functions

- `__object_info_base` (`__stack_t __stack`)
- `__object_info_base` (`const __object_info_base &__o`)
- `bool __is_valid` () `const`
- `__stack_t __stack` () `const`
- `virtual void __write` (`FILE *__f`) `const =0`

Protected Attributes

- `__stack_t _M_stack`
- `bool _M_valid`

5.225.1 Detailed Description

Base class for a line in the object table.

Definition at line 130 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler_node.h](#)

5.226 `__gnu_profile::__reentrance_guard` Struct Reference

Reentrance guard.

Static Public Member Functions

- `static bool __get_in` ()
- `static bool & __inside` ()

5.226.1 Detailed Description

Reentrance guard. Mechanism to protect all `__gnu_profile` operations against recursion, multithreaded and exception reentrance.

Definition at line 65 of file `profiler.h`.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

5.227 `__gnu_profile::__stack_hash` Class Reference

Hash function for summary trace using call stack as index.

Public Member Functions

- `std::size_t operator() (__stack_t __s) const`
- `bool operator() (__stack_t __stack1, __stack_t __stack2) const`

5.227.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 96 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler_node.h](#)

5.228 `__gnu_profile::__stack_info_base< __object_info >` Class Template Reference

Base class for a line in the stack table.

Public Member Functions

- `__stack_info_base (const __object_info &__info)=0`
- `virtual const char * __get_id () const =0`
- `virtual float __magnitude () const =0`
- `void __merge (const __object_info &__info)=0`

5.228.1 Detailed Description

`template<typename __object_info> class __gnu_profile::__stack_info_base< __object_info >`

Base class for a line in the stack table.

Definition at line 161 of file `profiler_node.h`.

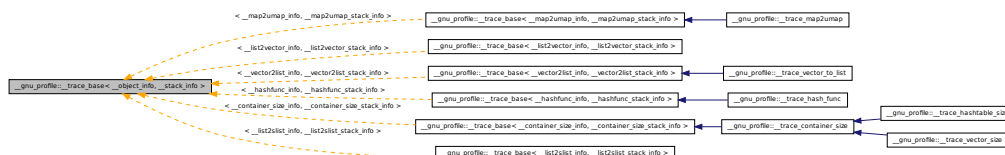
The documentation for this class was generated from the following file:

- [profiler_node.h](#)

5.229 `__gnu_profile::__trace_base< __object_info, __stack_info >` Class Template Reference

Base class for all trace producers.

Inheritance diagram for `__gnu_profile::__trace_base< __object_info, __stack_info >`:



Public Member Functions

- void `__add_object` (`__object_t` object, `__object_info` __info)
- void `__collect_warnings` (`__warning_vector_t` &__warnings)
- `__object_info` * `__get_object_info` (`__object_t` __object)
- void `__retire_object` (`__object_t` __object)
- void `__write` (FILE * __f)

Protected Attributes

- const char * `__id`

5.229.1 Detailed Description

`template<typename __object_info, typename __stack_info> class __gnu_profile::__trace_base< __object_info, __stack_info >`

Base class for all trace producers.

Definition at line 190 of file `profiler_trace.h`.

The documentation for this class was generated from the following file:

- [profiler_trace.h](#)

5.230 `__gnu_profile::__trace_container_size` Class Reference

Container size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_container_size`:



Public Member Functions

- void **__add_object** (`__object_t` object, `__container_size_info` info)
- void **__collect_warnings** (`__warning_vector_t` &__warnings)
- void **__construct** (const void *__obj, `std::size_t` __inum)
- void **__destruct** (const void *__obj, `std::size_t` __num, `std::size_t` __inum)
- `__container_size_info` * **__get_object_info** (`__object_t` __object)
- void **__insert** (const `__object_t` __obj, `__stack_t` __stack, `std::size_t` __num)
- void **__resize** (const void *__obj, int __from, int __to)
- void **__retire_object** (`__object_t` __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.230.1 Detailed Description

Container size instrumentation trace producer.

Definition at line 171 of file `profiler_container_size.h`.

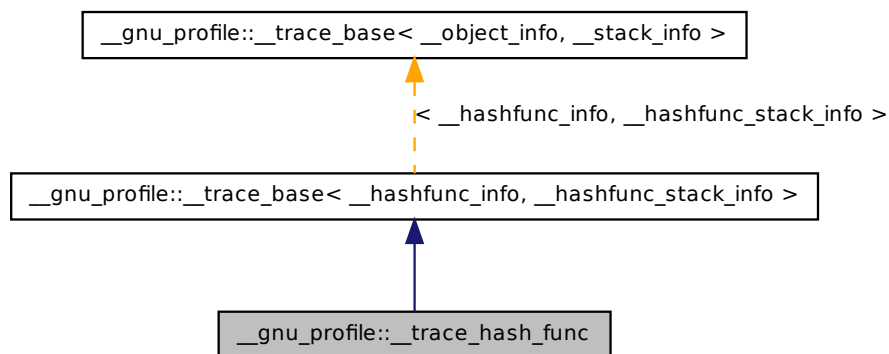
The documentation for this class was generated from the following file:

- `profiler_container_size.h`

5.231 `__gnu_profile::__trace_hash_func` Class Reference

Hash performance instrumentation producer.

Inheritance diagram for __gnu_profile::__trace_hash_func:



Public Member Functions

- void **__add_object** (__object_t object, __hashfunc_info __info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__destruct** (const void *__obj, std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- [__hashfunc_info](#) * **__get_object_info** (__object_t __object)
- void **__insert** (__object_t __obj, __stack_t __stack)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.231.1 Detailed Description

Hash performance instrumentation producer.

Definition at line 112 of file profiler_hash_func.h.

The documentation for this class was generated from the following file:

- profiler_hash_func.h

5.232 `__gnu_profile::__trace_hashtable_size` Class Reference

Hashtable size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_hashtable_size`:



Public Member Functions

- void **__add_object** (`__object_t` object, `__container_size_info` info)
- void **__collect_warnings** (`__warning_vector_t` &__warnings)
- void **__construct** (const void *__obj, `std::size_t` __inum)
- void **__destruct** (const void *__obj, `std::size_t` __num, `std::size_t` __inum)
- `__container_size_info` * **__get_object_info** (`__object_t` __object)
- void **__insert** (const `__object_t` __obj, `__stack_t` __stack, `std::size_t` __num)
- void **__resize** (const void *__obj, int __from, int __to)
- void **__retire_object** (`__object_t` __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.232.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 49 of file `profiler_hashtable_size.h`.

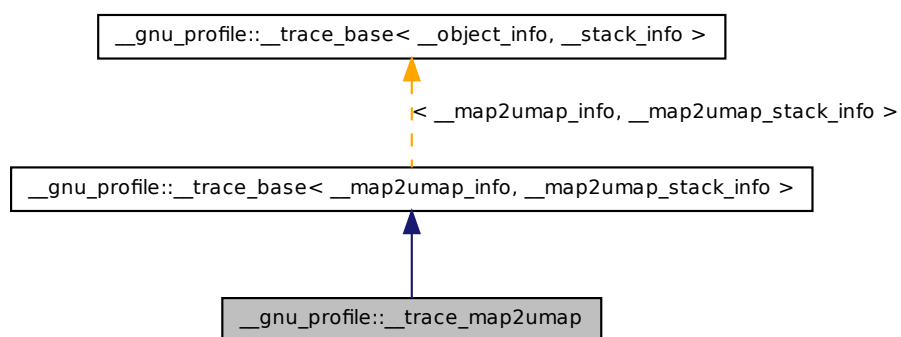
The documentation for this class was generated from the following file:

- [profiler_hashtable_size.h](#)

5.233 `__gnu_profile::__trace_map2umap` Class Reference

Map-to-unordered_map instrumentation producer.

Inheritance diagram for __gnu_profile::__trace_map2umap:



Public Member Functions

- void **__add_object** (__object_t object, __map2umap_info__info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- [__map2umap_info](#) * **__get_object_info** (__object_t __object)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.233.1 Detailed Description

Map-to-unordered_map instrumentation producer.

Definition at line 186 of file profiler_map_to_unordered_map.h.

The documentation for this class was generated from the following file:

- [profiler_map_to_unordered_map.h](#)

5.234 `__gnu_profile::__trace_vector_size` Class Reference

Hashtable size instrumentation trace producer.

Inheritance diagram for `__gnu_profile::__trace_vector_size`:



Public Member Functions

- void **__add_object** (`__object_t` object, `__container_size_info` __info)
- void **__collect_warnings** (`__warning_vector_t` &__warnings)
- void **__construct** (const void *__obj, `std::size_t` __inum)
- void **__destruct** (const void *__obj, `std::size_t` __num, `std::size_t` __inum)
- `__container_size_info` * **__get_object_info** (`__object_t` __object)
- void **__insert** (const `__object_t` __obj, `__stack_t` __stack, `std::size_t` __num)
- void **__resize** (const void *__obj, int __from, int __to)
- void **__retire_object** (`__object_t` __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.234.1 Detailed Description

Hashtable size instrumentation trace producer.

Definition at line 49 of file `profiler_vector_size.h`.

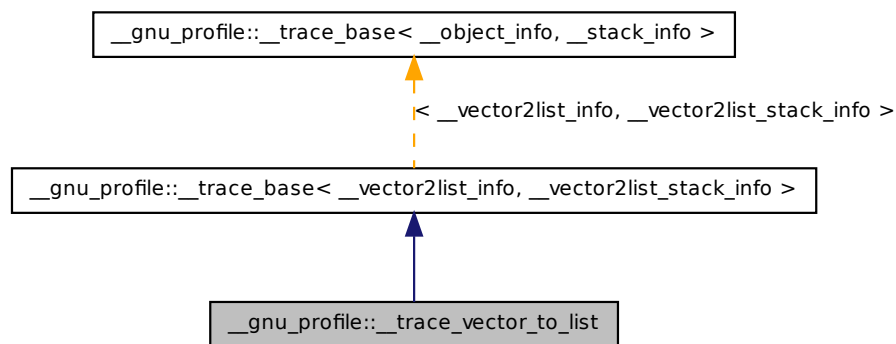
The documentation for this class was generated from the following file:

- [profiler_vector_size.h](#)

5.235 `__gnu_profile::__trace_vector_to_list` Class Reference

Vector-to-list instrumentation producer.

Inheritance diagram for __gnu_profile::__trace_vector_to_list:



Public Member Functions

- void **__add_object** (__object_t object, __vector2list_info __info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__destruct** (const void *__obj)
- [__vector2list_info](#) * **__find** (const void *__obj)
- [__vector2list_info](#) * **__get_object_info** (__object_t __object)
- void **__insert** (__object_t __obj, __stack_t __stack)
- void **__invalid_operator** (const void *__obj)
- float **__list_cost** (std::size_t __shift, std::size_t __iterate, std::size_t __resize)
- void **__opr_find** (const void *__obj, std::size_t __size)
- void **__opr_insert** (const void *__obj, std::size_t __pos, std::size_t __num)
- void **__opr_iterate** (const void *__obj, std::size_t __num)
- void **__resize** (const void *__obj, std::size_t __from, std::size_t __to)
- void **__retire_object** (__object_t __object)
- float **__vector_cost** (std::size_t __shift, std::size_t __iterate, std::size_t __resize)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

5.235.1 Detailed Description

Vector-to-list instrumentation producer.

Definition at line 165 of file profiler_vector_to_list.h.

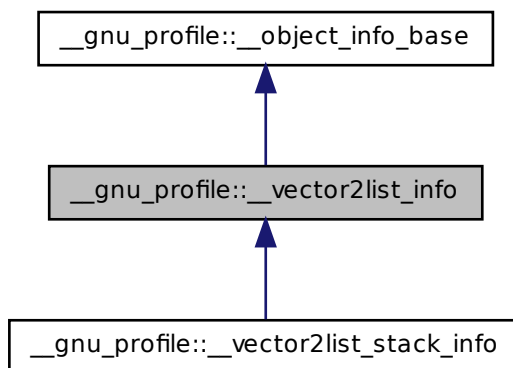
The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

5.236 `__gnu_profile::__vector2list_info` Class Reference

A vector-to-list instrumentation line in the object table.

Inheritance diagram for `__gnu_profile::__vector2list_info`:



Public Member Functions

- `__vector2list_info` (`__stack_t` `__stack`)
- `__vector2list_info` (`const` `__vector2list_info` &`__o`)
- `std::string` `__advice` () `const`
- `bool` `__is_valid` () `const`
- `bool` `__is_valid` ()

- `std::size_t __iterate ()`
- `float __list_cost ()`
- `float __magnitude () const`
- `void __merge (const __vector2list_info &__o)`
- `void __opr_find (std::size_t __size)`
- `void __opr_insert (std::size_t __pos, std::size_t __num)`
- `void __opr_iterate (std::size_t __num)`
- `void __resize (std::size_t __from, std::size_t)`
- `std::size_t __resize ()`
- `void __set_invalid ()`
- `void __set_list_cost (float __lc)`
- `void __set_vector_cost (float __vc)`
- `std::size_t __shift_count ()`
- `__stack_t __stack () const`
- `void __write (FILE *__f) const`

Protected Attributes

- `__stack_t _M_stack`

5.236.1 Detailed Description

A vector-to-list instrumentation line in the object table.

Definition at line 47 of file `profiler_vector_to_list.h`.

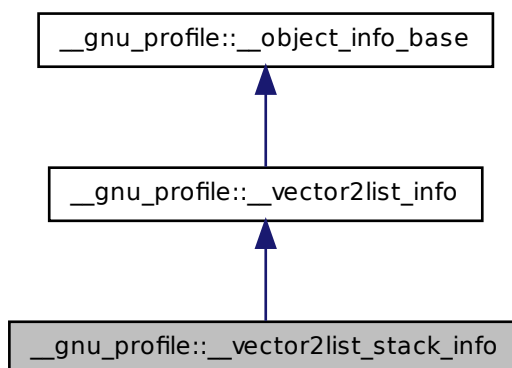
The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

5.237 `__gnu_profile::__vector2list_stack_info` Class Reference

A vector-to-list instrumentation line in the stack table.

Inheritance diagram for `__gnu_profile::__vector2list_stack_info`:



Public Member Functions

- `__vector2list_stack_info` (const `__vector2list_info` &__o)
- `std::string __advice` () const
- `bool __is_valid` ()
- `bool __is_valid` () const
- `std::size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () const
- `void __merge` (const `__vector2list_info` &__o)
- `void __opr_find` (std::size_t __size)
- `void __opr_insert` (std::size_t __pos, std::size_t __num)
- `void __opr_iterate` (std::size_t __num)
- `void __resize` (std::size_t __from, std::size_t)
- `std::size_t __resize` ()
- `void __set_invalid` ()
- `void __set_list_cost` (float __lc)
- `void __set_vector_cost` (float __vc)
- `std::size_t __shift_count` ()
- `__stack_t __stack` () const
- `void __write` (FILE *__f) const

Protected Attributes

- `__stack_t __M_stack`

5.237.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 155 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

5.238 `__gnu_profile::__warning_data` Struct Reference

Representation of a warning.

Public Member Functions

- `__warning_data` (float `__m`, `__stack_t` `__c`, const char *`__id`, const [std::string](#) &`__msg`)
- bool `operator<` (const [__warning_data](#) &`__other`) const

Public Attributes

- `__stack_t __context`
- float `__magnitude`
- const char * `__warning_id`
- [std::string](#) `__warning_message`

5.238.1 Detailed Description

Representation of a warning.

Definition at line 80 of file `profiler_trace.h`.

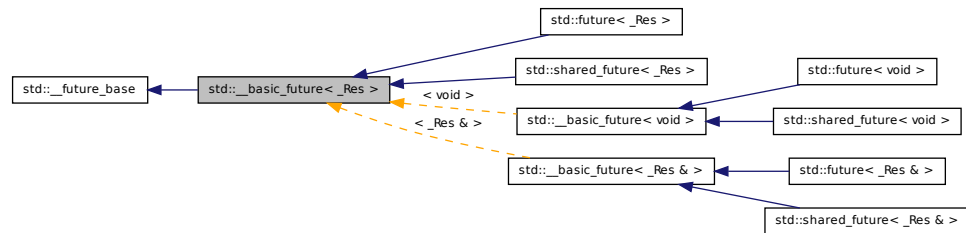
The documentation for this struct was generated from the following file:

- [profiler_trace.h](#)

5.239 `std::__basic_future< _Res >` Class Template Reference

Common implementation for future and [shared_future](#).

Inheritance diagram for `std::__basic_future< _Res >`:



Public Member Functions

- `__basic_future` (const `__basic_future` &)
- `__basic_future` & `operator=` (const `__basic_future` &)
- `bool valid` () const
- `void wait` () const
- `template<typename _Rep, typename _Period >`
`bool wait_for` (const `chrono::duration`< _Rep, _Period > &__rel) const
- `template<typename _Clock, typename _Duration >`
`bool wait_until` (const `chrono::time_point`< _Clock, _Duration > &__abs)
const

Static Public Member Functions

- `template<typename _Res, typename _Allocator >`
static `_Ptr`< `_Result_alloc`< _Res, _Allocator > >::type `_S_allocate_result`
(const _Allocator &__a)

Protected Types

- `typedef __future_base::_Result`< _Res > & `__result_type`
- `typedef shared_ptr`< _State > `__state_type`

Protected Member Functions

- `__basic_future` (const `__state_type` &__state)
- `__basic_future` (const `shared_future`< _Res > &)
- `__basic_future` (`shared_future`< _Res > &&)
- `__basic_future` (`future`< _Res > &&)
- `__result_type` `_M_get_result` ()
- void `_M_swap` (`__basic_future` &__that)

5.239.1 Detailed Description

`template<typename _Res> class std::__basic_future< _Res >`

Common implementation for future and `shared_future`.

Definition at line 475 of file future.

5.239.2 Member Function Documentation

5.239.2.1 `template<typename _Res> __result_type std::__basic_future< _Res
>::__M_get_result () [inline, protected]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 508 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

The documentation for this class was generated from the following file:

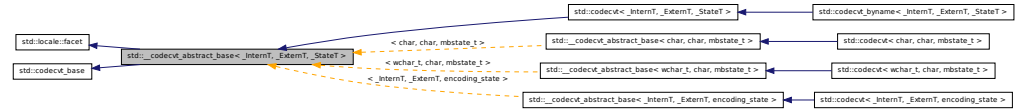
- `future`

5.240 `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >` Class Template Reference

Common base for codecvt functions.

Inheritance diagram for `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT`

>:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Protected Member Functions

- `__codecvt_abstract_base` (size_t __refs=0)
- virtual bool **do_always_noconv** () const =0 throw ()
- virtual int **do_encoding** () const =0 throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0

- virtual int **do_max_length** () const =0 throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Friends

- class **locale::_Impl**

5.240.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT> class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >

Common base for codecvt functions. This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 67 of file codecvt.h.

5.240.2 Member Function Documentation

5.240.2.1 **template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [protected, pure virtual]**

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

[out](#) for more information.

Implemented in `std::codecvt<_InternT, _ExternT, _StateT >`.

```
5.240.2.2 template<typename _InternT, typename _ExternT, typename
    _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
    _StateT >::in ( state_type & __state, const extern_type * __from,
    const extern_type * __from_end, const extern_type *& __from_next,
    intern_type * __to, intern_type * __to_end, intern_type *&
    __to_next ) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.240.2.3 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const [inline]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling [codecvt::do_out](#).

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 115 of file codecvt.h.

5.240.2.4 `template<typename _InternT, typename _ExternT, typename
_StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
_StateT >::unshift (state_type & __state, extern_type * __to,
extern_type * __to_end, extern_type *& __to_next) const
[inline]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to [in\(\)](#) had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 154 of file codecvt.h.

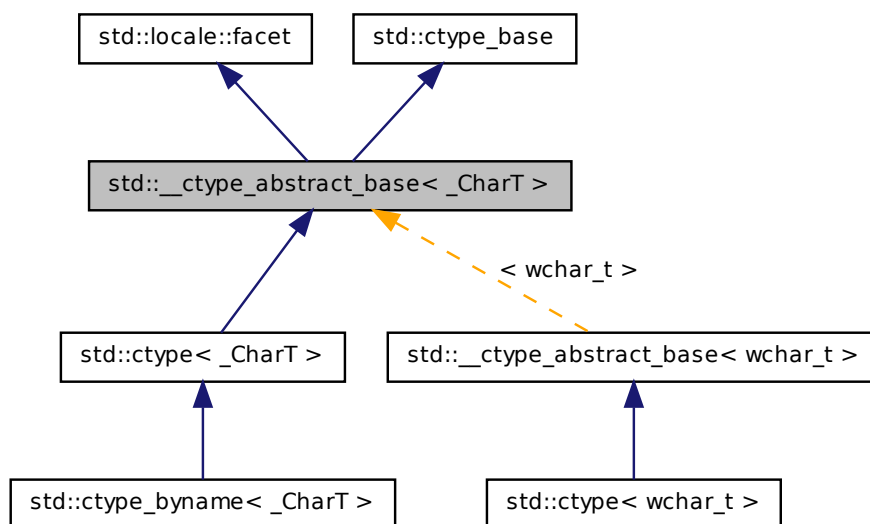
The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.241 `std::__ctype_abstract_base< _CharT >` Class Template Reference

Common base for ctype facet.

Inheritance diagram for `std::__ctype_abstract_base< _CharT >`:



Public Types

- typedef const int * **__to_type**
- typedef `_CharT` **char_type**
- typedef unsigned short **mask**

Public Member Functions

- bool **is** (mask `__m`, **char_type** `__c`) const
- const **char_type** * **is** (const **char_type** * `__lo`, const **char_type** * `__hi`, mask * `__vec`) const
- char **narrow** (**char_type** `__c`, char `__default`) const

- const [char_type](#) * [narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __default, char * __to) const
- const [char_type](#) * [scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [tolower](#) ([char_type](#) __c) const
- const [char_type](#) * [toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [toupper](#) ([char_type](#) __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, [char_type](#) * __to) const
- [char_type](#) [widen](#) (char __c) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- [__ctype_abstract_base](#) (size_t __refs=0)
- virtual const [char_type](#) * [do_is](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const =0
- virtual bool [do_is](#) (mask __m, [char_type](#) __c) const =0
- virtual char [do_narrow](#) ([char_type](#), char __default) const =0
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __default, char * __dest) const =0
- virtual const [char_type](#) * [do_scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const =0
- virtual const [char_type](#) * [do_scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const =0
- virtual [char_type](#) [do_tolower](#) ([char_type](#)) const =0
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const =0

- virtual `char_type do_toupper(char_type)` const =0
- virtual const `char_type * do_toupper(char_type *__lo, const char_type *__hi)` const =0
- virtual const `char * do_widen(const char *__lo, const char *__hi, char_type *__dest)` const =0
- virtual `char_type do_widen(char)` const =0

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale(__c_locale &__cloc)` throw ()
- static void `_S_create_c_locale(__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale(__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale()`
- static `_GLIBCXX_CONST const char * _S_get_c_name()` throw ()
- static `__c_locale _S_lc_ctype_c_locale(__c_locale __cloc, const char *__s)`

Friends

- class `locale::_Impl`

5.241.1 Detailed Description

`template<typename _CharT> class std::__ctype_abstract_base<_CharT>`

Common base for ctype facet. This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 143 of file `locale_facets.h`.

5.241.2 Member Typedef Documentation

5.241.2.1 `template<typename _CharT> typedef _CharT
std::__ctype_abstract_base<_CharT>::char_type`

Typedef for the template parameter.

Reimplemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

Definition at line 148 of file `locale_facets.h`.

5.241.3 Member Function Documentation

5.241.3.1 `template<typename _CharT> virtual bool std::__ctype_abstract_base<_CharT>::do_is (mask __m, char_type __c) const`
`[protected, pure virtual]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

c The `char_type` to find the mask of.

m The mask to compare against.

Returns

$(M \& m) \neq 0$.

Implemented in `std::ctype<_CharT>`.

5.241.3.2 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const`
`[protected, pure virtual]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Implemented in `std::ctype<_CharT>`.

5.241.3.3 `template<typename _CharT> virtual char std::__ctype_abstract_base<_CharT>::do_narrow(char_type, char __dfault) const [protected, pure virtual]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

Returns

- The converted `char`.

Implemented in `std::ctype<_CharT>`.

5.241.3.4 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_narrow(const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, pure virtual]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns*hi*.Implemented in [std::ctype<_CharT>](#).

5.241.3.5 `template<typename _CharT> virtual const char_type*
 std::__ctype_abstract_base<_CharT>::do_scan_is (mask
 __m, const char_type * __lo, const char_type * __hi) const
 [protected, pure virtual]`

Find char_type matching mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching.
[do_is\(\)](#) must always return the same result for the same input.

Parameters*m* The mask to compare against.*lo* Pointer to start of range.*hi* Pointer to end of range.**Returns**Pointer to a matching char_type if found, else *hi*.Implemented in [std::ctype<_CharT>](#).

5.241.3.6 `template<typename _CharT> virtual const char_type*
 std::__ctype_abstract_base<_CharT>::do_scan_not (mask
 __m, const char_type * __lo, const char_type * __hi) const
 [protected, pure virtual]`

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching.
[do_is\(\)](#) must always return the same result for the same input.

Parameters*m* The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implemented in `std::ctype<_CharT>`.

```
5.241.3.7 template<typename _CharT> virtual char_type
std::__ctype_abstract_base<_CharT>::do_tolower ( char_type )
const [protected, pure virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The lowercase `char_type` if convertible, else *c*.

Implemented in `std::ctype<_CharT>`.

```
5.241.3.8 template<typename _CharT> virtual const char_type*
std::__ctype_abstract_base<_CharT>::do_tolower ( char_type
* __lo, const char_type * __hi ) const [protected, pure
virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Implemented in [std::ctype<_CharT>](#).

5.241.3.9 `template<typename _CharT> virtual char_type
std::__ctype_abstract_base<_CharT>::do_toupper (char_type)
const [protected, pure virtual]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Implemented in [std::ctype<_CharT>](#).

5.241.3.10 `template<typename _CharT> virtual const char_type*
std::__ctype_abstract_base<_CharT>::do_toupper (char_type
* __lo, const char_type * __hi) const [protected, pure
virtual]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns*hi*.Implemented in `std::ctype<_CharT>`.

5.241.3.11 `template<typename _CharT> virtual char_type
std::__ctype_abstract_base<_CharT>::do_widen (char) const
[protected, pure virtual]`

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters*c* The char to convert.**Returns**

The converted char_type

Implemented in `std::ctype<_CharT>`, and `std::ctype<wchar_t>`.

5.241.3.12 `template<typename _CharT> virtual const char*
std::__ctype_abstract_base<_CharT>::do_widen (const
char * __lo, const char * __hi, char_type * __dest) const
[protected, pure virtual]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters*lo* Pointer to start range.*hi* Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Implemented in [std::ctype<_CharT>](#).

5.241.3.13 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::is (const char_type *
__lo, const char_type * __hi, mask * __vec) const [inline]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of [ctype<char_type>::do_is\(\)](#).

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Definition at line 178 of file locale_facets.h.

5.241.3.14 `template<typename _CharT> bool std::__ctype_abstract_base<
_CharT>::is (mask __m, char_type __c) const [inline]`

Test char_type classification.

This function finds a mask M for c and compares it to mask m. It does so by returning the value of [ctype<char_type>::do_is\(\)](#).

Parameters

c The char_type to compare the mask of.

m The mask to compare against.

Returns

(M & m) != 0.

5.241 std::__ctype_abstract_base<_CharT> Class Template Reference 1331

Definition at line 161 of file locale_facets.h.

Referenced by std::regex_traits<_Ch_type>::isctype(), and std::basic_istream<_CharT, _Traits>::sentry::sentry().

5.241.3.15 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::narrow (const char_type
* __lo, const char_type * __hi, char __dfault, char * __to) const
[inline]`

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 345 of file locale_facets.h.

5.241.3.16 `template<typename _CharT> char std::__ctype_abstract_base<
_CharT>::narrow (char_type __c, char __dfault) const
[inline]`

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. It does so by returning ctype<char_type>::do_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

- c* The char_type to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

Definition at line 323 of file locale_facets.h.

Referenced by `std::time_put<_CharT, _OutIter >::put()`.

5.241.3.17 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT >::scan_is (mask __m,
const char_type * __lo, const char_type * __hi) const [inline]`

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning `ctype<char_type>::do_scan_is()`.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to matching char_type if found, else *hi*.

Definition at line 194 of file locale_facets.h.

5.241.3.18 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT >::scan_not (mask __m,
const char_type * __lo, const char_type * __hi) const [inline]`

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning `ctype<char_type>::do_scan_not()`.

Parameters

m The mask to compare against.

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale_facets.h.

5.241.3.19 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower (char_type __c) const [inline]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper(c).

Parameters

c The char_type to convert.

Returns

The lowercase char_type if convertible, else *c*.

Definition at line 253 of file locale_facets.h.

5.241.3.20 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower (char_type * __lo, const char_type * __hi) const [inline]`

Convert array to lowercase.

This function converts each char_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 268 of file locale_facets.h.

5.241.3.21 `template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::toupper (char_type __c) const [inline]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

5.241.3.22 `template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::toupper (char_type * __lo, const char_type * __hi) const [inline]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 239 of file `locale_facets.h`.

5.241.3.23 `template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::widen (char __c) const [inline]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted `char_type`.

Definition at line 285 of file `locale_facets.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::regex_traits<_Char_type>::isctype()`, and `std::operator<<()`.

5.241.3.24 `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen (const char * __lo, const char * __hi, char_type * __to) const [inline]`

Widen array to `char_type`.

This function converts each char in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 304 of file `locale_facets.h`.

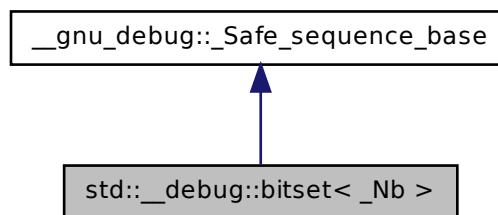
The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.242 `std::__debug::bitset< _Nb >` Class Template Reference

Class `std::bitset` with additional safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::bitset< _Nb >`:



Public Member Functions

- **bitset** (unsigned long long __val)
- template<class _CharT, class _Traits, class _Alloc >
bitset (const [std::basic_string](#)< _CharT, _Traits, _Alloc > &__str, typename [std::basic_string](#)< _CharT, _Traits, _Alloc >::size_type __pos, typename [std::basic_string](#)< _CharT, _Traits, _Alloc >::size_type __n, _CharT __zero, _CharT __one=_CharT('1'))
- **bitset** (const [_Base](#) &__x)
- template<typename _CharT, typename _Traits, typename _Alloc >
bitset (const [std::basic_string](#)< _CharT, _Traits, _Alloc > &__str, typename [std::basic_string](#)< _CharT, _Traits, _Alloc >::size_type __pos=0, typename [std::basic_string](#)< _CharT, _Traits, _Alloc >::size_type __n=([std::basic_string](#)< _CharT, _Traits, _Alloc >::npos))
- **bitset** (const char *__str)
- [_Base](#) & [_M_base](#) ()
- const [_Base](#) & [_M_base](#) () const
- void [_M_invalidate_all](#) () const
- [bitset](#)< _Nb > & **flip** ()
- [bitset](#)< _Nb > & **flip** (size_t __pos)
- bool **operator!=** (const [bitset](#)< _Nb > &__rhs) const

- [bitset](#)<_Nb> & **operator&=** (const [bitset](#)<_Nb> &__rhs)
- [bitset](#)<_Nb> **operator<<** (size_t __pos) const
- [bitset](#)<_Nb> & **operator<<=** (size_t __pos)
- bool **operator==** (const [bitset](#)<_Nb> &__rhs) const
- [bitset](#)<_Nb> **operator>>** (size_t __pos) const
- [bitset](#)<_Nb> & **operator>>=** (size_t __pos)
- reference **operator[]** (size_t __pos)
- bool **operator[]** (size_t __pos) const
- [bitset](#)<_Nb> & **operator^=** (const [bitset](#)<_Nb> &__rhs)
- [bitset](#)<_Nb> & **operator|=** (const [bitset](#)<_Nb> &__rhs)
- [bitset](#)<_Nb> **operator~** () const
- [bitset](#)<_Nb> & **reset** ()
- [bitset](#)<_Nb> & **reset** (size_t __pos)
- [bitset](#)<_Nb> & **set** ()
- [bitset](#)<_Nb> & **set** (size_t __pos, bool __val=true)
- [std::basic_string](#)< char, [std::char_traits](#)< char >, [std::allocator](#)< char > > **to_string** (char __zero, char __one= '1') const
- template<typename _CharT >
[std::basic_string](#)< _CharT, [std::char_traits](#)< _CharT >, [std::allocator](#)< _CharT > > **to_string** () const
- template<typename _CharT, typename _Traits >
[std::basic_string](#)< _CharT, _Traits, [std::allocator](#)< _CharT > > **to_string** () const
- template<class _CharT >
[std::basic_string](#)< _CharT, [std::char_traits](#)< _CharT >, [std::allocator](#)< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- [std::basic_string](#)< char, [std::char_traits](#)< char >, [std::allocator](#)< char > > **to_string** () const
- template<class _CharT, class _Traits >
[std::basic_string](#)< _CharT, _Traits, [std::allocator](#)< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits, class _Alloc >
[std::basic_string](#)< _CharT, _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<typename _CharT, typename _Traits, typename _Alloc >
[std::basic_string](#)< _CharT, _Traits, _Alloc > **to_string** () const

Public Attributes

- _Safe_iterator_base * [_M_const_iterators](#)
- _Safe_iterator_base * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) (_Safe_sequence_base &__x)

5.242.1 Detailed Description

`template<size_t _Nb> class std::__debug::bitset< _Nb >`

Class [std::bitset](#) with additional safety/checking/debug instrumentation.

Definition at line 43 of file `debug/bitset`.

5.242.2 Member Function Documentation

5.242.2.1 void `__gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.242.2.2 void `__gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.242.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw ()` [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.242.2.4 void __gnu_debug::Safe_sequence_base::M_invalidate_all ()
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file safe_base.h.

5.242.2.5 void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.242.2.6 void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.242.3 Member Data Documentation

5.242.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter().

5.242.3.2 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter().

5.242.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

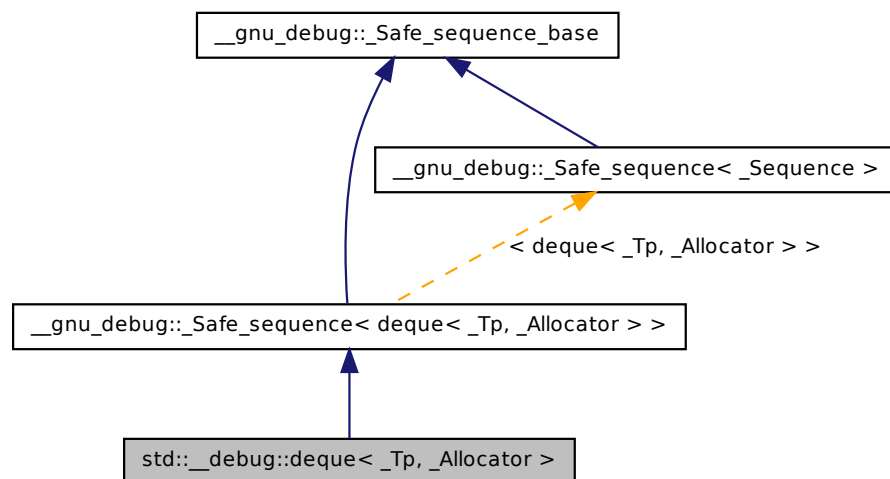
The documentation for this class was generated from the following file:

- [debug/bitset](#)

5.243 std::__debug::deque< _Tp, _Allocator > Class Template Reference

Class [std::deque](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::deque< _Tp, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**

- typedef [__gnu_debug::__Safe_iterator](#)< typename [_Base::const_iterator](#), [deque](#) > **const_iterator**
- typedef [_Base::const_pointer](#) **const_pointer**
- typedef [_Base::const_reference](#) **const_reference**
- typedef [std::reverse_iterator](#)< [const_iterator](#) > **const_reverse_iterator**
- typedef [_Base::difference_type](#) **difference_type**
- typedef [__gnu_debug::__Safe_iterator](#)< typename [_Base::iterator](#), [deque](#) > **iterator**
- typedef [_Base::pointer](#) **pointer**
- typedef [_Base::reference](#) **reference**
- typedef [std::reverse_iterator](#)< [iterator](#) > **reverse_iterator**
- typedef [_Base::size_type](#) **size_type**
- typedef [_Tp](#) **value_type**

Public Member Functions

- **deque** (const [_Allocator](#) &__a=_Allocator())
- **deque** (size_type __n)
- template<class [_InputIterator](#) >
deque ([_InputIterator](#) __first, [_InputIterator](#) __last, const [_Allocator](#) &__a=_Allocator())
- **deque** ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- **deque** (const [deque](#) &__x)
- **deque** (size_type __n, const [_Tp](#) &__value, const [_Allocator](#) &__a=_Allocator())
- **deque** (const [_Base](#) &__x)
- **deque** ([deque](#) &&__x)
- [_Base](#) & **_M_base** ()
- const [_Base](#) & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** ([_Predicate](#) __pred)
- void **_M_transfer_iter** (const [_Safe_iterator](#)< [_Iterator](#), [deque](#)< [_Tp](#), [_Allocator](#) > > &__x)
- template<class [_InputIterator](#) >
void **assign** ([_InputIterator](#) __first, [_InputIterator](#) __last)
- void **assign** (size_type __n, const [_Tp](#) &__t)
- void **assign** ([initializer_list](#)< value_type > __l)
- reference **back** ()
- const_reference **back** () const
- [iterator](#) **begin** ()
- [const_iterator](#) **begin** () const
- [const_iterator](#) **cbegin** () const

- [const_iterator](#) **cend** () const
- void **clear** ()
- [const_reverse_iterator](#) **crbegin** () const
- [const_reverse_iterator](#) **crend** () const
- template<typename... _Args>
[iterator](#) **emplace** ([iterator](#) __position, _Args &&...__args)
- template<typename... _Args>
void **emplace_back** (_Args &&...__args)
- template<typename... _Args>
void **emplace_front** (_Args &&...__args)
- [iterator](#) **end** ()
- [const_iterator](#) **end** () const
- [iterator](#) **erase** ([iterator](#) __first, [iterator](#) __last)
- [iterator](#) **erase** ([iterator](#) __position)
- reference **front** ()
- const_reference **front** () const
- [iterator](#) **insert** ([iterator](#) __position, const _Tp &__x)
- [iterator](#) **insert** ([iterator](#) __position, _Tp &&__x)
- void **insert** ([iterator](#) __p, [initializer_list](#)< value_type > __l)
- void **insert** ([iterator](#) __position, size_type __n, const _Tp &__x)
- template<class _InputIterator >
void **insert** ([iterator](#) __position, _InputIterator __first, _InputIterator __last)
- [deque](#) & **operator=** ([deque](#) &&__x)
- [deque](#) & **operator=** (const [deque](#) &__x)
- [deque](#) & **operator=** ([initializer_list](#)< value_type > __l)
- const_reference **operator[]** (size_type __n) const
- reference **operator[]** (size_type __n)
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (_Tp &&__x)
- void **push_back** (const _Tp &__x)
- void **push_front** (const _Tp &__x)
- void **push_front** (_Tp &&__x)
- [reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rbegin** () const
- [const_reverse_iterator](#) **rend** () const
- [reverse_iterator](#) **rend** ()
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **swap** ([deque](#) &__x)

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) (_Safe_sequence_base &__x)

5.243.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__debug::deque<_Tp, _Allocator>

Class [std::deque](#) with safety/checking/debug instrumentation.

Definition at line 43 of file debug/deque.

5.243.2 Member Function Documentation

5.243.2.1 void [__gnu_debug::Safe_sequence_base::_M_detach_all](#) ()
[protected, inherited]

Detach all iterators, leaving them singular.

5.243.2.2 void [__gnu_debug::Safe_sequence_base::_M_detach_singular](#) ()
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

5.243.2.3 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.243.2.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.243.2.5 `void __gnu_debug::_Safe_sequence< deque< _Tp, _Allocator > >::_M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.243.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.243.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.243.2.8 `void __gnu_debug::_Safe_sequence< deque< _Tp, _Allocator > >::_M_transfer_iter (const _Safe_iterator< _Iterator, deque< _Tp, _Allocator > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.243.3 Member Data Documentation

5.243.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.243.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.243.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

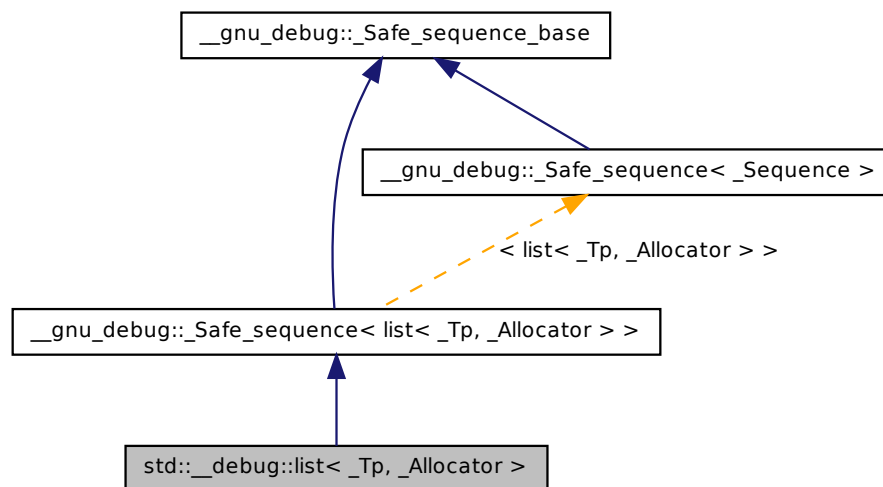
The documentation for this class was generated from the following file:

- [debug/deque](#)

5.244 `std::__debug::list< _Tp, _Allocator >` Class Template Reference

Class `std::list` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::list< _Tp, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, list >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, list >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **list** (const _Allocator &__a=_Allocator())
- **list** (size_type __n)
- template<class _InputIterator >
list (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **list** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- **list** (const list &__x)
- **list** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- **list** (const _Base &__x)
- **list** (list &&__x)
- _Base & _M_base ()
- const _Base & _M_base () const
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_iter (const _Safe_iterator< _Iterator, list< _Tp, _Allocator > &__x)
- void **assign** (size_type __n, const _Tp &__t)
- void **assign** (initializer_list< value_type > __l)
- template<class _InputIterator >
void **assign** (_InputIterator __first, _InputIterator __last)
- const_reference **back** () const
- reference **back** ()
- const_iterator **begin** () const
- iterator **begin** ()
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- template<typename... _Args>
iterator **emplace** (iterator __position, _Args &&...__args)
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __position, iterator __last)
- const_reference **front** () const
- reference **front** ()
- iterator **insert** (iterator __position, const _Tp &__x)
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __p, initializer_list< value_type > __l)

- `template<class _InputIterator >`
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `void insert (iterator __position, size_type __n, const _Tp &__x)`
- `template<typename _Compare >`
`void merge (list &__x, _Compare __comp)`
- `void merge (list &&__x)`
- `void merge (list &__x)`
- `template<class _Compare >`
`void merge (list &&__x, _Compare __comp)`
- `list & operator= (const list &__x)`
- `list & operator= (list &&__x)`
- `list & operator= (initializer_list< value_type > __l)`
- `void pop_back ()`
- `void pop_front ()`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `void remove (const _Tp &__value)`
- `template<class _Predicate >`
`void remove_if (_Predicate __pred)`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const _Tp &__c)`
- `template<typename _StrictWeakOrdering >`
`void sort (_StrictWeakOrdering __pred)`
- `void sort ()`
- `void splice (iterator __position, list &&__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &__x, iterator __i)`
- `void splice (iterator __position, list &&__x, iterator __i)`
- `void splice (iterator __position, list &__x, iterator __first, iterator __last)`
- `void splice (iterator __position, list &&__x)`
- `void splice (iterator __position, list &__x)`
- `void splice (iterator __position, list &__x)`
- `void swap (list &__x)`
- `template<class _BinaryPredicate >`
`void unique (_BinaryPredicate __binary_pred)`
- `void unique ()`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- __gnu_cxx::__mutex & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) (_Safe_sequence_base &__x)

5.244.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__debug::list< _Tp, _Allocator >

Class [std::list](#) with safety/checking/debug instrumentation.

Definition at line 43 of file debug/list.

5.244.2 Member Function Documentation

5.244.2.1 void __gnu_debug::Safe_sequence_base::_M_detach_all ()
[protected, inherited]

Detach all iterators, leaving them singular.

5.244.2.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular ()
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

5.244.2.3 __gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by [__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if\(\)](#), and [__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter\(\)](#).

5.244.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()
const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.244.2.5 `void __gnu_debug::Safe_sequence< list< _Tp, _Allocator >
>::_M_invalidate_if(_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.244.2.6 `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.244.2.7 `void __gnu_debug::Safe_sequence_base::_M_swap (
_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.244.2.8 `void __gnu_debug::Safe_sequence< list< _Tp, _Allocator >
>::_M_transfer_iter(const _Safe_iterator< _Iterator, list< _Tp,
_Allocator > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.244.3 Member Data Documentation

5.244.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_
const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

5.245 `std::__debug::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference 1351

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.244.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.244.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

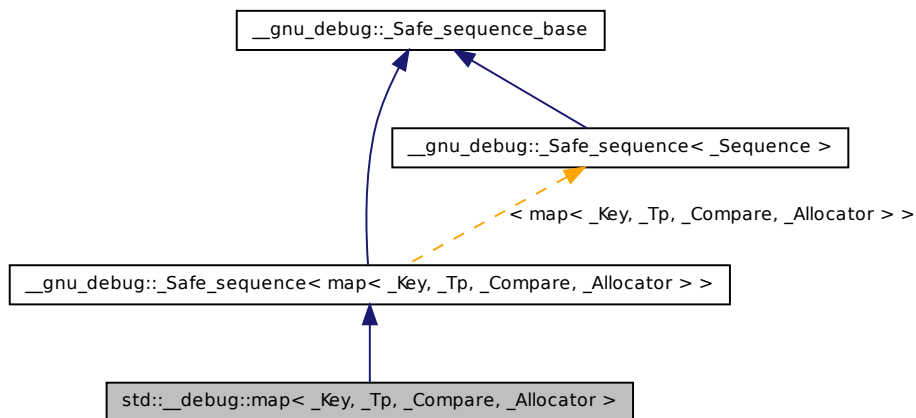
The documentation for this class was generated from the following file:

- [debug/list](#)

5.245 `std::__debug::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::map` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, map >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, map >` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- **map** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
map (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **map** (const _Base &__x)
- **map** (map &&__x)
- **map** (const map &__x)
- **map** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())
- _Base & _M_base ()
- const _Base & _M_base () const
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_iter (const _Safe_iterator< _Iterator, map< _Key, _Tp, _Compare, _Allocator > > &__x)
- iterator begin ()
- const_iterator begin () const
- const_iterator cbegin () const
- const_iterator cend () const
- void clear ()
- const_reverse_iterator crbegin () const
- const_reverse_iterator crend () const
- iterator end ()
- const_iterator end () const
- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- iterator erase (iterator __first, iterator __last)
- iterator erase (iterator __position)
- size_type erase (const key_type &__x)
- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const
- void insert (std::initializer_list< value_type > __list)
- std::pair< iterator, bool > insert (const value_type &__x)
- iterator insert (iterator __position, const value_type &__x)
- template<typename _InputIterator >
void insert (_InputIterator __first, _InputIterator __last)
- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const
- map & operator= (const map &__x)

- `map` & `operator=` (`map` &&__x)
- `map` & `operator=` (`initializer_list`< `value_type` > __l)
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rend` () const
- `reverse_iterator` `rend` ()
- void `swap` (`map` &__x)
- `const_iterator` `upper_bound` (const key_type &__x) const
- `iterator` `upper_bound` (const key_type &__x)

Public Attributes

- `_Safe_iterator_base` * `_M_const_iterators`
- `_Safe_iterator_base` * `_M_iterators`
- unsigned int `_M_version`

Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_sequence_base` &__x)

5.245.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::map< _Key, _Tp, _Compare, _Allocator >
```

Class `std::map` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/map.h`.

5.245.2 Member Function Documentation

5.245.2.1 void __gnu_debug::Safe_sequence_base::_M_detach_all () [protected, inherited]

Detach all iterators, leaving them singular.

5.245 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference 1355

5.245.2.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular () [protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.245.2.3 __gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by __gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if(), and __gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter().

5.245.2.4 void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file safe_base.h.

5.245.2.5 void __gnu_debug::Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which *pred*(*x*) returns `true`. The user of this routine should be careful not to make copies of the iterators passed to *pred*, as the copies may interfere with the invalidation.

5.245.2.6 void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.245.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (`
`_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.245.2.8 `void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare,`
`_Allocator > >::_M_transfer_iter (const _Safe_iterator<`
`_Iterator, map< _Key, _Tp, _Compare, _Allocator > > & __x)`
`[inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.245.3 Member Data Documentation

5.245.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_`
`const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.245.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_`
`iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.245.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version`
`[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

5.246 `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference 1357

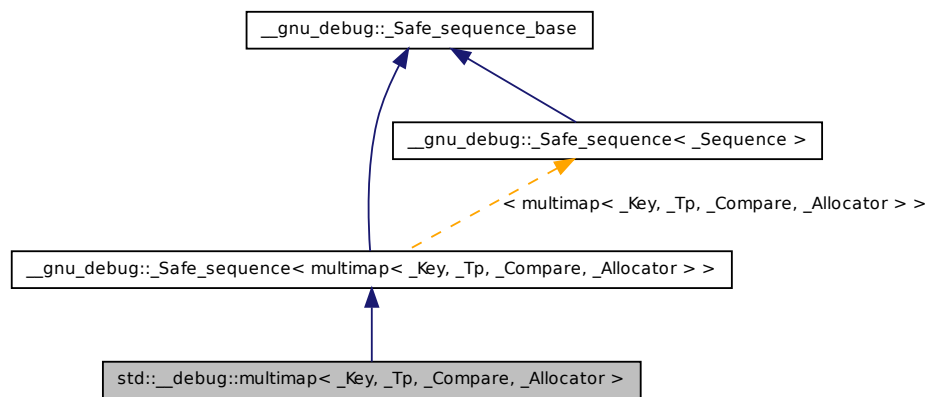
The documentation for this class was generated from the following file:

- [debug/map.h](#)

5.246 `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::multimap` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, multimap >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::iterator, multimap >` **iterator**

- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- **multimap** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
multimap (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **multimap** (const `_Base` &__x)
- **multimap** (`multimap` &&__x)
- **multimap** (const `multimap` &__x)
- **multimap** (`initializer_list`< `value_type` > __l, const `_Compare` &__c=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (`_Predicate` __pred)
- void **_M_transfer_iter** (const `_Safe_iterator`< `_Iterator`, `multimap`< `_Key`, `_Tp`, `_Compare`, `_Allocator` > > &__x)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal_range** (const `key_type` &__x)
- `std::pair`< `const_iterator`, `const_iterator` > **equal_range** (const `key_type` &__x) const
- `iterator` **erase** (`iterator` __first, `iterator` __last)
- `iterator` **erase** (`iterator` __position)
- `size_type` **erase** (const `key_type` &__x)
- `iterator` **find** (const `key_type` &__x)

- `const_iterator find` (const key_type &__x) const
- void `insert` (std::initializer_list< value_type > __list)
- `iterator insert` (const value_type &__x)
- `iterator insert` (iterator __position, const value_type &__x)
- template<typename _InputIterator >
 void `insert` (_InputIterator __first, _InputIterator __last)
- `iterator lower_bound` (const key_type &__x)
- `const_iterator lower_bound` (const key_type &__x) const
- `multimap & operator=` (const multimap &__x)
- `multimap & operator=` (multimap &&__x)
- `multimap & operator=` (initializer_list< value_type > __l)
- `const_reverse_iterator rbegin` () const
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rend` () const
- `reverse_iterator rend` ()
- void `swap` (multimap &__x)
- `const_iterator upper_bound` (const key_type &__x) const
- `iterator upper_bound` (const key_type &__x)

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- unsigned int `_M_version`

Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (_Safe_sequence_base &__x)

5.246.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class `std::multimap` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/multimap.h`.

5.246.2 Member Function Documentation

5.246.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.246.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.246.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw ()` **[protected, inherited]**

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.246.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()`
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.246.2.5 `void __gnu_debug::Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred)`
[inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.246.2.6 void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.246.2.7 void __gnu_debug::Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.246.2.8 void __gnu_debug::Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_iter (const _Safe_iterator< _Iterator, multimap< _Key, _Tp, _Compare, _Allocator > & __x) [inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.246.3 Member Data Documentation

5.246.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter().

5.246.3.2 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter().

5.246.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

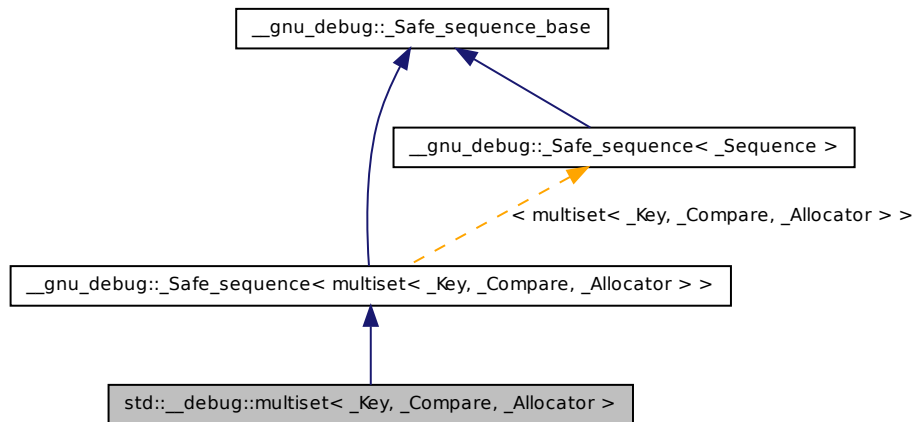
The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

5.247 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference

Class [std::multiset](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::multiset< _Key, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::_Safe_iterator< typename _Base::const_iterator, multiset >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**

- typedef _Base::const_reference **const_reference**
- typedef [std::reverse_iterator](#)< [const_iterator](#) > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef [__gnu_debug::_Safe_iterator](#)< typename _Base::iterator, [multiset](#) > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef [std::reverse_iterator](#)< [iterator](#) > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **multiset** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **multiset** (const [_Base](#) &__x)
- **multiset** ([multiset](#) &&__x)
- **multiset** (const [multiset](#) &__x)
- **multiset** ([initializer_list](#)< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- [_Base](#) & **_M_base** ()
- const [_Base](#) & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (_Predicate __pred)
- void **_M_transfer_iter** (const [_Safe_iterator](#)< _Iterator, [multiset](#)< _Key, _Compare, _Allocator > > &__x)
- [iterator](#) **begin** ()
- [const_iterator](#) **begin** () const
- [const_iterator](#) **cbegin** () const
- [const_iterator](#) **cend** () const
- void **clear** ()
- [const_reverse_iterator](#) **crbegin** () const
- [const_reverse_iterator](#) **crend** () const
- [iterator](#) **end** ()
- [const_iterator](#) **end** () const
- [std::pair](#)< [iterator](#), [iterator](#) > **equal_range** (const key_type &__x)

- `std::pair< const_iterator, const_iterator > equal_range` (const key_type &__x) const
- `iterator erase` (iterator __first, iterator __last)
- `iterator erase` (iterator __position)
- `size_type erase` (const key_type &__x)
- `iterator find` (const key_type &__x)
- `const_iterator find` (const key_type &__x) const
- `iterator insert` (iterator __position, const value_type &__x)
- `iterator insert` (const value_type &__x)
- `template<typename _InputIterator > void insert` (_InputIterator __first, _InputIterator __last)
- `void insert` (initializer_list< value_type > __l)
- `iterator lower_bound` (const key_type &__x)
- `const_iterator lower_bound` (const key_type &__x) const
- `multiset & operator=` (const multiset &__x)
- `multiset & operator=` (multiset &&__x)
- `multiset & operator=` (initializer_list< value_type > __l)
- `const_reverse_iterator rbegin` () const
- `reverse_iterator rbegin` ()
- `const_reverse_iterator rend` () const
- `reverse_iterator rend` ()
- `void swap` (multiset &__x)
- `const_iterator upper_bound` (const key_type &__x) const
- `iterator upper_bound` (const key_type &__x)

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `void _M_revalidate_singular` ()
- `void _M_swap` (_Safe_sequence_base &__x)

5.247.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename
_Allocator = std::allocator<_Key>> class std::__debug::multiset< _Key, _-
_Compare, _Allocator >
```

Class [std::multiset](#) with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/multiset.h`.

5.247.2 Member Function Documentation

5.247.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.247.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.247.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_-`
`base::_M_get_mutex () throw ()` [protected,
inherited]

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`,
and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.247.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()`
`const` [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.247.2.5 `void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred)`
[inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.247.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ()`
[protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.247.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x)`
[protected, inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.247.2.8 `void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_transfer_iter (const _Safe_iterator< _Iterator, multiset< _Key, _Compare, _Allocator > > & __x)`
[inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.247.3 Member Data Documentation

5.247.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` **[inherited]**

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.248 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference 1367

5.247.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter().

5.247.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

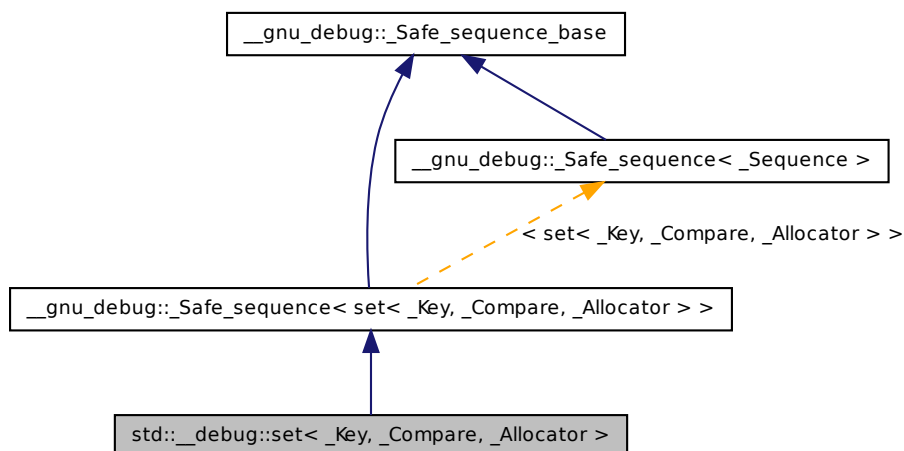
The documentation for this class was generated from the following file:

- [debug/multiset.h](#)

5.248 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference

Class [std::set](#) with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::set< _Key, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, set >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, set >` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Compare` **value_compare**
- typedef `_Key` **value_type**

Public Member Functions

- **set** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
 set (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **set** (const `_Base` &__x)
- **set** (`set` &&__x)
- **set** (const `set` &__x)
- **set** (`initializer_list`< `value_type` > __l, const `_Compare` &__comp=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (`_Predicate` __pred)
- void **_M_transfer_iter** (const `_Safe_iterator`< `_Iterator`, `set`< `_Key`, `_Compare`, `_Allocator` > > &__x)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `const_iterator` **cbegin** () const
- `const_iterator` **cend** () const
- void **clear** ()
- `const_reverse_iterator` **crbegin** () const
- `const_reverse_iterator` **crend** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `std::pair`< `iterator`, `iterator` > **equal_range** (const `key_type` &__x)
- `std::pair`< `const_iterator`, `const_iterator` > **equal_range** (const `key_type` &__x)
 const
- `iterator` **erase** (`iterator` __first, `iterator` __last)
- `iterator` **erase** (`iterator` __position)
- `size_type` **erase** (const `key_type` &__x)
- `iterator` **find** (const `key_type` &__x)
- `const_iterator` **find** (const `key_type` &__x) const
- `iterator` **insert** (`iterator` __position, const `value_type` &__x)
- `std::pair`< `iterator`, `bool` > **insert** (const `value_type` &__x)
- template<typename `_InputIterator` >
 void **insert** (`_InputIterator` __first, `_InputIterator` __last)
- void **insert** (`initializer_list`< `value_type` > __l)
- `iterator` **lower_bound** (const `key_type` &__x)
- `const_iterator` **lower_bound** (const `key_type` &__x) const
- `set` & **operator=** (const `set` &__x)

- `set & operator= (set &&__x)`
- `set & operator= (initializer_list< value_type > __l)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `void swap (set &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `iterator upper_bound (const key_type &__x)`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

5.248.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__debug::set< _Key, _Compare, _Allocator >`

Class `std::set` with safety/checking/debug instrumentation.

Definition at line 44 of file `debug/set.h`.

5.248.2 Member Function Documentation

5.248.2.1 `void __gnu_debug::_Safe_sequence_base::_M_detach_all ()` [protected, inherited]

Detach all iterators, leaving them singular.

5.248 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference 1371

5.248.2.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular ()
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.248.2.3 __gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]

For use in [_Safe_sequence](#).

Referenced by __gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if(),
and __gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter().

5.248.2.4 void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file safe_base.h.

5.248.2.5 void __gnu_debug::Safe_sequence< set< _Key, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred)
[inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which *pred*(*x*) returns `true`. The user of this routine should be careful not to make copies of the iterators passed to *pred*, as the copies may interfere with the invalidation.

5.248.2.6 void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.248.2.7 `void __gnu_debug::Safe_sequence_base::M_swap (`
`_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.248.2.8 `void __gnu_debug::Safe_sequence< set< _Key, _Compare,`
`_Allocator > >::M_transfer_iter (const _Safe_iterator< _Iterator,`
`set< _Key, _Compare, _Allocator > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.248.3 Member Data Documentation

5.248.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_`
`const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

5.248.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_`
`iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

5.248.3.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version`
`[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

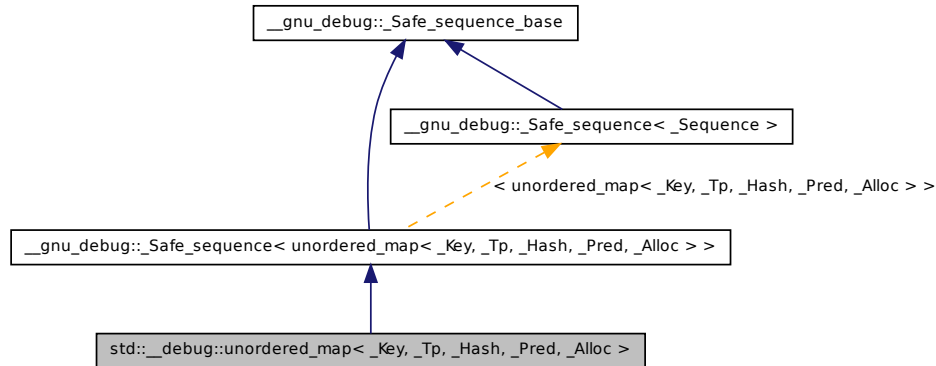
The documentation for this class was generated from the following file:

- [debug/set.h](#)

5.249 `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_map` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`:



Public Types

- typedef `_Base::allocator_type` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, unordered_map >` **const_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, unordered_map >` **iterator**
- typedef `_Base::key_equal` **key_equal**
- typedef `_Base::key_type` **key_type**
- typedef `_Base::size_type` **size_type**
- typedef `_Base::value_type` **value_type**

Public Member Functions

- **unordered_map** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_map (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (const [_Base](#) &__x)
- **unordered_map** ([unordered_map](#) &&__x)
- **unordered_map** (const [unordered_map](#) &__x)
- **unordered_map** ([initializer_list](#)< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- [_Base](#) & [_M_base](#) ()
- const [_Base](#) & [_M_base](#) () const
- void [_M_invalidate_all](#) () const
- void [_M_invalidate_if](#) (_Predicate __pred)
- void [_M_transfer_iter](#) (const _Safe_iterator< _Iterator, [unordered_map](#)< _Key, _Tp, _Hash, _Pred, _Alloc > &__x)
- [iterator](#) **begin** ()
- [const_iterator](#) **begin** () const
- [const_iterator](#) **cbegin** () const
- [const_iterator](#) **cend** () const
- void **clear** ()
- [const_iterator](#) **end** () const
- [iterator](#) **end** ()
- [std::pair](#)< [iterator](#), [iterator](#) > **equal_range** (const key_type &__key)
- [std::pair](#)< [const_iterator](#), [const_iterator](#) > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- [iterator](#) **erase** ([const_iterator](#) __it)
- [iterator](#) **erase** ([const_iterator](#) __first, [const_iterator](#) __last)
- [iterator](#) **find** (const key_type &__key)
- [const_iterator](#) **find** (const key_type &__key) const
- [iterator](#) **insert** ([const_iterator](#), const value_type &__obj)
- [std::pair](#)< [iterator](#), bool > **insert** (const value_type &__obj)
- void **insert** ([std::initializer_list](#)< value_type > __l)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- [unordered_map](#) & **operator=** ([unordered_map](#) &&__x)
- [unordered_map](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_map](#) & **operator=** (const [unordered_map](#) &__x)
- void **swap** ([unordered_map](#) &__x)

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

5.249.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>
```

Class `std::unordered_map` with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_map`.

5.249.2 Member Function Documentation

5.249.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.249.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.249.2.3 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.249.2.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.249.2.5 `void __gnu_debug::_Safe_sequence< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.249.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.249.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.249 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference 1377

5.249.2.8 void __gnu_debug::Safe_sequence< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > >::M_transfer_iter (const _Safe_iterator< _Iterator, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > > & __x) **[inherited]**

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.249.3 Member Data Documentation

5.249.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators **[inherited]**

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter().

5.249.3.2 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators **[inherited]**

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter().

5.249.3.3 unsigned int __gnu_debug::Safe_sequence_base::_M_version **[mutable, inherited]**

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

The documentation for this class was generated from the following file:

- [debug/unordered_map](#)

5.250 `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_multimap` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`:



Public Types

- typedef `_Base::allocator_type` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, unordered_multimap >` **const_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, unordered_multimap >` **iterator**
- typedef `_Base::key_equal` **key_equal**
- typedef `_Base::key_type` **key_type**
- typedef `_Base::size_type` **size_type**
- typedef `_Base::value_type` **value_type**

Public Member Functions

- **unordered_multimap** (`size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- `template<typename InputIterator >`
unordered_multimap (`InputIterator __first, InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- **unordered_multimap** (`const _Base &__x`)
- **unordered_multimap** (`unordered_multimap &&__x`)
- **unordered_multimap** (`const unordered_multimap &__x`)
- **unordered_multimap** (`initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`

- `void _M_invalidate_all () const`
- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_iter (const _Safe_iterator< _Iterator, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > > &__x)`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `const_iterator end () const`
- `iterator end ()`
- `std::pair< iterator, iterator > equal_range (const key_type &__key)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__key) const`
- `size_type erase (const key_type &__key)`
- `iterator erase (const_iterator __it)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator find (const key_type &__key)`
- `const_iterator find (const key_type &__key) const`
- `iterator insert (const_iterator, const value_type &__obj)`
- `iterator insert (const value_type &__obj)`
- `void insert (std::initializer_list< value_type > __l)`
- `template<typename _InputIterator >
void insert (_InputIterator __first, _InputIterator __last)`
- `unordered_multimap & operator= (unordered_multimap &&__x)`
- `unordered_multimap & operator= (initializer_list< value_type > __l)`
- `unordered_multimap & operator= (const unordered_multimap &__x)`
- `void swap (unordered_multimap &__x)`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

5.250.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class [std::unordered_multimap](#) with safety/checking/debug instrumentation.

Definition at line 311 of file `debug/unordered_map`.

5.250.2 Member Function Documentation

5.250.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all ()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.250.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular ()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.250.2.3 `__gnu_cxx::__mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw ()` **[protected, inherited]**

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.250.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()`
const [inline, inherited]

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.250.2.5 `void __gnu_debug::Safe_sequence< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.250.2.6 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.250.2.7 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.250.2.8 `void __gnu_debug::Safe_sequence< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::M_transfer_iter (const _Safe_iterator< _Iterator, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.250.3 Member Data Documentation

5.250.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::M_invalidate_if()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_invalidate_single()`, and `__gnu_debug::Safe_sequence< _Sequence >::M_transfer_iter()`.

5.250.3.2 `_Safe_iterator_base*` `__gnu_debug::_Safe_sequence_base::_M_iterators` `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_iter()`.

5.250.3.3 `unsigned int` `__gnu_debug::_Safe_sequence_base::_M_version` `[mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/unordered_map](#)

5.251 `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>` Class Template Reference

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`:



Public Types

- `typedef _Base::allocator_type` **allocator_type**
- `typedef __gnu_debug::_Safe_iterator< typename _Base::const_iterator, unordered_multiset >` **const_iterator**
- `typedef _Base::hasher` **hasher**
- `typedef __gnu_debug::_Safe_iterator< typename _Base::iterator, unordered_multiset >` **iterator**
- `typedef _Base::key_equal` **key_equal**
- `typedef _Base::key_type` **key_type**

- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multiset** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (const _Base &__x)
- **unordered_multiset** (unordered_multiset &&__x)
- **unordered_multiset** (const unordered_multiset &__x)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- _Base & **_M_base** ()
- const _Base & **_M_base** () const
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (_Predicate __pred)
- void **_M_transfer_iter** (const _Safe_iterator< _Iterator, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x)
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_iterator **end** () const
- iterator **end** ()
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __it)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- iterator **insert** (const_iterator, const value_type &__obj)
- iterator **insert** (const value_type &__obj)
- void **insert** (std::initializer_list< value_type > __l)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)

- `unordered_multiset` & `operator=` (`unordered_multiset` &&__x)
- `unordered_multiset` & `operator=` (`initializer_list`< `value_type` > __l)
- `unordered_multiset` & `operator=` (const `unordered_multiset` &__x)
- void `swap` (`unordered_multiset` &__x)

Public Attributes

- `_Safe_iterator_base` * `_M_const_iterators`
- `_Safe_iterator_base` * `_M_iterators`
- unsigned int `_M_version`

Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_sequence_base` &__x)

5.251.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Definition at line 308 of file `debug/unordered_set`.

5.251.2 Member Function Documentation

5.251.2.1 void `__gnu_debug::_Safe_sequence_base::_M_detach_all` ()
[`protected`, `inherited`]

Detach all iterators, leaving them singular.

5.251.2.2 void `__gnu_debug::_Safe_sequence_base::_M_detach_singular` ()
[`protected`, `inherited`]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

5.251.2.3 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.251.2.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all () const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.251.2.5 `void __gnu_debug::Safe_sequence< unordered_multiset< _Value, _Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.251.2.6 `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.251.2.7 `void __gnu_debug::Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.251.2.8 `void __gnu_debug::_Safe_sequence< unordered_multiset< _Value, _Hash, _Pred, _Alloc > >::_M_transfer_iter (const _Safe_iterator< _Iterator, unordered_multiset< _Value, _Hash, _Pred, _Alloc > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.251.3 Member Data Documentation

5.251.3.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.251.3.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]`

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.251.3.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable, inherited]`

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

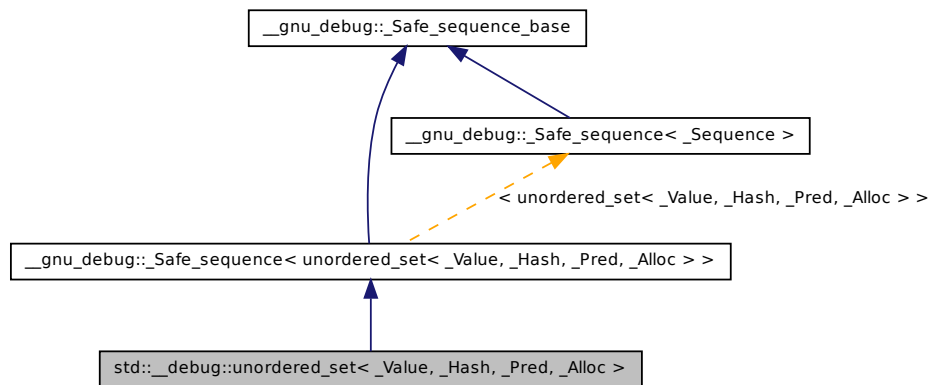
The documentation for this class was generated from the following file:

- [debug/unordered_set](#)

5.252 `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_set` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`:



Public Types

- typedef `_Base::allocator_type` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, unordered_set >` **const_iterator**
- typedef `_Base::hasher` **hasher**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, unordered_set >` **iterator**
- typedef `_Base::key_equal` **key_equal**
- typedef `_Base::key_type` **key_type**
- typedef `_Base::size_type` **size_type**
- typedef `_Base::value_type` **value_type**

Public Member Functions

- **unordered_set** (`size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type()`)

- `template<typename _InputIterator >`
`unordered_set` (`_InputIterator` __first, `_InputIterator` __last, `size_type` __n=0,
`const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const`
`allocator_type &__a=allocator_type()`)
- `unordered_set` (`const _Base &__x`)
- `unordered_set` (`unordered_set &&__x`)
- `unordered_set` (`const unordered_set &__x`)
- `unordered_set` (`initializer_list`< `value_type` > __l, `size_type` __n=0, `const`
`hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_-`
`type &__a=allocator_type()`)
- `_Base & _M_base` ()
- `const _Base & _M_base` () `const`
- `void _M_invalidate_all` () `const`
- `void _M_invalidate_if` (`_Predicate` __pred)
- `void _M_transfer_iter` (`const _Safe_iterator`< `_Iterator`, `unordered_set`< `_Value`,
`_Hash`, `_Pred`, `_Alloc` > > &__x)
- `iterator begin` ()
- `const_iterator begin` () `const`
- `const_iterator cbegin` () `const`
- `const_iterator cend` () `const`
- `void clear` ()
- `const_iterator end` () `const`
- `iterator end` ()
- `std::pair`< `iterator`, `iterator` > `equal_range` (`const key_type &__key`)
- `std::pair`< `const_iterator`, `const_iterator` > `equal_range` (`const key_type &__-`
`key`) `const`
- `size_type erase` (`const key_type &__key`)
- `iterator erase` (`const_iterator` __it)
- `iterator erase` (`const_iterator` __first, `const_iterator` __last)
- `iterator find` (`const key_type &__key`)
- `const_iterator find` (`const key_type &__key`) `const`
- `iterator insert` (`const_iterator`, `const value_type &__obj`)
- `std::pair`< `iterator`, `bool` > `insert` (`const value_type &__obj`)
- `void insert` (`std::initializer_list`< `value_type` > __l)
- `template<typename _InputIterator >`
`void insert` (`_InputIterator` __first, `_InputIterator` __last)
- `unordered_set & operator=` (`unordered_set &&__x`)
- `unordered_set & operator=` (`initializer_list`< `value_type` > __l)
- `unordered_set & operator=` (`const unordered_set &__x`)
- `void swap` (`unordered_set &__x`)

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all()`
- `void _M_detach_singular()`
- `__gnu_cxx::__mutex & _M_get_mutex() throw()`
- `void _M_revalidate_singular()`
- `void _M_swap(_Safe_sequence_base &__x)`

5.252.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>
```

Class `std::unordered_set` with safety/checking/debug instrumentation.

Definition at line 50 of file `debug/unordered_set`.

5.252.2 Member Function Documentation

5.252.2.1 `void __gnu_debug::Safe_sequence_base::_M_detach_all()`
[protected, inherited]

Detach all iterators, leaving them singular.

5.252.2.2 `void __gnu_debug::Safe_sequence_base::_M_detach_singular()`
[protected, inherited]

Detach all singular iterators.

Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

5.252.2.3 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected, inherited]`

For use in [_Safe_sequence](#).

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.252.2.4 `void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.252.2.5 `void __gnu_debug::_Safe_sequence< unordered_set< _Value, _Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.252.2.6 `void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.252.2.7 `void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.252.2.8 void __gnu_debug::Safe_sequence< unordered_set< _Value, _Hash, _Pred, _Alloc > >::_M_transfer_iter (const _Safe_iterator< _Iterator, unordered_set< _Value, _Hash, _Pred, _Alloc > > & __x)
 [inherited]

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.252.3 Member Data Documentation

5.252.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 166 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter().

5.252.3.2 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file safe_base.h.

Referenced by __gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single(), and __gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter().

5.252.3.3 unsigned int __gnu_debug::Safe_sequence_base::_M_version [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file safe_base.h.

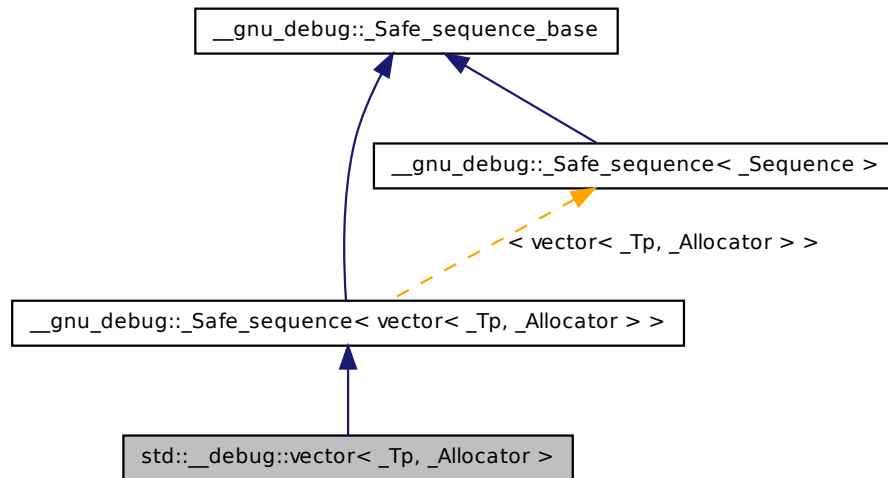
The documentation for this class was generated from the following file:

- [debug/unordered_set](#)

5.253 `std::__debug::vector< _Tp, _Allocator >` Class Template Reference

Class `std::vector` with safety/checking/debug instrumentation.

Inheritance diagram for `std::__debug::vector< _Tp, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::const_iterator, vector >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::Safe_iterator< typename _Base::iterator, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**

- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **vector** (const _Allocator &__a=_Allocator())
- **vector** (size_type __n)
- template<class _InputIterator >
vector (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **vector** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- **vector** (const vector &__x)
- **vector** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- vector (const _Base &__x)
- **vector** (vector &&__x)
- _Base & _M_base ()
- const _Base & _M_base () const
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_iter (const _Safe_iterator< _Iterator, vector< _Tp, _Allocator > > &__x)
- template<typename _InputIterator >
void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__u)
- void **assign** (initializer_list< value_type > __l)
- reference **back** ()
- const_reference **back** () const
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **capacity** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- template<typename... _Args>
iterator **emplace** (iterator __position, _Args &&...__args)
- template<typename... _Args>
void **emplace_back** (_Args &&...__args)
- iterator **end** ()
- const_iterator **end** () const

- [iterator erase](#) ([iterator](#) __first, [iterator](#) __last)
- [iterator erase](#) ([iterator](#) __position)
- reference **front** ()
- const_reference **front** () const
- void **insert** ([iterator](#) __position, [initializer_list](#)< value_type > __l)
- void **insert** ([iterator](#) __position, size_type __n, const _Tp &__x)
- [iterator insert](#) ([iterator](#) __position, const _Tp &__x)
- template<typename _Up = _Tp>
__gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, [iterator](#) >::__type **insert** ([iterator](#) __position, _Tp &&__x)
- template<class _InputIterator >
void **insert** ([iterator](#) __position, _InputIterator __first, _InputIterator __last)
- [vector](#) & **operator=** (const [vector](#) &__x)
- [vector](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [vector](#) & **operator=** ([vector](#) &&__x)
- reference **operator[]** (size_type __n)
- const_reference **operator[]** (size_type __n) const
- void **pop_back** ()
- void **push_back** (const _Tp &__x)
- template<typename _Up = _Tp>
__gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, void >::__type **push_back** (_Tp &&__x)
- [reverse_iterator](#) **rbegin** ()
- [const_reverse_iterator](#) **rbegin** () const
- [reverse_iterator](#) **rend** ()
- [const_reverse_iterator](#) **rend** () const
- void **reserve** (size_type __n)
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **swap** ([vector](#) &__x)

Public Attributes

- _Safe_iterator_base * [_M_const_iterators](#)
- _Safe_iterator_base * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- __gnu_cxx::__mutex & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) (_Safe_sequence_base &__x)

5.253.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__debug::vector< _Tp, _Allocator >
```

Class [std::vector](#) with safety/checking/debug instrumentation.

Definition at line 45 of file debug/vector.

5.253.2 Constructor & Destructor Documentation

```
5.253.2.1 template<typename _Tp, typename _Allocator =
std::allocator<_Tp>> std::__debug::vector< _Tp, _Allocator
>::vector ( const_Base & __x ) [inline]
```

Construction from a release-mode vector.

Definition at line 107 of file debug/vector.

5.253.3 Member Function Documentation

```
5.253.3.1 void __gnu_debug::Safe_sequence_base::_M_detach_all ( )
[protected, inherited]
```

Detach all iterators, leaving them singular.

```
5.253.3.2 void __gnu_debug::Safe_sequence_base::_M_detach_singular ( )
[protected, inherited]
```

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

```
5.253.3.3 __gnu_cxx::__mutex& __gnu_debug::Safe_sequence_-
base::_M_get_mutex ( ) throw () [protected,
inherited]
```

For use in [_Safe_sequence](#).

Referenced by [__gnu_debug::Safe_sequence< _Sequence >::_M_invalidate_if\(\)](#),
and [__gnu_debug::Safe_sequence< _Sequence >::_M_transfer_iter\(\)](#).

5.253.3.4 `void __gnu_debug::Safe_sequence_base::_M_invalidate_all ()
const [inline, inherited]`

Invalidates all iterators.

Definition at line 215 of file `safe_base.h`.

5.253.3.5 `void __gnu_debug::Safe_sequence< vector< _Tp, _Allocator >
>::_M_invalidate_if(_Predicate __pred) [inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `pred(x)` returns `true`. The user of this routine should be careful not to make copies of the iterators passed to `pred`, as the copies may interfere with the invalidation.

5.253.3.6 `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ()
[protected, inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

5.253.3.7 `void __gnu_debug::Safe_sequence_base::_M_swap (
_Safe_sequence_base & __x) [protected, inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.253.3.8 `void __gnu_debug::Safe_sequence< vector< _Tp, _Allocator >
>::_M_transfer_iter(const _Safe_iterator< _Iterator, vector< _Tp,
_Allocator > > & __x) [inherited]`

Transfers all iterators that reference this memory location to this sequence from whatever sequence they are attached to.

5.253.4 Member Data Documentation

5.253.4.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_
const_iterators [inherited]`

The list of constant iterators that reference this container.

Definition at line 166 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.253.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 163 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_invalidate_single()`, and `__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_iter()`.

5.253.4.3 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable, inherited]

The container version number. This number may never be 0.

Definition at line 169 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

5.254 `std::__declval_protector< _Tp >` Struct Template Reference

`declval`

Static Public Member Functions

- static [add_rvalue_reference](#)< _Tp >::type `__delegate` ()

Static Public Attributes

- static const bool `__stop`

5.254.1 Detailed Description

`template<typename _Tp> struct std::__declval_protector< _Tp >`

declval

Definition at line 672 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.255 `std::__exception_ptr::exception_ptr` Class Reference

An opaque pointer to an arbitrary exception.

Public Member Functions

- `exception_ptr` (const [exception_ptr](#) &) throw ()
- `exception_ptr` ([exception_ptr](#) &&__o) throw ()
- `exception_ptr` (nullptr_t) throw ()
- const [type_info](#) * `__cxa_exception_type` () const __attribute__((__pure__)) throw ()
- `operator bool` () const
- [exception_ptr](#) & `operator=` (const [exception_ptr](#) &) throw ()
- [exception_ptr](#) & `operator=` ([exception_ptr](#) &&__o) throw ()
- void `swap` ([exception_ptr](#) &) throw ()

Friends

- bool `operator==` (const [exception_ptr](#) &, const [exception_ptr](#) &) __attribute__((__pure__)) throw ()
- [exception_ptr](#) `std::current_exception` () throw ()
- void `std::rethrow_exception` ([exception_ptr](#))

5.255.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 73 of file `exception_ptr.h`.

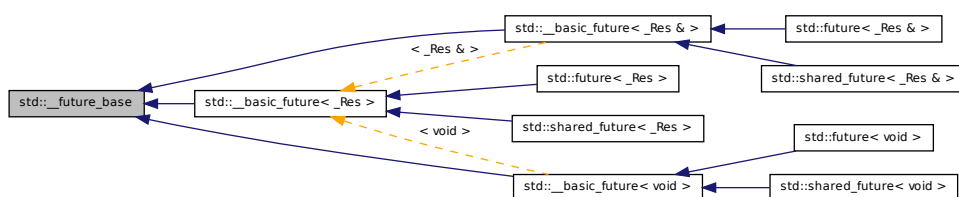
The documentation for this class was generated from the following file:

- [exception_ptr.h](#)

5.256 std::__future_base Struct Reference

Base class and enclosing scope.

Inheritance diagram for std::__future_base:



Classes

- [struct _Ptr](#)
A [unique_ptr](#) based on the instantiating type.
- [struct _Result](#)
Result.
- [struct _Result<_Res & >](#)
Partial specialization for reference types.
- [struct _Result<void >](#)
Explicit specialization for void.
- [struct _Result_alloc](#)
Result_alloc.
- [struct _Result_base](#)
Base class for results.
- [class _State](#)
Shared state between a promise and one or more associated futures.

Static Public Member Functions

- `template<typename _Res, typename _Allocator>
static _Ptr<_Result_alloc< _Res, _Allocator > >::type _S_allocate_result
(const _Allocator &__a)`

5.256.1 Detailed Description

Base class and enclosing scope.

Definition at line 136 of file `future`.

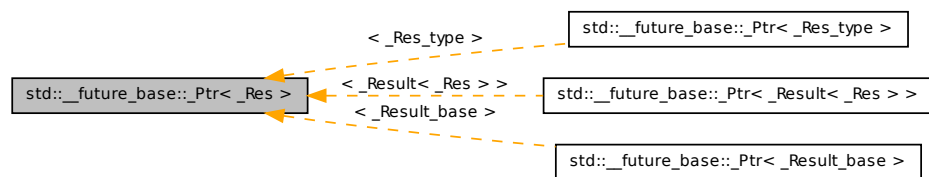
The documentation for this struct was generated from the following file:

- [future](#)

5.257 `std::__future_base::_Ptr< _Res >` Struct Template Reference

A [unique_ptr](#) based on the instantiating type.

Inheritance diagram for `std::__future_base::_Ptr< _Res >`:



Public Types

- `typedef unique_ptr< _Res, _Result_base::_Deleter > type`

5.257.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Ptr< _Res >`

A [unique_ptr](#) based on the instantiating type.

Definition at line 211 of file `future`.

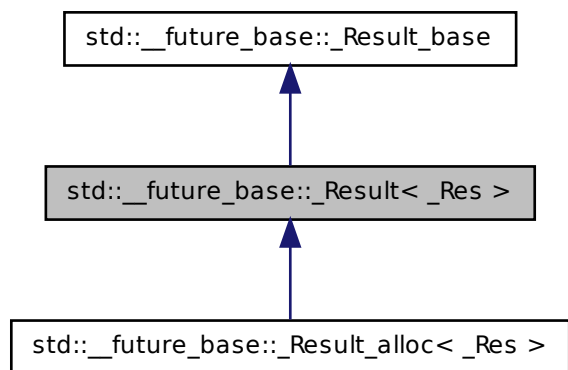
The documentation for this struct was generated from the following file:

- [future](#)

5.258 `std::__future_base::_Result< _Res >` Struct Template Reference

Result.

Inheritance diagram for `std::__future_base::_Result< _Res >`:



Public Member Functions

- `void _M_set (const _Res &__res)`
- `void _M_set (_Res &&__res)`
- `_Res & _M_value ()`

Public Attributes

- `exception_ptr _M_error`

5.258.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Result< _Res >`

Result.

Definition at line 161 of file `future`.

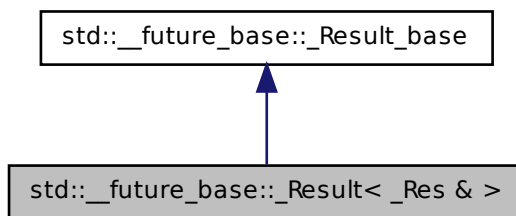
The documentation for this struct was generated from the following file:

- [future](#)

5.259 `std::__future_base::_Result< _Res & >` Struct Template Reference

Partial specialization for reference types.

Inheritance diagram for `std::__future_base::_Result< _Res & >`:



Public Member Functions

- `_Res & _M_get ()`
- `void _M_set (_Res &__res)`

Public Attributes

- `exception_ptr _M_error`

5.259.1 Detailed Description

`template<typename _Res> struct std::__future_base::_Result< _Res & >`

Partial specialization for reference types.

Definition at line 450 of file `future`.

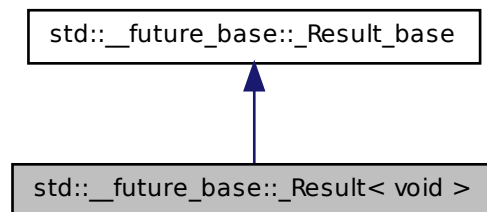
The documentation for this struct was generated from the following file:

- [future](#)

5.260 `std::__future_base::_Result< void >` Struct Template Reference

Explicit specialization for `void`.

Inheritance diagram for `std::__future_base::_Result< void >`:



Public Attributes

- `exception_ptr _M_error`

5.260.1 Detailed Description

template<> struct std::__future_base::_Result< void >

Explicit specialization for void.

Definition at line 466 of file future.

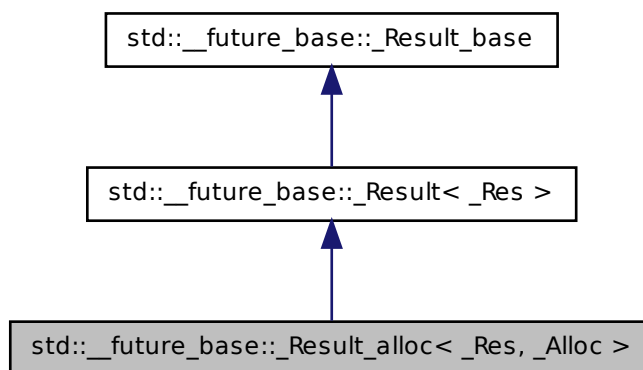
The documentation for this struct was generated from the following file:

- [future](#)

5.261 std::__future_base::_Result_alloc< _Res, _Alloc > Struct Template Reference

Result_alloc.

Inheritance diagram for std::__future_base::_Result_alloc< _Res, _Alloc >:



Public Types

- `typedef _Alloc::template rebind< _Result_alloc >::other __allocator_type`

Public Member Functions

- `_Result_alloc` (const _Alloc &__a)
- `void _M_set` (_Res &&__res)
- `void _M_set` (const _Res &__res)
- `_Res & _M_value` ()

Public Attributes

- `exception_ptr _M_error`

5.261.1 Detailed Description

`template<typename _Res, typename _Alloc> struct std::__future_base::_Result_alloc<_Res, _Alloc>`

Result_alloc.

Definition at line 218 of file future.

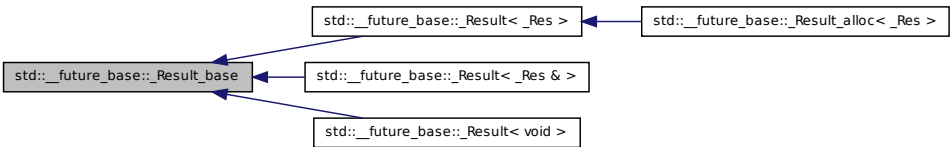
The documentation for this struct was generated from the following file:

- [future](#)

5.262 std::__future_base::_Result_base Struct Reference

Base class for results.

Inheritance diagram for std::__future_base::_Result_base:



Public Member Functions

- `_Result_base` (const [_Result_base](#) &)
- virtual void `_M_destroy` ()=0
- [_Result_base](#) & `operator=` (const [_Result_base](#) &)

Public Attributes

- exception_ptr `_M_error`

5.262.1 Detailed Description

Base class for results.

Definition at line 139 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

5.263 `std::__future_base::__State` Class Reference

Shared state between a promise and one or more associated futures.

Inherited by `std::__future_base::__Async_state< _Res >`, `std::__future_base::__Deferred_state< _Res >`, and `std::__future_base::__Task_state< _Res(_Args...)>`.

Public Member Functions

- `_State` (const [_State](#) &)
- void `_M_break_promise` (`_Ptr_type` __res)
- void `_M_set_result` (function< `_Ptr_type`()> __res, bool __ignore_failure=false)
- void `_M_set_retrieved_flag` ()
- [_State](#) & `operator=` (const [_State](#) &)
- [_Result_base](#) & `wait` ()
- template<typename `_Rep`, typename `_Period` >
bool `wait_for` (const [chrono::duration](#)< `_Rep`, `_Period` > &__rel)
- template<typename `_Clock`, typename `_Duration` >
bool `wait_until` (const [chrono::time_point](#)< `_Clock`, `_Duration` > &__abs)

Static Public Member Functions

- `template<typename _Res, typename _Arg >`
`static _Setter< _Res, _Arg && > __setter (promise< _Res > *__prom, _Arg &&__arg)`
- `template<typename _Res >`
`static _Setter< _Res, __exception_ptr_tag > __setter (exception_ptr &__ex, promise< _Res > *__prom)`
- `static _Setter< void, void > __setter (promise< void > *__prom)`
- `template<typename _Tp >`
`static bool _S_check (const shared_ptr< _Tp > &__p)`

5.263.1 Detailed Description

Shared state between a promise and one or more associated futures.

Definition at line 257 of file future.

The documentation for this class was generated from the following file:

- [future](#)

5.264 std::__is_location_invariant< _Tp > Struct Template Reference

Inherits `integral_constant< bool,(is_pointer< _Tp >::value||is_member_pointer< _Tp >::value)>`.

Inherited by `std::__is_location_invariant< _Simple_type_wrapper< _Tp > >`.

5.264.1 Detailed Description

`template<typename _Tp> struct std::__is_location_invariant< _Tp >`

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Also implies a trivial copy constructor and assignment operator.

Definition at line 1479 of file functional.

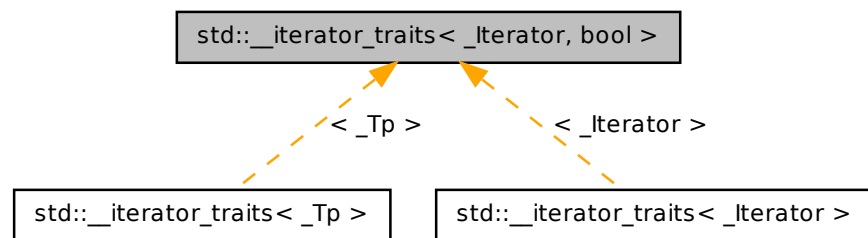
The documentation for this struct was generated from the following file:

- [functional](#)

5.265 `std::__iterator_traits< _Iterator, bool >` Struct Template Reference

Traits class for iterators.

Inheritance diagram for `std::__iterator_traits< _Iterator, bool >`:



5.265.1 Detailed Description

```
template<typename _Iterator,    bool    = __has_iterator_category<_
Iterator>::value> struct std::__iterator_traits< _Iterator, bool >
```

Traits class for iterators. This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the `Iterator` argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

Definition at line 145 of file `stl_iterator_base_types.h`.

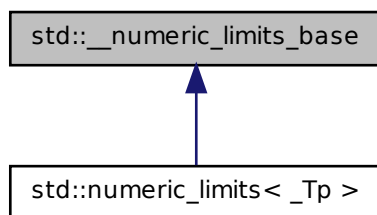
The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.266 `std::__numeric_limits_base` Struct Reference

Part of [std::numeric_limits](#).

Inheritance diagram for std::__numeric_limits_base:



Static Public Attributes

- static const int [digits](#)
- static const int [digits10](#)
- static const [float_denorm_style](#) [has_denorm](#)
- static const bool [has_denorm_loss](#)
- static const bool [has_infinity](#)
- static const bool [has_quiet_NaN](#)
- static const bool [has_signaling_NaN](#)
- static const bool [is_bounded](#)
- static const bool [is_exact](#)
- static const bool [is_iec559](#)
- static const bool [is_integer](#)
- static const bool [is_modulo](#)
- static const bool [is_signed](#)
- static const bool [is_specialized](#)
- static const int [max_digits10](#)
- static const int [max_exponent](#)
- static const int [max_exponent10](#)
- static const int [min_exponent](#)
- static const int [min_exponent10](#)
- static const int [radix](#)
- static const [float_round_style](#) [round_style](#)
- static const bool [tinyness_before](#)
- static const bool [traps](#)

5.266.1 Detailed Description

Part of [std::numeric_limits](#). The `static const` members are usable as integral constant expressions.

Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the [std::numeric_limits](#) class.

Definition at line 190 of file `limits`.

5.266.2 Member Data Documentation

5.266.2.1 `const int std::__numeric_limits_base::digits` `[static]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 199 of file `limits`.

5.266.2.2 `const int std::__numeric_limits_base::digits10` `[static]`

The number of base 10 digits that can be represented without change.

Definition at line 201 of file `limits`.

5.266.2.3 `const float_denorm_style std::__numeric_limits_base::has_denorm` `[static]`

See [std::float_denorm_style](#) for more information.

Definition at line 244 of file `limits`.

5.266.2.4 `const bool std::__numeric_limits_base::has_denorm_loss` `[static]`

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result. [18.2.1.2]/42

Definition at line 247 of file `limits`.

5.266.2.5 const bool std::__numeric_limits_base::has_infinity [static]

True if the type has a representation for positive infinity.

Definition at line 236 of file limits.

5.266.2.6 const bool std::__numeric_limits_base::has_quiet_NaN [static]

True if the type has a representation for a quiet (non-signaling) *Not a Number*.

Definition at line 239 of file limits.

5.266.2.7 const bool std::__numeric_limits_base::has_signaling_NaN [static]

True if the type has a representation for a signaling *Not a Number*.

Definition at line 242 of file limits.

5.266.2.8 const bool std::__numeric_limits_base::is_bounded [static]

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types. [18.2.1.2]/54

Definition at line 255 of file limits.

5.266.2.9 const bool std::__numeric_limits_base::is_exact [static]

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer. [18.2.1.2]/15

Definition at line 216 of file limits.

5.266.2.10 const bool std::__numeric_limits_base::is_iec559 [static]

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 251 of file limits.

5.266.2.11 const bool std::__numeric_limits_base::is_integer [static]

True if the type is integer. Is this supposed to be *if the type is integral*?

Definition at line 211 of file limits.

5.266.2.12 `const bool std::__numeric_limits_base::is_modulo [static]`

True if the type is *modulo*, that is, if it is possible to add two positive numbers and have a result that wraps around to a third number that is less. Typically false for floating types, true for unsigned integers, and true for signed integers.

Definition at line 260 of file limits.

5.266.2.13 `const bool std::__numeric_limits_base::is_signed [static]`

True if the type is signed.

Definition at line 208 of file limits.

5.266.2.14 `const bool std::__numeric_limits_base::is_specialized [static]`

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 194 of file limits.

5.266.2.15 `const int std::__numeric_limits_base::max_digits10 [static]`

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 205 of file limits.

5.266.2.16 `const int std::__numeric_limits_base::max_exponent [static]`

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 230 of file limits.

5.266.2.17 `const int std::__numeric_limits_base::max_exponent10 [static]`

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 233 of file limits.

5.266.2.18 const int std::__numeric_limits_base::min_exponent [static]

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 223 of file `limits`.

5.266.2.19 const int std::__numeric_limits_base::min_exponent10 [static]

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 226 of file `limits`.

5.266.2.20 const int std::__numeric_limits_base::radix [static]

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 219 of file `limits`.

5.266.2.21 const float_round_style std::__numeric_limits_base::round_style [static]

See [std::float_round_style](#) for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 269 of file `limits`.

5.266.2.22 const bool std::__numeric_limits_base::tinyness_before [static]

True if tininess is detected before rounding. (see IEC 559)

Definition at line 265 of file `limits`.

5.266.2.23 const bool std::__numeric_limits_base::traps [static]

True if trapping is implemented for this type.

Definition at line 263 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.267 `std::__parallel::CRandNumber< _MustBeInt >` Struct Template Reference

Functor wrapper for `std::rand()`.

Public Member Functions

- `int operator() (int __limit)`

5.267.1 Detailed Description

```
template<typename _MustBeInt = int> struct std::__parallel::_-
CRandNumber< _MustBeInt >
```

Functor wrapper for `std::rand()`.

Definition at line 1656 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

5.268 `std::__profile::bitset< _Nb >` Class Template Reference

Class [std::bitset](#) wrapper with performance instrumentation.

Inherits `bitset< _Nb >`.

Public Member Functions

- `bitset (unsigned long long __val)`
- `template<class _CharT, class _Traits, class _Alloc >`
`bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__str, type-`
`name std::basic_string< _CharT, _Traits, _Alloc >::size_type __pos, typename`
`std::basic_string< _CharT, _Traits, _Alloc >::size_type __n, _CharT __zero,`
`_CharT __one=_CharT('1'))`
- `bitset (const _Base &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__str, type-`
`name std::basic_string< _CharT, _Traits, _Alloc >::size_type __pos=0, type-`

- name `std::basic_string<_CharT, _Traits, _Alloc>::size_type __n=(std::basic_string<_CharT, _Traits, _Alloc>::npos)`
- `bitset` (const char *__str)
 - `_Base & _M_base ()`
 - `const _Base & _M_base () const`
 - `bitset<_Nb> & flip ()`
 - `bitset<_Nb> & flip (size_t __pos)`
 - `bool operator!= (const bitset<_Nb> &__rhs) const`
 - `bitset<_Nb> & operator&= (const bitset<_Nb> &__rhs)`
 - `bitset<_Nb> operator<< (size_t __pos) const`
 - `bitset<_Nb> & operator<<= (size_t __pos)`
 - `bool operator== (const bitset<_Nb> &__rhs) const`
 - `bitset<_Nb> operator>> (size_t __pos) const`
 - `bitset<_Nb> & operator>>= (size_t __pos)`
 - reference `operator[]` (size_t __pos)
 - `bool operator[]` (size_t __pos) const
 - `bitset<_Nb> & operator^= (const bitset<_Nb> &__rhs)`
 - `bitset<_Nb> & operator|= (const bitset<_Nb> &__rhs)`
 - `bitset<_Nb> operator~ () const`
 - `bitset<_Nb> & reset ()`
 - `bitset<_Nb> & reset (size_t __pos)`
 - `bitset<_Nb> & set (size_t __pos, bool __val=true)`
 - `bitset<_Nb> & set ()`
 - `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_string` (char __zero, char __one= '1') const
 - `template<typename _CharT, typename _Traits>`
`std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string ()`
const
 - `template<typename _CharT, typename _Traits, typename _Alloc>`
`std::basic_string<_CharT, _Traits, _Alloc> to_string ()` const
 - `template<class _CharT, class _Traits>`
`std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string` (_CharT __zero, _CharT __one=_CharT('1')) const
 - `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_string` () const
 - `template<typename _CharT>`
`std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>> to_string` () const
 - `template<class _CharT>`
`std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>> to_string` (_CharT __zero, _CharT __one=_CharT('1')) const
 - `template<class _CharT, class _Traits, class _Alloc>`
`std::basic_string<_CharT, _Traits, _Alloc> to_string` (_CharT __zero, _CharT __one=_CharT('1')) const

5.268.1 Detailed Description

`template<size_t _Nb> class std::__profile::bitset< _Nb >`

Class [std::bitset](#) wrapper with performance instrumentation.

Definition at line 40 of file `profile/bitset`.

The documentation for this class was generated from the following file:

- [profile/bitset](#)

5.269 `std::__profile::deque< _Tp, _Allocator >` Class Template Reference

Class [std::deque](#) wrapper with performance instrumentation.

Inherits `deque< _Tp, _Allocator >`.

Public Types

- typedef `_Allocator` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `_Base::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **deque** (`const _Allocator &__a=_Allocator()`)
- **deque** (`size_type __n`)
- `template<class _InputIterator >`
deque (`_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator()`)
- **deque** ([initializer_list](#)< `value_type` > `__l`, `const allocator_type &__a=allocator_type()`)

- **deque** (const [deque](#) &__x)
- **deque** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- **deque** (const [_Base](#) &__x)
- **deque** ([deque](#) &&__x)
- [_Base](#) & [_M_base](#) ()
- const [_Base](#) & [_M_base](#) () const
- template<class _InputIterator >
void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__t)
- void **assign** ([initializer_list](#)< value_type > __l)
- reference **back** ()
- const_reference **back** () const
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- template<typename... _Args>
iterator **emplace** (iterator __position, _Args &&...__args)
- template<typename... _Args>
void **emplace_back** (_Args &&...__args)
- template<typename... _Args>
void **emplace_front** (_Args &&...__args)
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __position)
- reference **front** ()
- const_reference **front** () const
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __p, [initializer_list](#)< value_type > __l)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- iterator **insert** (iterator __position, const _Tp &__x)
- template<class _InputIterator >
void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- [deque](#) & **operator=** (const [deque](#) &__x)
- [deque](#) & **operator=** ([deque](#) &&__x)
- [deque](#) & **operator=** ([initializer_list](#)< value_type > __l)
- const_reference **operator[]** (size_type __n) const
- reference **operator[]** (size_type __n)

- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (const _Tp &__x)
- void **push_back** (_Tp &&__x)
- void **push_front** (_Tp &&__x)
- void **push_front** (const _Tp &__x)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- const_reverse_iterator **rend** () const
- reverse_iterator **rend** ()
- void **resize** (size_type __sz, const _Tp &__c)
- void **resize** (size_type __sz)
- void **swap** (deque &__x)

5.269.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class
std::__profile::deque< _Tp, _Allocator >

Class [std::deque](#) wrapper with performance instrumentation.

Definition at line 40 of file profile/deque.

The documentation for this class was generated from the following file:

- [profile/deque](#)

5.270 std::__profile::list< _Tp, _Allocator > Class Template Reference

List wrapper with performance instrumentation.

Inherits list< _Tp, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef __iterator_tracker< typename _Base::const_iterator, [list](#) > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef [std::reverse_iterator](#)< const_iterator > **const_reverse_iterator**

- typedef _Base::difference_type **difference_type**
- typedef __iterator_tracker< typename _Base::iterator, [list](#) > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef [std::reverse_iterator](#)< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **list** (const _Allocator &__a=_Allocator())
- **list** (size_type __n)
- template<class _InputIterator >
 list (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **list** ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- **list** (const [list](#) &__x)
- **list** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- **list** (const [_Base](#) &__x)
- **list** ([list](#) &&__x)
- const [_Base](#) & **_M_base** () const
- [_Base](#) & **_M_base** ()
- void **_M_profile_find** () const
- void **_M_profile_iterate** (int __rewind=0) const
- void **assign** ([initializer_list](#)< value_type > __l)
- template<class _InputIterator >
 void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__t)
- const_reference **back** () const
- reference **back** ()
- const_iterator **begin** () const
- iterator **begin** ()
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- [const_reverse_iterator](#) **crbegin** () const
- [const_reverse_iterator](#) **crend** () const
- template<typename... _Args>
 iterator **emplace** (iterator __position, _Args &&...__args)
- iterator **end** ()
- const_iterator **end** () const

- iterator **erase** (iterator __position)
- iterator **erase** (iterator __position, iterator __last)
- const_reference **front** () const
- reference **front** ()
- iterator **insert** (iterator __position, const _Tp &__x)
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- void **insert** (iterator __position, [initializer_list](#)< value_type > __l)
- template<class _InputIterator >
void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- template<class _Compare >
void **merge** ([list](#) &&__x, _Compare __comp)
- template<typename _Compare >
void **merge** ([list](#) &&__x, _Compare __comp)
- void **merge** ([list](#) &&__x)
- void **merge** ([list](#) &__x)
- [list](#) & **operator=** (const [list](#) &__x)
- [list](#) & **operator=** ([list](#) &&__x)
- [list](#) & **operator=** ([initializer_list](#)< value_type > __l)
- void **pop_back** ()
- void **pop_front** ()
- void **push_front** (const value_type &__x)
- [const_reverse_iterator](#) **rbegin** () const
- [reverse_iterator](#) **rbegin** ()
- void **remove** (const _Tp &__value)
- template<class _Predicate >
void **remove_if** (_Predicate __pred)
- [const_reverse_iterator](#) **rend** () const
- [reverse_iterator](#) **rend** ()
- void **resize** (size_type __sz, const _Tp &__c)
- void **resize** (size_type __sz)
- template<typename _StrictWeakOrdering >
void **sort** (_StrictWeakOrdering __pred)
- void **sort** ()
- void **splice** (iterator __position, [list](#) &&__x)
- void **splice** (iterator __position, [list](#) &&__x, iterator __first, iterator __last)
- void **splice** (iterator __position, [list](#) &__x, iterator __i)
- void **splice** (iterator __position, [list](#) &&__x, iterator __i)
- void **splice** (iterator __position, [list](#) &__x, iterator __first, iterator __last)
- void **splice** (iterator __position, [list](#) &__x)
- void **swap** ([list](#) &__x)
- template<class _BinaryPredicate >
void **unique** (_BinaryPredicate __binary_pred)
- void **unique** ()

5.270.1 Detailed Description

`template<typename _Tp, typename _Allocator = std::allocator<_Tp>> class std::__profile::list< _Tp, _Allocator >`

List wrapper with performance instrumentation.

Definition at line 42 of file `profile/list`.

The documentation for this class was generated from the following file:

- [profile/list](#)

5.271 `std::__profile::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Class `std::map` wrapper with performance instrumentation.

Inherits `map< _Key, _Tp, _Compare, _Allocator >`.

Public Types

- typedef `_Allocator` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- **map** (`const _Compare &__comp=_Compare()`, `const _Allocator &__a=_Allocator()`)

- `template<typename _InputIterator >`
`map (_InputIterator __first, _InputIterator __last, const _Compare &__comp= _`
`Compare(), const _Allocator &__a=_Allocator())`
- `map (const _Base &__x)`
- `map (map &&__x)`
- `map (const map &__x)`
- `map (initializer_list< value_type > __l, const _Compare &__c=_Compare(),`
`const allocator_type &__a=allocator_type())`
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `mapped_type & at (const key_type &__k)`
- `const mapped_type & at (const key_type &__k) const`
- `iterator begin ()`
- `const_iterator begin () const`
- `const_iterator cbegin () const`
- `const_iterator cend () const`
- `void clear ()`
- `size_type count (const key_type &__x) const`
- `const_reverse_iterator crbegin () const`
- `const_reverse_iterator crend () const`
- `iterator end ()`
- `const_iterator end () const`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x)`
`const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `size_type erase (const key_type &__x)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `template<typename _InputIterator >`
`void insert (_InputIterator __first, _InputIterator __last)`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `iterator insert (iterator __position, const value_type &__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `map & operator= (const map &__x)`
- `map & operator= (initializer_list< value_type > __l)`
- `map & operator= (map &&__x)`
- `mapped_type & operator[] (const key_type &__k)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`

- [reverse_iterator](#) **rend** ()
- [const_reverse_iterator](#) **rend** () const
- void **swap** ([map](#) &__x)
- const_iterator **upper_bound** (const key_type &__x) const
- iterator **upper_bound** (const key_type &__x)

5.271.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::map< _Key, _Tp, _Compare, _Allocator >
```

Class [std::map](#) wrapper with performance instrumentation.

Definition at line 47 of file profile/map.h.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

5.272 **std::__profile::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference**

Class [std::multimap](#) wrapper with performance instrumentation.

Inherits `multimap< _Key, _Tp, _Compare, _Allocator >`.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::const_reverse_iterator **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**

- typedef `_Base::reverse_iterator` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- **multimap** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
multimap (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **multimap** (const `_Base` &__x)
- **multimap** (`multimap` &&__x)
- **multimap** (const `multimap` &__x)
- **multimap** (`initializer_list`< `value_type` > __l, const `_Compare` &__c=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- iterator **end** ()
- const_iterator **end** () const
- `std::pair`< iterator, iterator > **equal_range** (const key_type &__x)
- `std::pair`< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- iterator **insert** (const `value_type` &__x)
- void **insert** (`std::initializer_list`< `value_type` > __list)
- iterator **insert** (iterator __position, const `value_type` &__x)
- template<typename `_InputIterator` >
void **insert** (`_InputIterator` __first, `_InputIterator` __last)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- `multimap` & **operator=** (const `multimap` &__x)

5.273 `std::__profile::multiset< _Key, _Compare, _Allocator >` Class Template Reference 1425

- `multimap` & `operator=` (`initializer_list`< `value_type` > `__l`)
- `multimap` & `operator=` (`multimap` &&`__x`)
- `const_reverse_iterator` `rbegin` () `const`
- `reverse_iterator` `rbegin` ()
- `reverse_iterator` `rend` ()
- `const_reverse_iterator` `rend` () `const`
- `void` `swap` (`multimap` &`__x`)
- `iterator` `upper_bound` (`const key_type` &`__x`)
- `const_iterator` `upper_bound` (`const key_type` &`__x`) `const`

5.272.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class `std::multimap` wrapper with performance instrumentation.

Definition at line 41 of file `profile/multimap.h`.

The documentation for this class was generated from the following file:

- [profile/multimap.h](#)

5.273 `std::__profile::multiset< _Key, _Compare, _Allocator >` Class Template Reference

Class `std::multiset` wrapper with performance instrumentation.

Inherits `multiset< _Key, _Compare, _Allocator >`.

Public Types

- `typedef` `_Allocator` `allocator_type`
- `typedef` `_Base::const_iterator` `const_iterator`
- `typedef` `_Base::const_pointer` `const_pointer`
- `typedef` `_Base::const_reference` `const_reference`
- `typedef` `_Base::const_reverse_iterator` `const_reverse_iterator`
- `typedef` `_Base::difference_type` `difference_type`
- `typedef` `_Base::iterator` `iterator`
- `typedef` `_Compare` `key_compare`
- `typedef` `_Key` `key_type`

- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Compare` **value_compare**
- typedef `_Key` **value_type**

Public Member Functions

- **multiset** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
multiset (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **multiset** (const `_Base` &__x)
- **multiset** (`multiset` &&__x)
- **multiset** (const `multiset` &__x)
- **multiset** (`initializer_list`< `value_type` > __l, const `_Compare` &__comp=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- const_iterator **end** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- iterator **end** ()
- const_iterator **end** () const
- `std::pair`< iterator, iterator > **equal_range** (const `key_type` &__x)
- `std::pair`< const_iterator, const_iterator > **equal_range** (const `key_type` &__x) const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __position)
- size_type **erase** (const `key_type` &__x)
- iterator **find** (const `key_type` &__x)
- const_iterator **find** (const `key_type` &__x) const
- iterator **insert** (const `value_type` &__x)
- iterator **insert** (iterator __position, const `value_type` &__x)
- template<typename `_InputIterator` >
void **insert** (`_InputIterator` __first, `_InputIterator` __last)
- void **insert** (`initializer_list`< `value_type` > __l)

- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- [multiset](#) & **operator=** (const [multiset](#) &__x)
- [multiset](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [multiset](#) & **operator=** ([multiset](#) &&__x)
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rbegin** ()
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- void **swap** ([multiset](#) &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

5.273.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__profile::multiset< _Key, _Compare, _Allocator >

Class [std::multiset](#) wrapper with performance instrumentation.

Definition at line 41 of file profile/multiset.h.

The documentation for this class was generated from the following file:

- [profile/multiset.h](#)

5.274 std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference

Class [std::set](#) wrapper with performance instrumentation.

Inherits [set< _Key, _Compare, _Allocator >](#).

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::const_reverse_iterator **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**

- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Compare` **value_compare**
- typedef `_Key` **value_type**

Public Member Functions

- **set** (const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
 set (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **set** (const `_Base` &__x)
- **set** (`set` &&__x)
- **set** (const `set` &__x)
- **set** (`initializer_list`< `value_type` > __l, const `_Compare` &__comp=_Compare(), const `allocator_type` &__a=allocator_type())
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **cbegin** () const
- const_iterator **end** () const
- void **clear** ()
- const_reverse_iterator **crbegin** () const
- const_reverse_iterator **crend** () const
- iterator **end** ()
- const_iterator **end** () const
- `std::pair`< iterator, iterator > **equal_range** (const `key_type` &__x)
- `std::pair`< const_iterator, const_iterator > **equal_range** (const `key_type` &__x) const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __position)
- `size_type` **erase** (const `key_type` &__x)
- iterator **find** (const `key_type` &__x)
- const_iterator **find** (const `key_type` &__x) const
- `std::pair`< iterator, bool > **insert** (const `value_type` &__x)

5.275 `std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference 1429

- iterator **insert** (iterator __position, const value_type &__x)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** (initializer_list< value_type > __l)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- [set](#) & **operator=** (const [set](#) &__x)
- [set](#) & **operator=** (initializer_list< value_type > __l)
- [set](#) & **operator=** ([set](#) &&__x)
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rbegin** ()
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- void **swap** ([set](#) &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

5.274.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>> class std::__profile::set< _Key, _Compare, _Allocator >

Class [std::set](#) wrapper with performance instrumentation.

Definition at line 41 of file [profile/set.h](#).

The documentation for this class was generated from the following file:

- [profile/set.h](#)

5.275 `std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class [std::unordered_map](#) wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_PR::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Base::const_iterator **const_iterator**

- typedef `_Base::const_reference` **const_reference**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::hasher` **hasher**
- typedef `_Base::iterator` **iterator**
- typedef `_Base::key_equal` **key_equal**
- typedef `_Base::key_type` **key_type**
- typedef `_Base::mapped_type` **mapped_type**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size_type**
- typedef `_Base::value_type` **value_type**

Public Member Functions

- **unordered_map** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_map (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** ([unordered_map](#) &&__x)
- **unordered_map** ([initializer_list](#)< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (const `_Base` &__x)
- `_Base` & **_M_base** ()
- const `_Base` & **_M_base** () const
- void **clear** ()
- template<typename _InputIter >
void **insert** (_InputIter __first, _InputIter __last)
- void **insert** (const value_type * __first, const value_type * __last)
- iterator **insert** (const_iterator __iter, const value_type &__v)
- void **insert** ([std::initializer_list](#)< value_type > __l)
- [std::pair](#)< iterator, bool > **insert** (const value_type &__obj)
- [unordered_map](#) & **operator=** ([unordered_map](#) &&__x)
- [unordered_map](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_map](#) & **operator=** (const [unordered_map](#) &__x)
- mapped_type & **operator[]** (const `_Key` &__k)
- void **rehash** (size_type __n)
- void **swap** ([unordered_map](#) &__x)

5.275.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>
```

Class `std::unordered_map` wrapper with performance instrumentation.

Definition at line 56 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered_map](#)

5.276 `std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Class `std::unordered_multimap` wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_PR::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::difference_type difference_type`
- `typedef _Base::hasher hasher`
- `typedef _Base::iterator iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

Public Member Functions

- `unordered_multimap (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())`

- `template<typename _InputIterator >`
unordered_multimap (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered_multimap** (`unordered_multimap &&__x`)
- **unordered_multimap** (`initializer_list< value_type > __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered_multimap** (`const _Base &__x`)
- `_Base & _M_base ()`
- `const _Base & _M_base () const`
- `void clear ()`
- `void insert (const value_type *__first, const value_type *__last)`
- `template<typename _InputIter >`
`void insert (_InputIter __first, _InputIter __last)`
- `iterator insert (const_iterator __iter, const value_type &__v)`
- `void insert (std::initializer_list< value_type > __l)`
- `iterator insert (const value_type &__obj)`
- `unordered_multimap & operator= (unordered_multimap &&__x)`
- `unordered_multimap & operator= (initializer_list< value_type > __l)`
- `unordered_multimap & operator= (const unordered_multimap &__x)`
- `void rehash (size_type __n)`
- `void swap (unordered_multimap &__x)`

5.276.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>,
typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multimap` wrapper with performance instrumentation.

Definition at line 296 of file `profile/unordered_map`.

The documentation for this class was generated from the following file:

- [profile/unordered_map](#)

5.277 `std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

`Unordered_multiset` wrapper with performance instrumentation.

Inherits `_GLIBCXX_STD_BASE`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Base::const_iterator const_iterator`
- `typedef _Base::const_reference const_reference`
- `typedef _Base::difference_type difference_type`
- `typedef _Base::hasher hasher`
- `typedef _Base::iterator iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Base::key_type key_type`
- `typedef _Base::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

Public Member Functions

- **unordered_multiset** (`size_type __n=10`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- `template<typename _InputIterator >`
unordered_multiset (`_InputIterator __f`, `_InputIterator __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered_multiset** (`unordered_multiset &&__x`)
- **unordered_multiset** (`initializer_list< value_type > __l`, `size_type __n=0`, `const hasher &__hf=hasher()`, `const key_equal &__eq=key_equal()`, `const allocator_type &__a=allocator_type()`)
- **unordered_multiset** (`const _Base &__x`)
- `void clear ()`
- `template<typename _InputIter >`
`void insert (_InputIter __first, _InputIter __last)`
- `void insert (const value_type *__first, const value_type *__last)`
- `void insert (std::initializer_list< value_type > __l)`
- `iterator insert (const value_type &__obj)`
- `iterator insert (const_iterator __iter, const value_type &__v)`
- `unordered_multiset & operator= (initializer_list< value_type > __l)`
- `unordered_multiset & operator= (unordered_multiset &&__x)`
- `unordered_multiset & operator= (const unordered_multiset &__x)`
- `void rehash (size_type __n)`
- `void swap (unordered_multiset &__x)`

5.277.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename
_Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

Unordered_multiset wrapper with performance instrumentation.

Definition at line 284 of file profile/unordered_set.

The documentation for this class was generated from the following file:

- [profile/unordered_set](#)

5.278 std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference

Unordered_set wrapper with performance instrumentation.

Inherits _GLIBCXX_STD_BASE.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef _Base::iterator **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_set** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_set (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())

- **unordered_set** ([unordered_set](#) &&__x)
- **unordered_set** ([initializer_list](#)< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (const _Base &__x)
- void **clear** ()
- template<typename _InputIter >
void **insert** (_InputIter __first, _InputIter __last)
- void **insert** (const value_type *__first, const value_type *__last)
- void **insert** ([std::initializer_list](#)< value_type > __l)
- [std::pair](#)< iterator, bool > **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __iter, const value_type &__v)
- [unordered_set](#) & **operator=** ([initializer_list](#)< value_type > __l)
- [unordered_set](#) & **operator=** ([unordered_set](#) &&__x)
- [unordered_set](#) & **operator=** (const [unordered_set](#) &__x)
- void **rehash** (size_type __n)
- void **swap** ([unordered_set](#) &__x)

5.278.1 Detailed Description

template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>> class std::_profile::unordered_set< _Key, _Hash, _Pred, _Alloc >

Unordered_set wrapper with performance instrumentation.

Definition at line 56 of file profile/unordered_set.

The documentation for this class was generated from the following file:

- [profile/unordered_set](#)

5.279 std::_Base_bitset< _Nw > Struct Template Reference

Inheritance diagram for std::_Base_bitset< _Nw >:



Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- **_Base_bitset** (unsigned long long __val)
- size_t **_M_are_all_aux** () const
- void **_M_do_and** (const [_Base_bitset](#)< _Nw > &__x)
- size_t **_M_do_count** () const
- size_t **_M_do_find_first** (size_t __not_found) const
- size_t **_M_do_find_next** (size_t __prev, size_t __not_found) const
- void **_M_do_flip** ()
- void **_M_do_left_shift** (size_t __shift)
- void **_M_do_or** (const [_Base_bitset](#)< _Nw > &__x)
- void **_M_do_reset** ()
- void **_M_do_right_shift** (size_t __shift)
- void **_M_do_set** ()
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const [_Base_bitset](#)< _Nw > &__x)
- const _WordT * **_M_getdata** () const
- _WordT **_M_getword** (size_t __pos) const
- _WordT & **_M_getword** (size_t __pos)
- _WordT **_M_hiword** () const
- _WordT & **_M_hiword** ()
- bool **_M_is_any** () const
- bool **_M_is_equal** (const [_Base_bitset](#)< _Nw > &__x) const

Static Public Member Functions

- static _WordT **_S_maskbit** (size_t __pos)
- static size_t **_S_whichbit** (size_t __pos)
- static size_t **_S_whichbyte** (size_t __pos)
- static size_t **_S_whichword** (size_t __pos)

Public Attributes

- _WordT [_M_w](#) [_Nw]

5.279.1 Detailed Description

`template<size_t _Nw> struct std::_Base_bitset< _Nw >`

Base class, general case. It is a class invariant that `_Nw` will be nonnegative.

See documentation for `bitset`.

Definition at line 68 of file `bitset`.

5.279.2 Member Data Documentation

5.279.2.1 `template<size_t _Nw> _WordT std::_Base_bitset< _Nw >::_M_w[_Nw]`

0 is the least significant word.

Definition at line 73 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

5.280 `std::_Base_bitset< 0 >` Struct Template Reference

Public Types

- typedef unsigned long `_WordT`

Public Member Functions

- `_Base_bitset` (unsigned long long)
- `size_t _M_are_all_aux ()` const
- `void _M_do_and` (const [_Base_bitset< 0 >](#) &)
- `size_t _M_do_count ()` const
- `size_t _M_do_find_first` (size_t) const
- `size_t _M_do_find_next` (size_t, size_t) const
- `void _M_do_flip ()`
- `void _M_do_left_shift` (size_t)
- `void _M_do_or` (const [_Base_bitset< 0 >](#) &)
- `void _M_do_reset ()`
- `void _M_do_right_shift` (size_t)

- void **_M_do_set** ()
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const [_Base_bitset](#)< 0 > &)
- **_WordT** & **_M_getword** (size_t) const
- **_WordT** **_M_hiword** () const
- bool **_M_is_any** () const
- bool **_M_is_equal** (const [_Base_bitset](#)< 0 > &) const

Static Public Member Functions

- static **_WordT** **_S_maskbit** (size_t __pos)
- static size_t **_S_whichbit** (size_t __pos)
- static size_t **_S_whichbyte** (size_t __pos)
- static size_t **_S_whichword** (size_t __pos)

5.280.1 Detailed Description

template<> struct std::_Base_bitset< 0 >

Base class, specialization for no storage (zero-length bitset).

See documentation for bitset.

Definition at line 510 of file bitset.

The documentation for this struct was generated from the following file:

- [bitset](#)

5.281 std::_Base_bitset< 1 > Struct Template Reference

Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- **_Base_bitset** (unsigned long long __val)
- size_t **_M_are_all_aux** () const
- void **_M_do_and** (const [_Base_bitset](#)< 1 > &__x)

- `size_t _M_do_count () const`
- `size_t _M_do_find_first (size_t __not_found) const`
- `size_t _M_do_find_next (size_t __prev, size_t __not_found) const`
- `void _M_do_flip ()`
- `void _M_do_left_shift (size_t __shift)`
- `void _M_do_or (const _Base_bitset< 1 > &__x)`
- `void _M_do_reset ()`
- `void _M_do_right_shift (size_t __shift)`
- `void _M_do_set ()`
- `unsigned long long _M_do_to_ullong () const`
- `unsigned long _M_do_to_ulong () const`
- `void _M_do_xor (const _Base_bitset< 1 > &__x)`
- `const _WordT * _M_getdata () const`
- `_WordT _M_getword (size_t) const`
- `_WordT & _M_getword (size_t)`
- `_WordT _M_hiword () const`
- `_WordT & _M_hiword ()`
- `bool _M_is_any () const`
- `bool _M_is_equal (const _Base_bitset< 1 > &__x) const`

Static Public Member Functions

- `static _WordT _S_maskbit (size_t __pos)`
- `static size_t _S_whichbit (size_t __pos)`
- `static size_t _S_whichbyte (size_t __pos)`
- `static size_t _S_whichword (size_t __pos)`

Public Attributes

- `_WordT _M_w`

5.281.1 Detailed Description

template<> struct std::_Base_bitset< 1 >

Base class, specialization for a single word.

See documentation for bitset.

Definition at line 366 of file bitset.

The documentation for this struct was generated from the following file:

- [bitset](#)

5.282 `std::_Build_index_tuple< _Num >` Struct Template Reference

Builds an [_Index_tuple](#)<0, 1, 2, ..., _Num-1>.

Public Types

- typedef [_Build_index_tuple](#)< _Num-1 >::__type::__next __type

5.282.1 Detailed Description

`template<std::size_t _Num> struct std::_Build_index_tuple< _Num >`

Builds an [_Index_tuple](#)<0, 1, 2, ..., _Num-1>.

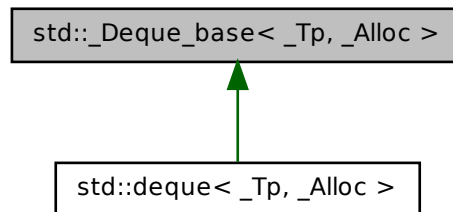
Definition at line 764 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.283 `std::_Deque_base< _Tp, _Alloc >` Class Template Reference

Inheritance diagram for `std::_Deque_base< _Tp, _Alloc >`:



Public Types

- typedef _Alloc **allocator_type**
- typedef [_Deque_iterator](#)< _Tp, const _Tp &, const _Tp * > **const_iterator**
- typedef [_Deque_iterator](#)< _Tp, _Tp &, _Tp * > **iterator**

Public Member Functions

- [_Deque_base](#) (const allocator_type &__a, size_t __num_elements)
- [_Deque_base](#) ([_Deque_base](#) &&__x)
- [_Deque_base](#) (const allocator_type &__a)
- [_Deque_base](#) (size_t __num_elements)
- allocator_type **get_allocator** () const

Protected Types

- enum { **_S_initial_map_size** }
- typedef _Alloc::template rebind< _Tp * >::other **_Map_alloc_type**
- typedef _Alloc::template rebind< _Tp >::other **_Tp_alloc_type**

Protected Member Functions

- _Tp ** **_M_allocate_map** (size_t __n)
- _Tp * **_M_allocate_node** ()
- void **_M_create_nodes** (_Tp **__nstart, _Tp **__nfinish)
- void **_M_deallocate_map** (_Tp **__p, size_t __n)
- void **_M_deallocate_node** (_Tp *__p)
- void **_M_destroy_nodes** (_Tp **__nstart, _Tp **__nfinish)
- _Map_alloc_type **_M_get_map_allocator** () const
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const
- _Tp_alloc_type & **_M_get_Tp_allocator** ()
- void [_M_initialize_map](#) (size_t)

Protected Attributes

- [_Deque_impl](#) **_M_impl**

5.283.1 Detailed Description

```
template<typename _Tp, typename _Alloc> class std::_Deque_base< _Tp, _-
Alloc >
```

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual `Tp` element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 436 of file `stl_deque.h`.

5.283.2 Member Function Documentation

```
5.283.2.1 template<typename _Tp , typename _Alloc > void
std::_Deque_base< _Tp, _Alloc >::_M_initialize_map ( size_t
__num_elements ) [protected]
```

Layout storage.

Parameters

num_elements The count of `T`'s for which to allocate space at first.

Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 572 of file `stl_deque.h`.

References `std::max()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_range_initialize()`.

The documentation for this class was generated from the following file:

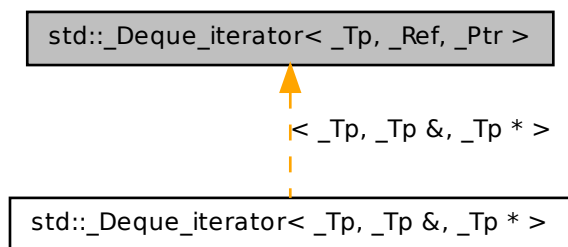
- [stl_deque.h](#)

5.284 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

A `deque::iterator`.

5.284 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference 1443

Inheritance diagram for std::_Deque_iterator< _Tp, _Ref, _Ptr >:



Public Types

- typedef `_Tp **` **_Map_pointer**
- typedef `_Deque_iterator` **_Self**
- typedef `_Deque_iterator< _Tp, const _Tp &, const _Tp * >` **const_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Deque_iterator< _Tp, _Tp &, _Tp * >` **iterator**
- typedef `std::random_access_iterator_tag` **iterator_category**
- typedef `_Ptr` **pointer**
- typedef `_Ref` **reference**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- `_Deque_iterator` (`_Tp *__x`, `_Map_pointer __y`)
- `_Deque_iterator` (`const iterator &__x`)
- `void _M_set_node` (`_Map_pointer __new_node`)
- reference **operator*** () const
- `_Self operator+` (`difference_type __n`) const
- `_Self operator++` (`int`)
- `_Self & operator++` ()
- `_Self & operator+=` (`difference_type __n`)
- `_Self operator-` (`difference_type __n`) const

- [_Self](#) & **operator--** ()
- [_Self](#) **operator--** (int)
- [_Self](#) & **operator-=** (difference_type __n)
- pointer **operator->** () const
- reference **operator[]** (difference_type __n) const

Static Public Member Functions

- static size_t **_S_buffer_size** ()

Public Attributes

- `_Tp * _M_cur`
- `_Tp * _M_first`
- `_Tp * _M_last`
- `_Map_pointer _M_node`

5.284.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr> struct std::_Deque_iterator< _Tp, _Ref, _Ptr >

A deque::iterator. Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

Definition at line 103 of file `stl_deque.h`.

5.284.2 Member Function Documentation

5.284.2.1 **template<typename _Tp, typename _Ref, typename _Ptr>**
void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (
_Map_pointer __new_node) [inline]

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

Definition at line 231 of file `stl_deque.h`.

The documentation for this struct was generated from the following file:

- [stl_deque.h](#)

5.285 `std::_Derives_from_binary_function< _Tp >` Struct Template Reference

Determines if the type `_Tp` derives from [binary_function](#).

Inherits `__sfinae_types`.

Static Public Attributes

- static const bool **value**

5.285.1 Detailed Description

```
template<typename _Tp> struct std::_Derives_from_binary_function< _Tp >
```

Determines if the type `_Tp` derives from [binary_function](#).

Definition at line 199 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.286 `std::_Derives_from_unary_function< _Tp >` Struct Template Reference

Determines if the type `_Tp` derives from [unary_function](#).

Inherits `__sfinae_types`.

Static Public Attributes

- static const bool **value**

5.286.1 Detailed Description

```
template<typename _Tp> struct std::_Derives_from_unary_function< _Tp >
```

Determines if the type `_Tp` derives from [unary_function](#).

Definition at line 183 of file `functional`.

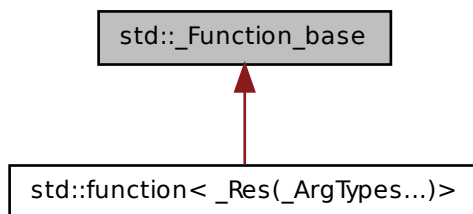
The documentation for this struct was generated from the following file:

- [functional](#)

5.287 std::_Function_base Class Reference

Base class of all polymorphic function object wrappers.

Inheritance diagram for std::_Function_base:



Public Types

- `typedef bool(* _Manager_type)(_Any_data &, const _Any_data &, _Manager_operation)`

Public Member Functions

- `bool _M_empty() const`

Public Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

Static Public Attributes

- `static const std::size_t _M_max_align`
- `static const std::size_t _M_max_size`

5.287.1 Detailed Description

Base class of all polymorphic function object wrappers.

Definition at line 1557 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

5.288 std::_Function_to_function_pointer< _Tp, _IsFunctionType > Struct Template Reference

Turns a function type into a function pointer type.

Public Types

- `typedef _Tp type`

5.288.1 Detailed Description

```
template<typename _Tp, bool _IsFunctionType = is_function<_Tp>::value>
struct std::_Function_to_function_pointer< _Tp, _IsFunctionType >
```

Turns a function type into a function pointer type.

Definition at line 215 of file functional.

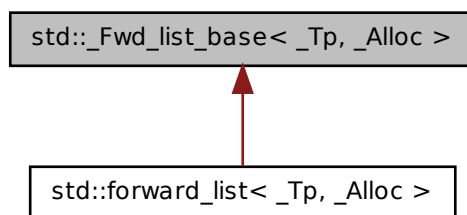
The documentation for this struct was generated from the following file:

- [functional](#)

5.289 std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference

Base class for forward_list.

Inheritance diagram for `std::_Fwd_list_base< _Tp, _Alloc >`:



Public Types

- typedef `_Fwd_list_node< _Tp >` **_Node**
- typedef `_Fwd_list_const_iterator< _Tp >` **const_iterator**
- typedef `_Fwd_list_iterator< _Tp >` **iterator**

Public Member Functions

- `_Fwd_list_base` (const `_Alloc` &__a)
- `_Fwd_list_base` (`_Fwd_list_base` &&__lst)
- `_Fwd_list_base` (const `_Fwd_list_base` &__lst, const `_Alloc` &__a)
- `_Fwd_list_base` (`_Fwd_list_base` &&__lst, const `_Alloc` &__a)
- const `_Node_alloc_type` & `_M_get_Node_allocator` () const
- `_Node_alloc_type` & `_M_get_Node_allocator` ()

Protected Types

- typedef `_Alloc::template rebind< _Fwd_list_node< _Tp > >::other` **_Node_alloc_type**
- typedef `_Alloc::template rebind< _Tp >::other` **_Tp_alloc_type**

Protected Member Functions

- template<typename... _Args>
`_Node` * `_M_create_node` (_Args &&...__args)

- `void _M_erase_after (_Fwd_list_node_base * __pos)`
- `void _M_erase_after (_Fwd_list_node_base * __pos, _Fwd_list_node_base * __last)`
- `_Node * _M_get_node ()`
- `template<typename... _Args> _Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... _args)`
- `void _M_put_node (_Node * __p)`

Protected Attributes

- `_Fwd_list_impl _M_impl`

5.289.1 Detailed Description

`template<typename _Tp, typename _Alloc> struct std::_Fwd_list_base< _Tp, _Alloc >`

Base class for `forward_list`.

Definition at line 274 of file `forward_list.h`.

The documentation for this struct was generated from the following files:

- [forward_list.h](#)
- [forward_list.tcc](#)

5.290 `std::_Fwd_list_const_iterator< _Tp >` Struct Template Reference

A `forward_list::const_iterator`.

Public Types

- `typedef const _Fwd_list_node< _Tp > _Node`
- `typedef _Fwd_list_const_iterator< _Tp > _Self`
- `typedef ptrdiff_t difference_type`
- `typedef _Fwd_list_iterator< _Tp > iterator`
- `typedef std::forward_iterator_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

Public Member Functions

- `_Fwd_list_const_iterator` (const [_Fwd_list_node_base](#) * __n)
- `_Fwd_list_const_iterator` (const [iterator](#) & __iter)
- `_Self _M_next` () const
- `bool operator!=` (const [_Self](#) & __x) const
- `reference operator*` () const
- `_Self operator++` (int)
- `_Self & operator++` ()
- `pointer operator->` () const
- `bool operator==` (const [_Self](#) & __x) const

Public Attributes

- `const _Fwd_list_node_base * _M_node`

5.290.1 Detailed Description

`template<typename _Tp> struct std::_Fwd_list_const_iterator< _Tp >`

A `forward_list::const_iterator`. All the functions are op overloads.

Definition at line 186 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

5.291 `std::_Fwd_list_iterator< _Tp >` Struct Template Reference

A `forward_list::iterator`.

Public Types

- `typedef _Fwd_list_node< _Tp > _Node`
- `typedef _Fwd_list_iterator< _Tp > _Self`
- `typedef ptrdiff_t difference_type`
- `typedef std::forward_iterator_tag iterator_category`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef _Tp value_type`

Public Member Functions

- `_Fwd_list_iterator` (`_Fwd_list_node_base *__n`)
- `_Self _M_next` () const
- bool `operator!=` (const `_Self` &__x) const
- reference `operator*` () const
- `_Self` & `operator++` ()
- `_Self` `operator++` (int)
- pointer `operator->` () const
- bool `operator==` (const `_Self` &__x) const

Public Attributes

- `_Fwd_list_node_base * _M_node`

5.291.1 Detailed Description

`template<typename _Tp> struct std::_Fwd_list_iterator< _Tp >`

A `forward_list::iterator`. All the functions are op overloads.

Definition at line 118 of file `forward_list.h`.

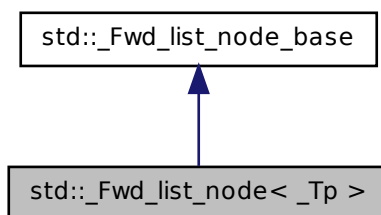
The documentation for this struct was generated from the following file:

- `forward_list.h`

5.292 `std::_Fwd_list_node< _Tp >` Struct Template Reference

A helper node class for `forward_list`. This is just a linked list with a data value in each node. There is a sorting utility method.

Inheritance diagram for `std::_Fwd_list_node< _Tp >`:



Public Member Functions

- `template<typename... _Args>`
`_Fwd_list_node` (`_Args &&...__args`)
- `void _M_reverse_after ()`
- `_Fwd_list_node_base * _M_transfer_after` (`_Fwd_list_node_base * __begin, _Fwd_list_node_base * __end`)
- `_Fwd_list_node_base * _M_transfer_after` (`_Fwd_list_node_base * __begin`)

Static Public Member Functions

- `static void swap` (`_Fwd_list_node_base & __x, _Fwd_list_node_base & __y`)

Public Attributes

- `_Fwd_list_node_base * _M_next`
- `_Tp _M_value`

5.292.1 Detailed Description

`template<typename _Tp> struct std::_Fwd_list_node< _Tp >`

A helper node class for `forward_list`. This is just a linked list with a data value in each node. There is a sorting utility method.

Definition at line 101 of file forward_list.h.

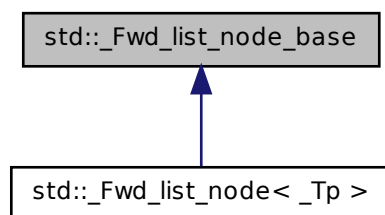
The documentation for this struct was generated from the following file:

- [forward_list.h](#)

5.293 std::_Fwd_list_node_base Struct Reference

A helper basic node class for forward_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Inheritance diagram for std::_Fwd_list_node_base:



Public Member Functions

- void **_M_reverse_after** ()
- [_Fwd_list_node_base](#) * **_M_transfer_after** ([_Fwd_list_node_base](#) * __begin, [_Fwd_list_node_base](#) * __end)
- [_Fwd_list_node_base](#) * **_M_transfer_after** ([_Fwd_list_node_base](#) * __begin)

Static Public Member Functions

- static void **swap** ([_Fwd_list_node_base](#) & __x, [_Fwd_list_node_base](#) & __y)

Public Attributes

- [_Fwd_list_node_base](#) * **_M_next**

5.293.1 Detailed Description

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Definition at line 44 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

5.294 `std::_Index_tuple< _Indexes >` Struct Template Reference

Public Types

- typedef [_Index_tuple](#)< _Indexes..., sizeof...(_Indexes)> **__next**

5.294.1 Detailed Description

`template<int... _Indexes> struct std::_Index_tuple< _Indexes >`

Stores a tuple of indices. Used by [bind\(\)](#) to extract the elements in a tuple.

Definition at line 757 of file `tuple`.

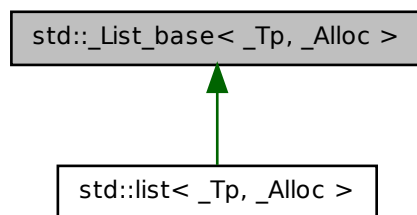
The documentation for this struct was generated from the following file:

- [tuple](#)

5.295 `std::_List_base< _Tp, _Alloc >` Class Template Reference

See [bits/stl_deque.h](#)'s `_Deque_base` for an explanation.

Inheritance diagram for std::_List_base< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**

Public Member Functions

- **_List_base** (const allocator_type &__a)
- **_List_base** ([_List_base](#) &&__x)
- void **_M_clear** ()
- const _Node_alloc_type & **_M_get_Node_allocator** () const
- _Node_alloc_type & **_M_get_Node_allocator** ()
- _Tp_alloc_type **_M_get_Tp_allocator** () const
- void **_M_init** ()
- allocator_type **get_allocator** () const

Protected Types

- typedef _Alloc::template rebind< [_List_node](#)< _Tp > >::other **_Node_alloc_type**
- typedef _Alloc::template rebind< _Tp >::other **_Tp_alloc_type**

Protected Member Functions

- [_List_node](#)< _Tp > * **_M_get_node** ()
- void **_M_put_node** ([_List_node](#)< _Tp > *__p)

Protected Attributes

- `_List_impl _M_impl`

5.295.1 Detailed Description

`template<typename _Tp, typename _Alloc> class std::_List_base< _Tp, _Alloc >`

See [bits/stl_deque.h](#)'s `_Deque_base` for an explanation.

Definition at line 277 of file `stl_list.h`.

The documentation for this class was generated from the following files:

- [stl_list.h](#)
- [list.tcc](#)

5.296 `std::_List_const_iterator< _Tp >` Struct Template Reference

A `list::const_iterator`.

Public Types

- `typedef const _List_node< _Tp > _Node`
- `typedef _List_const_iterator< _Tp > _Self`
- `typedef ptrdiff_t difference_type`
- `typedef _List_iterator< _Tp > iterator`
- `typedef std::bidirectional_iterator_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

Public Member Functions

- `_List_const_iterator (const _List_node_base *__x)`
- `_List_const_iterator (const iterator &__x)`
- `bool operator!= (const _Self &__x) const`
- `reference operator* () const`
- `_Self operator++ (int)`

- `_Self` & `operator++` ()
- `_Self` & `operator--` ()
- `_Self` `operator--` (int)
- pointer `operator->` () const
- bool `operator==` (const `_Self` &__x) const

Public Attributes

- const `_List_node_base` * `_M_node`

5.296.1 Detailed Description

`template<typename _Tp> struct std::_List_const_iterator<_Tp>`

A `list::const_iterator`. All the functions are op overloads.

Definition at line 188 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl_list.h](#)

5.297 `std::_List_iterator<_Tp>` Struct Template Reference

A `list::iterator`.

Public Types

- typedef `_List_node<_Tp>` `_Node`
- typedef `_List_iterator<_Tp>` `_Self`
- typedef `ptrdiff_t` `difference_type`
- typedef `std::bidirectional_iterator_tag` `iterator_category`
- typedef `_Tp` * `pointer`
- typedef `_Tp` & `reference`
- typedef `_Tp` `value_type`

Public Member Functions

- [_List_iterator](#) ([_List_node_base](#) *__x)
- bool **operator!=** (const [_Self](#) &__x) const
- reference **operator*** () const
- [_Self](#) & **operator++** ()
- [_Self](#) **operator++** (int)
- [_Self](#) & **operator--** ()
- [_Self](#) **operator--** (int)
- pointer **operator->** () const
- bool **operator==** (const [_Self](#) &__x) const

Public Attributes

- [_List_node_base](#) * **_M_node**

5.297.1 Detailed Description

template<typename _Tp> struct std::_List_iterator< _Tp >

A list::iterator. All the functions are op overloads.

Definition at line 113 of file stl_list.h.

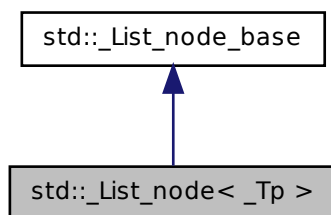
The documentation for this struct was generated from the following file:

- [stl_list.h](#)

5.298 **std::_List_node< _Tp > Struct Template Reference**

An actual node in the list.

Inheritance diagram for std::_List_node< _Tp >:



Public Member Functions

- `template<typename... _Args>`
`_List_node` (`_Args` &&...__args)
- `void _M_hook` (`_List_node_base` *const __position) throw ()
- `void _M_reverse` () throw ()
- `void _M_transfer` (`_List_node_base` *const __first, `_List_node_base` *const __last) throw ()
- `void _M_unhook` () throw ()

Static Public Member Functions

- `static void swap` (`_List_node_base` &__x, `_List_node_base` &__y) throw ()

Public Attributes

- `_Tp _M_data`
- `_List_node_base` * `_M_next`
- `_List_node_base` * `_M_prev`

5.298.1 Detailed Description

`template<typename _Tp> struct std::_List_node< _Tp >`

An actual node in the list.

Definition at line 95 of file stl_list.h.

5.298.2 Member Data Documentation

5.298.2.1 `template<typename _Tp> _Tp std::_List_node< _Tp >::_M_data`

< User's data.

Definition at line 98 of file stl_list.h.

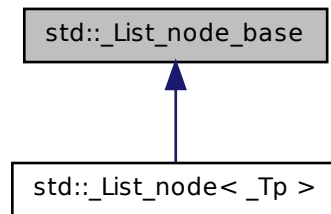
The documentation for this struct was generated from the following file:

- [stl_list.h](#)

5.299 `std::_List_node_base` Struct Reference

Common part of a node in the list.

Inheritance diagram for `std::_List_node_base`:



Public Member Functions

- `void _M_hook (_List_node_base *const __position) throw ()`
- `void _M_reverse () throw ()`
- `void _M_transfer (_List_node_base *const __first, _List_node_base *const __last) throw ()`
- `void _M_unhook () throw ()`

Static Public Member Functions

- static void `swap` (`_List_node_base` &__x, `_List_node_base` &__y) throw ()

Public Attributes

- `_List_node_base` * `_M_next`
- `_List_node_base` * `_M_prev`

5.299.1 Detailed Description

Common part of a node in the list.

Definition at line 71 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl_list.h](#)

5.300 `std::_Maybe_get_result_type< _Has_result_type, _Functor > Struct` Template Reference

If we have found a `result_type`, extract it.

Inheritance diagram for `std::_Maybe_get_result_type< _Has_result_type, _Functor >`:



5.300.1 Detailed Description

`template<bool _Has_result_type, typename _Functor> struct std::_Maybe_get_result_type< _Has_result_type, _Functor >`

If we have found a `result_type`, extract it.

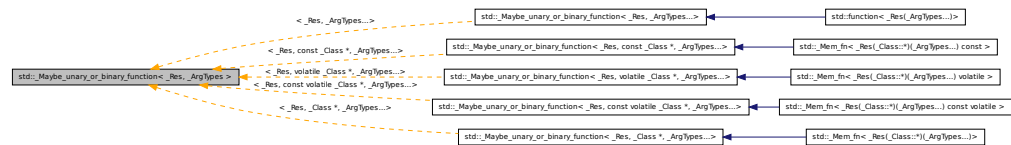
Definition at line 67 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.301 `std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >` Struct Template Reference

Inheritance diagram for `std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >`:



5.301.1 Detailed Description

`template<typename _Res, typename... _ArgTypes> struct std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >`

Derives from `unary_function` or `binary_function`, or perhaps nothing, depending on the number of arguments provided. The primary template is the basis case, which derives nothing.

Definition at line 500 of file `functional`.

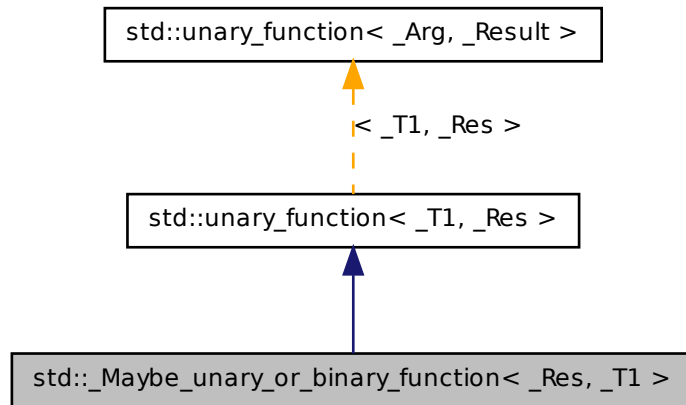
The documentation for this struct was generated from the following file:

- [functional](#)

5.302 `std::_Maybe_unary_or_binary_function< _Res, _T1 >` Struct Template Reference

Derives from `unary_function`, as appropriate.

Inheritance diagram for `std::_Maybe_unary_or_binary_function<_Res, _T1>`:



Public Types

- typedef `_T1` [argument_type](#)
- typedef `_Res` [result_type](#)

5.302.1 Detailed Description

`template<typename _Res, typename _T1> struct std::_Maybe_unary_or_binary_function<_Res, _T1>`

Derives from [unary_function](#), as appropriate.

Definition at line 504 of file `functional`.

5.302.2 Member Typedef Documentation

5.302.2.1 `typedef _T1 std::unary_function<_T1, _Res>::argument_type`
[[inherited](#)]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.302.2.2 `typedef _Res std::unary_function< _T1 , _Res >::result_type`
[inherited]

`result_type` is the return type

Definition at line 105 of file stl_function.h.

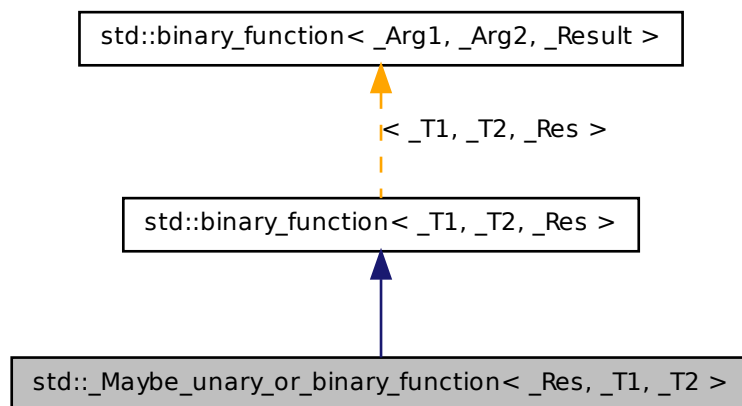
The documentation for this struct was generated from the following file:

- [functional](#)

5.303 `std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >` Struct Template Reference

Derives from [binary_function](#), as appropriate.

Inheritance diagram for `std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >`:



Public Types

- `typedef _T1` [first_argument_type](#)

- typedef _Res [result_type](#)
- typedef _T2 [second_argument_type](#)

5.303.1 Detailed Description

template<typename _Res, typename _T1, typename _T2> struct std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >

Derives from [binary_function](#), as appropriate.

Definition at line 509 of file functional.

5.303.2 Member Typedef Documentation

5.303.2.1 typedef _T1 std::binary_function< _T1 , _T2 , _Res >::first_argument_type [inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.303.2.2 typedef _Res std::binary_function< _T1 , _T2 , _Res >::result_type [inherited]

type of the return type

Definition at line 118 of file stl_function.h.

5.303.2.3 typedef _T2 std::binary_function< _T1 , _T2 , _Res >::second_argument_type [inherited]

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [functional](#)

5.304 `std::_Maybe_wrap_member_pointer< _Tp >` Struct Template Reference

Public Types

- `typedef _Tp type`

Static Public Member Functions

- `static const _Tp & __do_wrap (const _Tp &__x)`
- `static _Tp && __do_wrap (_Tp &&__x)`

5.304.1 Detailed Description

`template<typename _Tp> struct std::_Maybe_wrap_member_pointer< _Tp >`

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. The primary template handles the non--member-pointer case.

Definition at line 1062 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.305 `std::_Maybe_wrap_member_pointer< _Tp _- Class::* >` Struct Template Reference

Public Types

- `typedef _Mem_fn< _Tp _Class::* > type`

Static Public Member Functions

- `static type __do_wrap (_Tp _Class::* __pm)`

5.305.1 Detailed Description

```
template<typename _Tp, typename _Class> struct std::_Maybe_wrap_
member_pointer< _Tp _Class::* >
```

Maps member pointers into instances of `_Mem_fn` but leaves all other function objects untouched. Used by `tr1::bind()`. This partial specialization handles the member pointer case.

Definition at line 1081 of file `functional`.

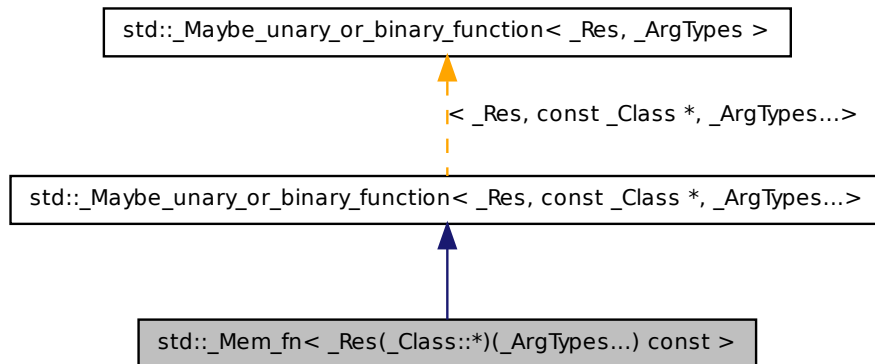
The documentation for this struct was generated from the following file:

- [functional](#)

5.306 `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const > Class Template Reference`

Implementation of `mem_fn` for const member function pointers.

Inheritance diagram for `std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >`:



Public Types

- `typedef _Res result_type`

Public Member Functions

- **_Mem_fn** (_Functor __pmf)
- template<typename _Tp >
_Res **operator**() (_Tp &__object, _ArgTypes...__args) const
- _Res **operator**() (const _Class *__object, _ArgTypes...__args) const
- _Res **operator**() (const _Class &__object, _ArgTypes...__args) const

5.306.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes> class
std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >

Implementation of mem_fn for const member function pointers.

Definition at line 560 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

5.307 std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile > Class Template Reference

Implementation of mem_fn for const volatile member function pointers.

Inheritance diagram for std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile >:



Public Types

- typedef _Res **result_type**

Public Member Functions

- **_Mem_fn** (_Functor __pmf)
- template<typename _Tp >
_Res **operator**() (_Tp &__object, _ArgTypes...__args) const

5.308 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >` Class Template Reference 1469

- `_Res operator()` (const volatile `_Class *`__object, `_ArgTypes...`__args) const
- `_Res operator()` (const volatile `_Class &`__object, `_ArgTypes...`__args) const

5.307.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> class std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) const volatile >`

Implementation of `mem_fn` for const volatile member function pointers.

Definition at line 653 of file `functional`.

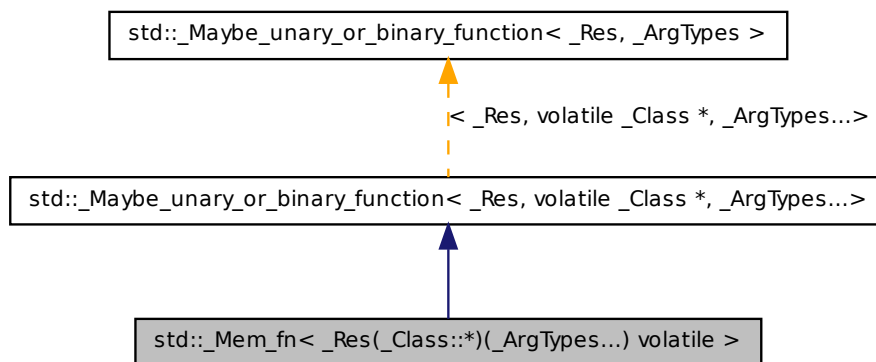
The documentation for this class was generated from the following file:

- [functional](#)

5.308 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >` Class Template Reference

Implementation of `mem_fn` for volatile member function pointers.

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >`:



Public Types

- typedef `_Res` **result_type**

Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp >`
`_Res operator() (_Tp &__object, _ArgTypes...__args) const`
- `_Res operator() (volatile _Class *__object, _ArgTypes...__args) const`
- `_Res operator() (volatile _Class &__object, _ArgTypes...__args) const`

5.308.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> class`
`std::_Mem_fn<_Res(_Class::*)(_ArgTypes...) volatile >`

Implementation of `mem_fn` for volatile member function pointers.

Definition at line 606 of file `functional`.

The documentation for this class was generated from the following file:

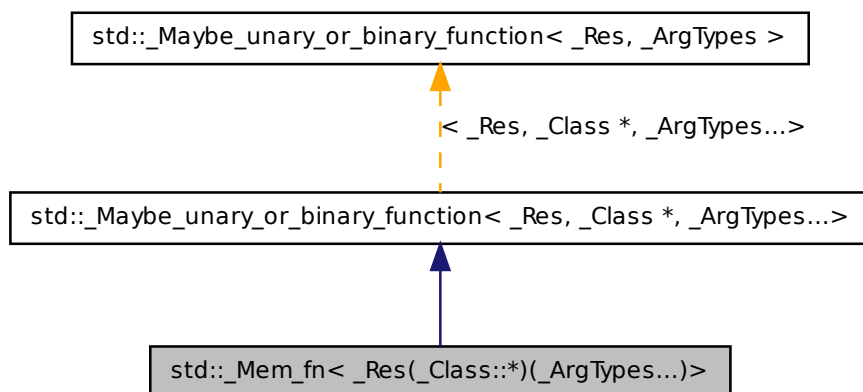
- [functional](#)

5.309 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>` Class Template Reference

Implementation of `mem_fn` for member function pointers.

5.309 `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>` Class Template Reference

Inheritance diagram for `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>`:



Public Types

- `typedef _Res result_type`

Public Member Functions

- `_Mem_fn` (`_Functor __pmf`)
- `template<typename _Tp > _Res operator() (_Tp &__object, _ArgTypes... __args) const`
- `_Res operator() (_Class *__object, _ArgTypes... __args) const`
- `_Res operator() (_Class &__object, _ArgTypes... __args) const`

5.309.1 Detailed Description

`template<typename _Res, typename _Class, typename... _ArgTypes> class std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>`

Implementation of `mem_fn` for member function pointers.

Definition at line 514 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.310 `std::_Mu< _Arg, false, false >` Class Template Reference

Public Member Functions

- `template<typename _CVarArg, typename _Tuple > _CVarArg && operator() (_CVarArg &&__arg, _Tuple &) const volatile`

5.310.1 Detailed Description

`template<typename _Arg> class std::_Mu< _Arg, false, false >`

If the argument is just a value, returns a reference to that value. The cv-qualifiers on the reference are the same as the cv-qualifiers on the `_Mu` object. [TR1 3.6.3/5 bullet 4]

Definition at line 1038 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.311 `std::_Mu< _Arg, false, true >` Class Template Reference

Public Member Functions

- `template<typename _Tuple > result< _Mu(_Arg, _Tuple)>::type operator() (const volatile _Arg &, _Tuple &__tuple) const volatile`

5.311.1 Detailed Description

`template<typename _Arg> class std::_Mu< _Arg, false, true >`

If the argument is a placeholder for the Nth argument, returns a reference to the Nth argument to the bind function object. [TR1 3.6.3/5 bullet 3]

Definition at line 1004 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.312 `std::_Mu< _Arg, true, false >` Class Template Reference

Public Member Functions

- `template<typename _CVArg, typename... _Args>
auto operator() (_CVArg &__arg, tuple< _Args...> &__tuple) const volatile->
decltype(__arg(declval< _Args >()....))`

5.312.1 Detailed Description

`template<typename _Arg> class std::_Mu< _Arg, true, false >`

If the argument is a bind expression, we invoke the underlying function object with the same cv-qualifiers as we are given and pass along all of our arguments (unwrapped).
[TR1 3.6.3/5 bullet 2]

Definition at line 970 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.313 `std::_Mu< reference_wrapper< _Tp >, false, false >` Class Template Reference

Public Types

- `typedef _Tp & result_type`

Public Member Functions

- `template<typename _CVRef, typename _Tuple >
result_type operator() (_CVRef &__arg, _Tuple &) const volatile`

5.313.1 Detailed Description

`template<typename _Tp> class std::_Mu< reference_wrapper< _Tp >, false, false >`

If the argument is `reference_wrapper<_Tp>`, returns the underlying reference. [TR1 3.6.3/5 bullet 1]

Definition at line 949 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

5.314 `std::_Placeholder< _Num >` Struct Template Reference

The type of placeholder objects defined by `libstdc++`.

5.314.1 Detailed Description

`template<int _Num> struct std::_Placeholder< _Num >`

The type of placeholder objects defined by `libstdc++`.

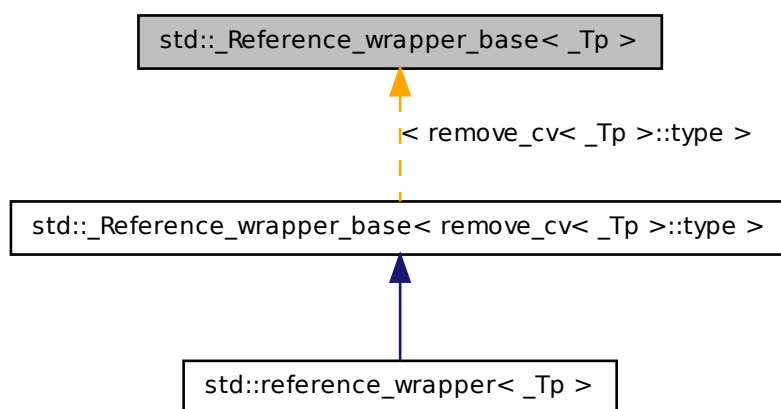
Definition at line 836 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.315 `std::_Reference_wrapper_base< _Tp >` Struct Template Reference

Inheritance diagram for `std::_Reference_wrapper_base< _Tp >`:



5.315.1 Detailed Description

```
template<typename _Tp> struct std::_Reference_wrapper_base< _Tp >
```

Derives from [unary_function](#) or [binary_function](#) when it can. Specializations handle all of the easy cases. The primary template determines what to do with a class type, which may derive from both [unary_function](#) and [binary_function](#).

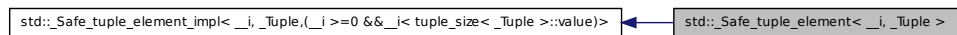
Definition at line 305 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.316 `std::_Safe_tuple_element< __i, _Tuple > Struct Template Reference`

Inheritance diagram for `std::_Safe_tuple_element< __i, _Tuple >`:



5.316.1 Detailed Description

`template<int __i, typename _Tuple> struct std::_Safe_tuple_element< __i, _Tuple >`

Like `tuple_element`, but returns `_No_tuple_element` when `tuple_element` would return an error.

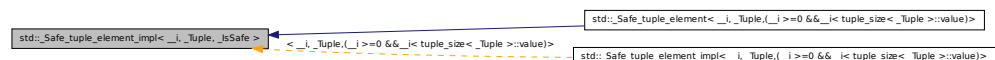
Definition at line 923 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.317 `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe > Struct Template Reference`

Inheritance diagram for `std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >`:



5.317.1 Detailed Description

```
template<int __i, typename _Tuple, bool _IsSafe> struct std::_Safe_tuple_
element_impl< __i, _Tuple, _IsSafe >
```

Implementation helper for [_Safe_tuple_element](#). This primary template handles the case where it is safe to use `tuple_element`.

Definition at line 904 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.318 std::_Safe_tuple_element_impl< __i, _Tuple, false > Struct Template Reference

Public Types

- `typedef _No_tuple_element type`

5.318.1 Detailed Description

```
template<int __i, typename _Tuple> struct std::_Safe_tuple_element_impl< __i,
_Tuple, false >
```

Implementation helper for [_Safe_tuple_element](#). This partial specialization handles the case where it is not safe to use `tuple_element`. We just return `_No_tuple_element`.

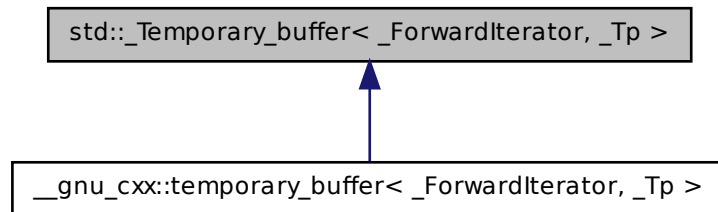
Definition at line 913 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.319 `std::_Temporary_buffer<_ForwardIterator, _Tp>` Class Template Reference

Inheritance diagram for `std::_Temporary_buffer<_ForwardIterator, _Tp>`:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [_Temporary_buffer](#) (`_ForwardIterator __first, _ForwardIterator __last`)
- iterator [begin](#) ()
- iterator [end](#) ()
- size_type [requested_size](#) () const
- size_type [size](#) () const

Protected Attributes

- pointer **_M_buffer**
- size_type **_M_len**
- size_type **_M_original_len**

5.319.1 Detailed Description

template<typename _ForwardIterator, typename _Tp> class std::_Temporary_buffer< _ForwardIterator, _Tp >

This class is used in two places: [stl_algo.h](#) and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

Definition at line 121 of file `stl_tembuf.h`.

5.319.2 Constructor & Destructor Documentation

5.319.2.1 **template<typename _ForwardIterator, typename _Tp>
> std::_Temporary_buffer< _ForwardIterator, _Tp
>::_Temporary_buffer (_ForwardIterator __first,
_ForwardIterator __last)**

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 244 of file `stl_tembuf.h`.

References `std::pair< _T1, _T2 >::first`, `std::get_temporary_buffer()`, `std::return_temporary_buffer()`, and `std::pair< _T1, _T2 >::second`.

5.319.3 Member Function Documentation

5.319.3.1 **template<typename _ForwardIterator, typename _Tp> iterator
std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()
[inline]**

As per Table mumble.

Definition at line 150 of file `stl_tembuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

5.319.3.2 **template<typename _ForwardIterator, typename _Tp> iterator
std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()
[inline]**

As per Table mumble.

Definition at line 155 of file `stl_tembuf.h`.

5.319.3.3 `template<typename _ForwardIterator, typename _Tp> size_type
std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size () const [inline]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 145 of file `stl_tempbuf.h`.

Referenced by `std::stable_partition()`.

5.319.3.4 `template<typename _ForwardIterator, typename _Tp> size_type
std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline]`

As per Table mumble.

Definition at line 140 of file `stl_tempbuf.h`.

Referenced by `std::inplace_merge()`, `std::stable_partition()`, and `std::stable_sort()`.

The documentation for this class was generated from the following file:

- [stl_tempbuf.h](#)

5.320 `std::_Tuple_impl< _Idx >` Struct Template Reference

Protected Member Functions

- `void _M_swap_impl (_Tuple_impl &)`

5.320.1 Detailed Description

`template<std::size_t _Idx> struct std::_Tuple_impl< _Idx >`

Zero-element tuple implementation. This is the basis case for the inheritance recursion.

Definition at line 125 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.321 std::Tuple_impl< _Idx, _Head, _Tail...> Struct Template Reference

Inherits Tuple_impl< _Idx+1, _Tail...>, and Head_base< _Idx, _Head, std::is_empty< _Head >::value >.

Public Types

- typedef Head_base< _Idx, _Head, std::is_empty< _Head >::value > **_Base**
- typedef Tuple_impl< _Idx+1, _Tail...> **_Inherited**

Public Member Functions

- **_Tuple_impl** (const _Tuple_impl &)
- **_Tuple_impl** (_Tuple_impl &&__in)
- template<typename... _UElements>
_Tuple_impl (const _Tuple_impl< _Idx, _UElements...> &__in)
- template<typename... _UElements>
_Tuple_impl (_Tuple_impl< _Idx, _UElements...> &&__in)
- **_Tuple_impl** (const _Head &__head, const _Tail &...__tail)
- template<typename _UHead, typename... _UTail>
_Tuple_impl (_UHead &&__head, _UTail &&...__tail)
- _Head & **_M_head** ()
- const _Head & **_M_head** () const
- const _Inherited & **_M_tail** () const
- _Inherited & **_M_tail** ()
- template<typename... _UElements>
_Tuple_impl & **operator=** (_Tuple_impl< _Idx, _UElements...> &&__in)
- _Tuple_impl & **operator=** (const _Tuple_impl &__in)
- _Tuple_impl & **operator=** (_Tuple_impl &&__in)
- template<typename... _UElements>
_Tuple_impl & **operator=** (const _Tuple_impl< _Idx, _UElements...> &__in)

Protected Member Functions

- void **_M_swap_impl** (_Tuple_impl &__in)

5.321.1 Detailed Description

```
template<std::size_t _Idx, typename _Head, typename... _Tail> struct std::_  
Tuple_impl<_Idx, _Head, _Tail...>
```

Recursive tuple implementation. Here we store the `Head` element and derive from a `Tuple_impl` containing the remaining elements (which contains the `Tail`).

Definition at line 137 of file `tuple`.

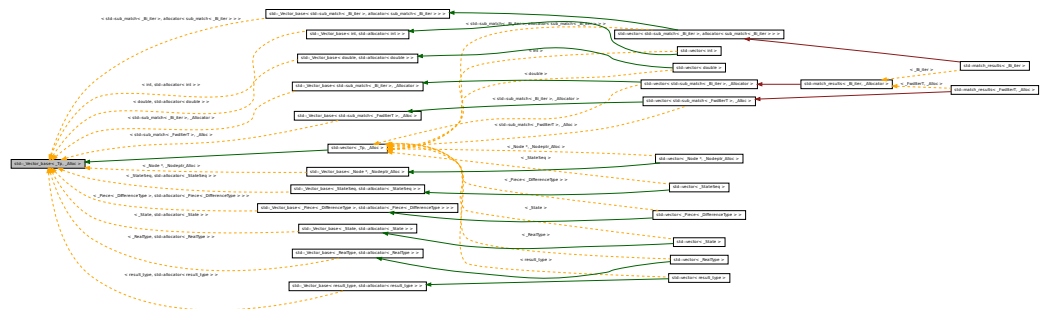
The documentation for this struct was generated from the following file:

- [tuple](#)

5.322 std::_Vector_base< _Tp, _Alloc > Struct Template Reference

See [bits/stl_deque.h](#)'s [_Deque_base](#) for an explanation.

Inheritance diagram for `std::_Vector_base< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc::template rebind< _Tp >::other _Tp_alloc_type`
- typedef `_Alloc` **allocator_type**

Public Member Functions

- `_Vector_base` (const `allocator_type` &__a)
- `_Vector_base` (`_Vector_base` &&__x)
- `_Vector_base` (size_t __n)

- [functional](#)

5.324 `std::_Weak_result_type_impl< _Functor >` Struct Template Reference

Inheritance diagram for `std::_Weak_result_type_impl< _Functor >`:



5.324.1 Detailed Description

```
template<typename _Functor> struct std::_Weak_result_type_impl< _Functor
>
```

Base class for any function object that has a weak result type, as defined in 3.3/3 of TR1.

Definition at line 79 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.325 `std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function reference.

Public Types

- `typedef _Res result_type`

5.325.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_ -
type_impl< _Res(&)(_ArgTypes...)>
```

Retrieve the result type for a function reference.

5.326 `std::_Weak_result_type_impl<_Res(*)(_ArgTypes...)>` Struct Template Reference 1485

Definition at line 118 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.326 `std::_Weak_result_type_impl<_Res(*)(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function pointer.

Public Types

- typedef `_Res result_type`

5.326.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_type_impl<_Res(*)(_ArgTypes...)>
```

Retrieve the result type for a function pointer.

Definition at line 127 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.327 `std::_Weak_result_type_impl<_Res(_ArgTypes...)>` Struct Template Reference

Retrieve the result type for a function type.

Public Types

- typedef `_Res result_type`

5.327.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> struct std::_Weak_result_
type_impl< _Res(_ArgTypes...)>
```

Retrieve the result type for a function type.

Definition at line 85 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.328 `std::_Weak_result_type_impl< _Res(_- Class::*)(_ArgTypes...) const >` Struct Template Reference

Retrieve result type for a const member function pointer.

Public Types

- typedef `_Res result_type`

5.328.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >
```

Retrieve result type for a const member function pointer.

Definition at line 145 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.329 `std::_Weak_result_type_impl< _Res(_- Class::*)(_ArgTypes...) const volatile >` Struct Template Reference

Retrieve result type for a const volatile member function pointer.

Public Types

- typedef _Res **result_type**

5.329.1 Detailed Description

**template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >**

Retrieve result type for a const volatile member function pointer.

Definition at line 163 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.330 std::_Weak_result_type_impl< _Res(_- Class::*)(_ArgTypes...) volatile > Struct Tem- plate Reference

Retrieve result type for a volatile member function pointer.

Public Types

- typedef _Res **result_type**

5.330.1 Detailed Description

**template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >**

Retrieve result type for a volatile member function pointer.

Definition at line 154 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

5.331 `std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)>` Struct Template Reference

Retrieve result type for a member function pointer.

Public Types

- `typedef _Res result_type`

5.331.1 Detailed Description

```
template<typename _Res, typename _Class, typename... _ArgTypes> struct
std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...)>
```

Retrieve result type for a member function pointer.

Definition at line 136 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.332 `std::add_lvalue_reference< _Tp >` Struct Template Reference

[add_lvalue_reference](#)

Inherits `std::__add_lvalue_reference_helper< _Tp >`.

Public Types

- `typedef _Tp type`

5.332.1 Detailed Description

```
template<typename _Tp> struct std::add_lvalue_reference< _Tp >
```

[add_lvalue_reference](#)

Definition at line 125 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.333 `std::add_rvalue_reference< _Tp >` Struct Template Reference

[add_rvalue_reference](#)

Inherits `std::__add_rvalue_reference_helper< _Tp >`.

Public Types

- `typedef _Tp type`

5.333.1 Detailed Description

```
template<typename _Tp> struct std::add_rvalue_reference< _Tp >
```

[add_rvalue_reference](#)

Definition at line 140 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.334 `std::adopt_lock_t` Struct Reference

Assume the calling thread has already obtained mutex ownership /// and manage it.

5.334.1 Detailed Description

Assume the calling thread has already obtained mutex ownership /// and manage it.

Definition at line 382 of file `mutex`.

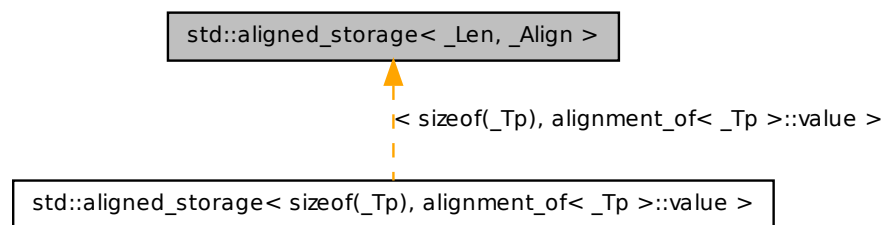
The documentation for this struct was generated from the following file:

- [mutex](#)

5.335 `std::aligned_storage< _Len, _Align >` Struct Template Reference

Alignment type.

Inheritance diagram for `std::aligned_storage< _Len, _Align >`:



5.335.1 Detailed Description

```
template<std::size_t _Len, std::size_t _Align = __alignof__(typename __-
aligned_storage_msa<_Len>::__type)> struct std::aligned_storage< _Len, _-
Align >
```

Alignment type. The value of `_Align` is a default-alignment which shall be the most stringent alignment requirement for any C++ object type whose size is no greater than `_Len` (3.9). The member typedef `type` shall be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.

Definition at line 370 of file `type_traits`.

The documentation for this struct was generated from the following file:

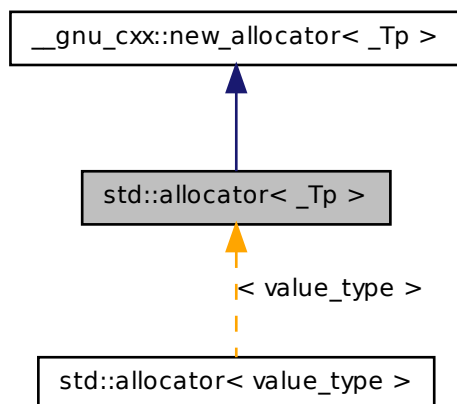
- [type_traits](#)

5.336 `std::allocator< _Tp >` Class Template Reference

The *standard* allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

Inheritance diagram for std::allocator< _Tp >:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **allocator** (const [allocator](#) &__a) throw ()
- template<typename _Tp1 >
 allocator (const [allocator](#)< _Tp1 > &) throw ()
- pointer **address** (reference __x) const
- const_pointer **address** (const_reference __x) const
- pointer **allocate** (size_type __n, const void * = 0)
- void **construct** (pointer __p, const _Tp & __val)

- `template<typename... _Args>`
`void construct (pointer __p, _Args &&...__args)`
- `void deallocate (pointer __p, size_type)`
- `void destroy (pointer __p)`
- `size_type max_size () const throw ()`

5.336.1 Detailed Description

`template<typename _Tp> class std::allocator< _Tp >`

The *standard* allocator, as per [20.4].

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.

Definition at line 90 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

5.337 std::allocator< void > Class Template Reference

[allocator<void>](#) specialization.

Public Types

- `typedef const void * const_pointer`
- `typedef ptrdiff_t difference_type`
- `typedef void * pointer`
- `typedef size_t size_type`
- `typedef void value_type`

5.337.1 Detailed Description

`template<> class std::allocator< void >`

[allocator<void>](#) specialization.

Definition at line 68 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

5.338 std::allocator_arg_t Struct Reference

[allocator.tag]

5.338.1 Detailed Description

[allocator.tag]

Definition at line 209 of file allocator.h.

The documentation for this struct was generated from the following file:

- [allocator.h](#)

5.339 std::atomic< _Tp > Struct Template Reference

atomic /// 29.4.3, Generic atomic type, primary class template.

Public Member Functions

- **atomic** (_Tp __i)
- **atomic** (const [atomic](#) &)
- bool **compare_exchange_strong** (_Tp &, _Tp, [memory_order](#), [memory_order](#)) volatile
- bool **compare_exchange_strong** (_Tp &, _Tp, [memory_order](#)=memory_order_order_seq_cst) volatile
- bool **compare_exchange_weak** (_Tp &, _Tp, [memory_order](#)=memory_order_order_seq_cst) volatile
- bool **compare_exchange_weak** (_Tp &, _Tp, [memory_order](#), [memory_order](#)) volatile
- _Tp **exchange** (_Tp __i, [memory_order](#)=memory_order_order_seq_cst) volatile
- bool **is_lock_free** () const volatile
- _Tp **load** ([memory_order](#)=memory_order_order_seq_cst) const volatile
- **operator _Tp** () const
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- _Tp **operator=** (_Tp __i)
- void **store** (_Tp, [memory_order](#)=memory_order_order_seq_cst) volatile

5.339.1 Detailed Description

template<typename _Tp> struct std::atomic< _Tp >

atomic /// 29.4.3, Generic atomic type, primary class template.

Definition at line 86 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.340 std::atomic< _Tp * > Struct Template Reference

Partial specialization for pointer types.

Inherits atomic_address.

Public Member Functions

- **atomic** (_Tp *__v)
- **atomic** (const [atomic](#) &)
- bool **compare_exchange_strong** (_Tp * &, _Tp *, [memory_order](#), [memory_order](#))
- bool **compare_exchange_strong** (_Tp * &, _Tp *, [memory_order](#)=memory_order_seq_cst)
- bool **compare_exchange_weak** (_Tp * &, _Tp *, [memory_order](#), [memory_order](#))
- bool **compare_exchange_weak** (_Tp * &, _Tp *, [memory_order](#)=memory_order_seq_cst)
- _Tp * **exchange** (_Tp *__v, [memory_order](#) __m=memory_order_seq_cst)
- _Tp * **fetch_add** (ptrdiff_t, [memory_order](#)=memory_order_seq_cst)
- _Tp * **fetch_sub** (ptrdiff_t, [memory_order](#)=memory_order_seq_cst)
- _Tp * **load** ([memory_order](#) __m=memory_order_seq_cst) const
- **operator _Tp** * () const
- _Tp * **operator++** (int)
- _Tp * **operator++** ()
- _Tp * **operator+=** (ptrdiff_t __d)
- _Tp * **operator--** ()
- _Tp * **operator--** (int)
- _Tp * **operator-=** (ptrdiff_t __d)
- _Tp * **operator=** (_Tp *__v)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile
- void **store** (_Tp *__v, [memory_order](#) __m=memory_order_seq_cst)

5.340.1 Detailed Description

`template<typename _Tp> struct std::atomic< _Tp * >`

Partial specialization for pointer types.

Definition at line 134 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.341 `std::atomic< bool >` Struct Template Reference

Explicit specialization for `bool`.

Inherits `__atomic_bool_base`.

Public Types

- `typedef atomic_bool __base_type`
- `typedef bool __integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic & operator=` (`const atomic &`) volatile

Public Attributes

- `bool _M_i`

5.341.1 Detailed Description

`template<> struct std::atomic< bool >`

Explicit specialization for `bool`.

Definition at line 225 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.342 `std::atomic< char >` Struct Template Reference

Explicit specialization for char.

Inherits `__atomic_char_base`.

Public Types

- typedef `atomic_char` `__base_type`
- typedef char `__integral_type`

Public Member Functions

- `atomic` (`__integral_type` `__i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- char `_M_i`

5.342.1 Detailed Description

`template<> struct std::atomic< char >`

Explicit specialization for char.

Definition at line 243 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.343 `std::atomic< char16_t >` Struct Template Reference

Explicit specialization for `char16_t`.

Inherits `__atomic_short_base`.

Public Types

- typedef `atomic_char16_t __base_type`
- typedef `char16_t __integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic` & `operator=` (`const atomic &`) volatile

Public Attributes

- `short _M_i`

5.343.1 Detailed Description

`template<> struct std::atomic< char16_t >`

Explicit specialization for `char16_t`.

Definition at line 459 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.344 `std::atomic< char32_t >` Struct Template Reference

Explicit specialization for `char32_t`.

Inherits `__atomic_int_base`.

Public Types

- typedef `atomic_char32_t __base_type`
- typedef `char32_t __integral_type`

Public Member Functions

- **atomic** (__integral_type __i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- int **_M_i**

5.344.1 Detailed Description

template<> struct std::atomic< char32_t >

Explicit specialization for char32_t.

Definition at line 477 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.345 std::atomic< int > Struct Template Reference

Explicit specialization for int.

Inherits [__atomic_int_base](#).

Public Types

- typedef [atomic_int](#) **__base_type**
- typedef int **__integral_type**

Public Member Functions

- **atomic** (__integral_type __i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- int **_M_i**

5.345.1 Detailed Description

`template<> struct std::atomic< int >`

Explicit specialization for int.

Definition at line 333 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.346 `std::atomic< long >` Struct Template Reference

Explicit specialization for long.

Inherits `__atomic_long_base`.

Public Types

- typedef [atomic_long](#) `__base_type`
- typedef long `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const [atomic](#) &)
- [atomic](#) & `operator=` (const [atomic](#) &) volatile

Public Attributes

- long `_M_i`

5.346.1 Detailed Description

`template<> struct std::atomic< long >`

Explicit specialization for long.

Definition at line 369 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.347 `std::atomic< long long >` Struct Template Reference

Explicit specialization for long long.

Inherits `__atomic_llong_base`.

Public Types

- typedef `atomic_llong` `__base_type`
- typedef long long `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- long long `_M_i`

5.347.1 Detailed Description

`template<> struct std::atomic< long long >`

Explicit specialization for long long.

Definition at line 405 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.348 `std::atomic< short >` Struct Template Reference

Explicit specialization for short.

Inherits `__atomic_short_base`.

Public Types

- typedef `atomic_short` `__base_type`
- typedef short `__integral_type`

Public Member Functions

- `atomic` (`__integral_type` `__i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- short `_M_i`

5.348.1 Detailed Description

`template<> struct std::atomic< short >`

Explicit specialization for short.

Definition at line 297 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.349 `std::atomic< signed char >` Struct Template Reference

Explicit specialization for signed char.

Inherits `__atomic_schar_base`.

Public Types

- typedef `atomic_schar` `__base_type`
- typedef signed char `__integral_type`

Public Member Functions

- **atomic** (__integral_type __i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- signed char **_M_i**

5.349.1 Detailed Description

template<> struct std::atomic< signed char >

Explicit specialization for signed char.

Definition at line 261 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.350 std::atomic< unsigned char > Struct Template Reference

Explicit specialization for unsigned char.

Inherits `__atomic_uchar_base`.

Public Types

- typedef [atomic_uchar](#) **__base_type**
- typedef unsigned char **__integral_type**

Public Member Functions

- **atomic** (__integral_type __i)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- unsigned char `_M_i`

5.350.1 Detailed Description

`template<> struct std::atomic< unsigned char >`

Explicit specialization for unsigned char.

Definition at line 279 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.351 `std::atomic< unsigned int >` Struct Template Reference

Explicit specialization for unsigned int.

Inherits `__atomic_uint_base`.

Public Types

- typedef [atomic_uint](#) `__base_type`
- typedef unsigned int `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const [atomic](#) &)
- [atomic](#) & `operator=` (const [atomic](#) &) volatile

Public Attributes

- unsigned int `_M_i`

5.351.1 Detailed Description

template<> struct std::atomic< unsigned int >

Explicit specialization for unsigned int.

Definition at line 351 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.352 std::atomic< unsigned long > Struct Template Reference

Explicit specialization for unsigned long.

Inherits `__atomic_ulong_base`.

Public Types

- typedef [atomic_ulong](#) `__base_type`
- typedef unsigned long `__integral_type`

Public Member Functions

- **atomic** (`__integral_type __i`)
- **atomic** (const [atomic](#) &)
- [atomic](#) & **operator=** (const [atomic](#) &) volatile

Public Attributes

- unsigned long `_M_i`

5.352.1 Detailed Description

template<> struct std::atomic< unsigned long >

Explicit specialization for unsigned long.

Definition at line 387 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.353 `std::atomic< unsigned long long >` Struct Template Reference

Explicit specialization for unsigned long long.

Inherits `__atomic_ullong_base`.

Public Types

- typedef [atomic_ullong](#) `__base_type`
- typedef unsigned long long `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const [atomic](#) &)
- [atomic](#) & `operator=` (const [atomic](#) &) volatile

Public Attributes

- unsigned long long `_M_i`

5.353.1 Detailed Description

`template<> struct std::atomic< unsigned long long >`

Explicit specialization for unsigned long long.

Definition at line 423 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.354 `std::atomic< unsigned short >` Struct Template Reference

Explicit specialization for unsigned short.

Inherits `__atomic_ushort_base`.

Public Types

- typedef `atomic_ushort` `__base_type`
- typedef unsigned short `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (const `atomic` &)
- `atomic` & `operator=` (const `atomic` &) volatile

Public Attributes

- unsigned short `_M_i`

5.354.1 Detailed Description

`template<> struct std::atomic< unsigned short >`

Explicit specialization for unsigned short.

Definition at line 315 of file `atomic`.

The documentation for this struct was generated from the following file:

- `atomic`

5.355 `std::atomic< void * >` Struct Template Reference

Explicit specialization for `void*`.

Inherits `atomic_address`.

Public Types

- typedef `atomic_address` `__base_type`
- typedef `void *` `__integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic` & `operator=` (`const atomic &`) volatile

5.355.1 Detailed Description

`template<> struct std::atomic< void * >`

Explicit specialization for `void*`.

Definition at line 207 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.356 `std::atomic< wchar_t >` Struct Template Reference

Explicit specialization for `wchar_t`.

Inherits `__atomic_wchar_t_base`.

Public Types

- typedef `atomic_wchar_t __base_type`
- typedef `wchar_t __integral_type`

Public Member Functions

- `atomic` (`__integral_type __i`)
- `atomic` (`const atomic &`)
- `atomic` & `operator=` (`const atomic &`) volatile

Public Attributes

- `wchar_t _M_i`

5.356.1 Detailed Description

`template<> struct std::atomic< wchar_t >`

Explicit specialization for `wchar_t`.

Definition at line 441 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

5.357 `std::auto_ptr< _Tp >` Class Template Reference

A simple smart pointer providing strict ownership semantics.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `auto_ptr (element_type *__p=0) throw ()`
- `auto_ptr (auto_ptr &__a) throw ()`
- `template<typename _Tp1 >`
`auto_ptr (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr (auto_ptr_ref< element_type > __ref) throw ()`
- `~auto_ptr ()`
- `element_type * get () const throw ()`
- `template<typename _Tp1 >`
`operator auto_ptr< _Tp1 > () throw ()`
- `template<typename _Tp1 >`
`operator auto_ptr_ref< _Tp1 > () throw ()`
- `element_type & operator* () const throw ()`
- `element_type * operator-> () const throw ()`
- `auto_ptr & operator= (auto_ptr &__a) throw ()`
- `template<typename _Tp1 >`
`auto_ptr & operator= (auto_ptr< _Tp1 > &__a) throw ()`
- `auto_ptr & operator= (auto_ptr_ref< element_type > __ref) throw ()`
- `element_type * release () throw ()`
- `void reset (element_type *__p=0) throw ()`

5.357.1 Detailed Description

template<typename _Tp> class std::auto_ptr< _Tp >

A simple smart pointer providing strict ownership semantics. The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyCon requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.

_GLIBCXX_RESOLVE_LIB_DEFECTS 127. auto_ptr<> conversion issues These resolutions have all been incorporated.

Definition at line 85 of file auto_ptr.h.

5.357.2 Member Typedef Documentation

5.357.2.1 template<typename _Tp> typedef _Tp std::auto_ptr< _Tp >::element_type

The pointed-to type.

Definition at line 92 of file auto_ptr.h.

5.357.3 Constructor & Destructor Documentation

5.357.3.1 template<typename _Tp> std::auto_ptr< _Tp >::auto_ptr (element_type * __p = 0) throw () [inline, explicit]

An `auto_ptr` is usually constructed from a raw pointer.

Parameters

p A pointer (defaults to NULL).

This object now *owns* the object pointed to by *p*.

Definition at line 101 of file auto_ptr.h.

5.357.3.2 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr (auto_ptr<_Tp> & __a) throw () [inline]`

An auto_ptr can be constructed from another auto_ptr.

Parameters

a Another auto_ptr of the same type.

This object now *owns* the object previously owned by *a*, which has given up ownership.

Definition at line 110 of file auto_ptr.h.

5.357.3.3 `template<typename _Tp> template<typename _Tp1 > std::auto_ptr<_Tp>::auto_ptr (auto_ptr<_Tp1> & __a) throw () [inline]`

An auto_ptr can be constructed from another auto_ptr.

Parameters

a Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.

This object now *owns* the object previously owned by *a*, which has given up ownership.

Definition at line 123 of file auto_ptr.h.

5.357.3.4 `template<typename _Tp> std::auto_ptr<_Tp>::~~auto_ptr () [inline]`

When the auto_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get ()` is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if `_Tp::~~_Tp()` throws, but this is prohibited. [17.4.3.6]/2

Definition at line 168 of file auto_ptr.h.

5.357.3.5 `template<typename _Tp> std::auto_ptr< _Tp >::auto_ptr (auto_ptr_ref< element_type > __ref) throw () [inline]`

Automatic conversions.

These operations convert an `auto_ptr` into and from an `auto_ptr_ref` automatically as needed. This allows constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(....);  
...  
auto_ptr<Base> ptr = func_returning_auto_ptr(....);
```

Definition at line 258 of file `auto_ptr.h`.

5.357.4 Member Function Documentation

5.357.4.1 `template<typename _Tp> element_type* std::auto_ptr< _Tp >::get (void) const throw () [inline]`

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This `auto_ptr` still owns the memory.

Definition at line 209 of file `auto_ptr.h`.

5.357.4.2 `template<typename _Tp> element_type& std::auto_ptr< _Tp >::operator*() const throw () [inline]`

Smart pointer dereferencing.

If this `auto_ptr` no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 179 of file `auto_ptr.h`.

5.357.4.3 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::operator-> () const throw () [inline]`

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

Definition at line 192 of file `auto_ptr.h`.

5.357.4.4 `template<typename _Tp> template<typename _Tp1 > auto_ptr& std::auto_ptr<_Tp>::operator= (auto_ptr<_Tp1> & __a) throw () [inline]`

`auto_ptr` assignment operator.

Parameters

a Another `auto_ptr` of a different but related type.

A pointer-to-`Tp1` must be convertible to a pointer-to-`Tp`/`element_type`.

This object now *owns* the object previously owned by *a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 152 of file `auto_ptr.h`.

5.357.4.5 `template<typename _Tp> auto_ptr& std::auto_ptr<_Tp>::operator= (auto_ptr<_Tp> & __a) throw () [inline]`

`auto_ptr` assignment operator.

Parameters

a Another `auto_ptr` of the same type.

This object now *owns* the object previously owned by *a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 134 of file `auto_ptr.h`.

5.357.4.6 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::release () throw () [inline]`

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This `auto_ptr` no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 223 of file `auto_ptr.h`.

5.357.4.7 `template<typename _Tp> void std::auto_ptr<_Tp>::reset (`
`element_type * __p = 0) throw () [inline]`

Forcibly deletes the managed object.

Parameters

p A pointer (defaults to NULL).

This object now *owns* the object pointed to by *p*. The previous object has been deleted.

Definition at line 238 of file `auto_ptr.h`.

The documentation for this class was generated from the following file:

- [auto_ptr.h](#)

5.358 `std::auto_ptr_ref<_Tp1>` Struct Template Reference

Public Member Functions

- `auto_ptr_ref(_Tp1 * __p)`

Public Attributes

- `_Tp1 * _M_ptr`

5.358.1 Detailed Description

`template<typename _Tp1> struct std::auto_ptr_ref< _Tp1 >`

A wrapper class to provide [auto_ptr](#) with reference semantics. For example, an [auto_ptr](#) can be assigned (or constructed from) the result of a function which returns an [auto_ptr](#) by value.

All the [auto_ptr_ref](#) stuff should happen behind the scenes.

Definition at line 46 of file `auto_ptr.h`.

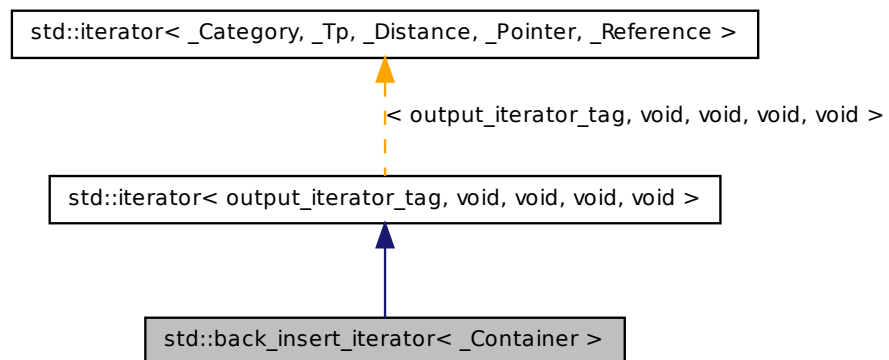
The documentation for this struct was generated from the following file:

- [auto_ptr.h](#)

5.359 `std::back_insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::back_insert_iterator< _Container >`:



Public Types

- `typedef _Container container_type`

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) [iterator_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)

Public Member Functions

- [back_insert_iterator](#) ([_Container](#) &__x)
- [back_insert_iterator](#) & [operator*](#) ()
- [back_insert_iterator](#) & [operator++](#) ()
- [back_insert_iterator](#) [operator++](#) (int)
- [back_insert_iterator](#) & [operator=](#) (const typename [_Container::value_type](#) &__value)
- [back_insert_iterator](#) & [operator=](#) (typename [_Container::value_type](#) &&__value)

Protected Attributes

- [_Container](#) * [container](#)

5.359.1 Detailed Description

template<typename [_Container](#)> class `std::back_insert_iterator<_Container>`

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using `push_back`.

Tip: Using the `back_inserter` function to create these iterators can save typing.

Definition at line 394 of file `stl_iterator.h`.

5.359.2 Member Typedef Documentation

5.359.2.1 **template<typename [_Container](#)> typedef [_Container](#) `std::back_insert_iterator<_Container>::container_type`**

A nested typedef for the type of whatever container you used.

Definition at line 402 of file `stl_iterator.h`.

5.359.2.2 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 124 of file `stl_iterator_base_types.h`.

5.359.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 120 of file `stl_iterator_base_types.h`.

5.359.2.4 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 126 of file `stl_iterator_base_types.h`.

5.359.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 128 of file `stl_iterator_base_types.h`.

5.359.2.6 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file `stl_iterator_base_types.h`.

5.359.3 Constructor & Destructor Documentation

5.359.3.1 `template<typename _Container > std::back_insert_iterator< _Container >::back_insert_iterator (_Container & __x) [inline, explicit]`

The only way to create this iterator is with a container.

Definition at line 406 of file `stl_iterator.h`.

5.359.4 Member Function Documentation

5.359.4.1 `template<typename _Container > back_insert_iterator&
std::back_insert_iterator< _Container >::operator* ()
[inline]`

Simply returns *this.

Definition at line 444 of file stl_iterator.h.

5.359.4.2 `template<typename _Container > back_insert_iterator
std::back_insert_iterator< _Container >::operator++ (int)
[inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 454 of file stl_iterator.h.

5.359.4.3 `template<typename _Container > back_insert_iterator&
std::back_insert_iterator< _Container >::operator++ ()
[inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 449 of file stl_iterator.h.

5.359.4.4 `template<typename _Container > back_insert_iterator&
std::back_insert_iterator< _Container >::operator= (const
typename _Container::value_type & __value) [inline]`

Parameters

value An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

Definition at line 428 of file stl_iterator.h.

The documentation for this class was generated from the following file:

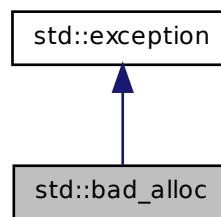
- [stl_iterator.h](#)

5.360 std::bad_alloc Class Reference

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Inheritance diagram for `std::bad_alloc`:



Public Member Functions

- virtual const char * `what` () const throw ()

5.360.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Definition at line 56 of file `new`.

5.360.2 Member Function Documentation

5.360.2.1 virtual const char* std::bad_alloc::what () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from `std::exception`.

The documentation for this class was generated from the following file:

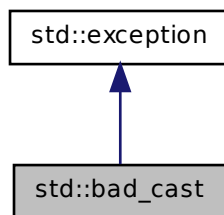
- [new](#)

5.361 `std::bad_cast` Class Reference

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Inheritance diagram for `std::bad_cast`:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.361.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Definition at line 191 of file `typeinfo`.

5.361.2 Member Function Documentation

5.361.2.1 `virtual const char* std::bad_cast::what () const throw ()` `[virtual]`

Returns a C-style character string describing the general cause of the current error.

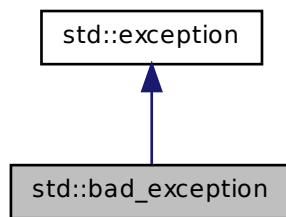
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

5.362 `std::bad_exception` Class Reference

Inheritance diagram for `std::bad_exception`:



Public Member Functions

- `virtual const char * what () const throw ()`

5.362.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

Definition at line 74 of file `exception`.

5.362.2 Member Function Documentation

5.362.2.1 virtual const char* std::bad_exception::what () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

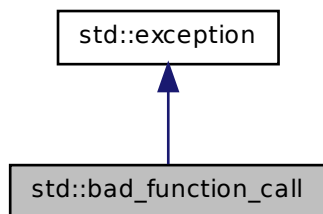
The documentation for this class was generated from the following file:

- [exception](#)

5.363 std::bad_function_call Class Reference

Exception class thrown when class template function's operator() is called with an empty target.

Inheritance diagram for std::bad_function_call:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.363.1 Detailed Description

Exception class thrown when class template function's operator() is called with an empty target.

Definition at line 1471 of file functional.

5.363.2 Member Function Documentation

5.363.2.1 `virtual const char* std::exception::what () const throw ()` [`virtual`, `inherited`]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_exception](#), [std::bad_alloc](#), [std::bad_cast](#), [std::bad_typeid](#), [std::future_error](#), [std::logic_error](#), [std::runtime_error](#), [std::tr1::bad_weak_ptr](#), and [std::ios_base::failure](#).

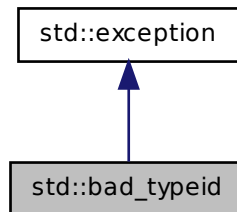
The documentation for this class was generated from the following file:

- [functional](#)

5.364 `std::bad_typeid` Class Reference

Thrown when a NULL pointer in a `typeid` expression is used.

Inheritance diagram for `std::bad_typeid`:



Public Member Functions

- `virtual const char * what () const throw ()`

- typedef `_CharT` [char_type](#)
- typedef `traits_type::int_type` [int_type](#)
- typedef `traits_type::off_type` [off_type](#)
- typedef `traits_type::pos_type` [pos_type](#)
- typedef `_Traits` [traits_type](#)

Public Member Functions

- [basic_filebuf](#) ()
- virtual [~basic_filebuf](#) ()
- [__filebuf_type](#) * [close](#) ()
- [streamsize](#) [in_avail](#) ()
- bool [is_open](#) () const throw ()
- [__filebuf_type](#) * [open](#) (const char * __s, [ios_base::openmode](#) __mode)
- [__filebuf_type](#) * [open](#) (const std::string & __s, [ios_base::openmode](#) __mode)
- [int_type](#) [sbumpc](#) ()
- [int_type](#) [sgetc](#) ()
- [streamsize](#) [sgetn](#) ([char_type](#) * __s, [streamsize](#) __n)
- [int_type](#) [snextc](#) ()
- [int_type](#) [sputbackc](#) ([char_type](#) __c)
- [int_type](#) [sputc](#) ([char_type](#) __c)
- [streamsize](#) [sputn](#) (const [char_type](#) * __s, [streamsize](#) __n)
- void [stossc](#) ()
- [int_type](#) [sungetc](#) ()

Protected Member Functions

- void [_M_allocate_internal_buffer](#) ()
- bool [_M_convert_to_external](#) ([char_type](#) *, [streamsize](#))
- void [_M_create_pback](#) ()
- void [_M_destroy_internal_buffer](#) () throw ()
- void [_M_destroy_pback](#) () throw ()
- int [_M_get_ext_pos](#) (__state_type & __state)
- [pos_type](#) [_M_seek](#) ([off_type](#) __off, [ios_base::seekdir](#) __way, __state_type __state)
- void [_M_set_buffer](#) ([streamsize](#) __off)
- bool [_M_terminate_output](#) ()
- void [gbump](#) (int __n)
- virtual void [imbue](#) (const [locale](#) & __loc)
- virtual [int_type](#) [overflow](#) ([int_type](#) __c=_Traits::eof())
- virtual [int_type](#) [overflow](#) ([int_type](#)=[traits_type::eof](#)())
- virtual [int_type](#) [pbackfail](#) ([int_type](#) __c=_Traits::eof())

- virtual [int_type pbackfail](#) ([int_type](#)=traits_type::eof())
- void [pbump](#) ([int](#) __n)
- virtual [pos_type seekoff](#) ([off_type](#) __off, [ios_base::seekdir](#) __way, [ios_base::openmode](#) __mode=[ios_base::in](#)|[ios_base::out](#))
- virtual [pos_type seekoff](#) ([off_type](#), [ios_base::seekdir](#), [ios_base::openmode](#)=[ios_base::in](#)|[ios_base::out](#))
- virtual [pos_type seekpos](#) ([pos_type](#) __pos, [ios_base::openmode](#) __mode=[ios_base::in](#)|[ios_base::out](#))
- virtual [pos_type seekpos](#) ([pos_type](#), [ios_base::openmode](#)=[ios_base::in](#)|[ios_base::out](#))
- virtual [__streambuf_type * setbuf](#) ([char_type](#) *__s, [streamsize](#) __n)
- virtual [basic_streambuf< char_type, _Traits > * setbuf](#) ([char_type](#) *, [streamsize](#))
- void [setg](#) ([char_type](#) *__gbeg, [char_type](#) *__gnext, [char_type](#) *__gend)
- void [setp](#) ([char_type](#) *__pbeg, [char_type](#) *__pend)
- virtual [streamsize showmanyc](#) ()
- virtual [int sync](#) ()
- virtual [int_type uflow](#) ()
- virtual [int_type underflow](#) ()
- virtual [streamsize xsgetn](#) ([char_type](#) *__s, [streamsize](#) __n)
- virtual [streamsize xsgetn](#) ([char_type](#) *__s, [streamsize](#) __n)
- virtual [streamsize xspun](#) (const [char_type](#) *__s, [streamsize](#) __n)
- virtual [streamsize xspun](#) (const [char_type](#) *__s, [streamsize](#) __n)

- [char_type * eback](#) () const
- [char_type * gptr](#) () const
- [char_type * egptr](#) () const

- [char_type * pbase](#) () const
- [char_type * pptr](#) () const
- [char_type * ep_ptr](#) () const

Protected Attributes

- [char_type * _M_buf](#)
- [bool _M_buf_allocated](#)
- [size_t _M_buf_size](#)
- [const __codecvt_type * _M_codecvt](#)
- [char * _M_ext_buf](#)
- [streamsize _M_ext_buf_size](#)
- [char * _M_ext_end](#)
- [const char * _M_ext_next](#)
- [__file_type _M_file](#)
- [__c_lock _M_lock](#)

- [ios_base::openmode _M_mode](#)
 - [bool _M_reading](#)
 - [__state_type _M_state_beg](#)
 - [__state_type _M_state_cur](#)
 - [__state_type _M_state_last](#)
 - [bool _M_writing](#)
-
- [char_type _M_pback](#)
 - [char_type * _M_pback_cur_save](#)
 - [char_type * _M_pback_end_save](#)
 - [bool _M_pback_init](#)

Friends

- `template<bool _IsMove, typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`
`_CharT2 >, _CharT2 *)`
 - `streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *,`
`bool &)`
 - `class basic_ios< char_type, traits_type >`
 - `class basic_istream< char_type, traits_type >`
 - `class basic_ostream< char_type, traits_type >`
 - `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf -`
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >, -`
`istreambuf_iterator< _CharT2 >, const _CharT2 &)`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, -`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
 - `class ios_base`
 - `class istreambuf_iterator< char_type, traits_type >`
 - `template<typename _CharT2, typename _Traits2 >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, _CharT2 *)`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
 - `class ostreambuf_iterator< char_type, traits_type >`
-
- [char_type * _M_in_beg](#)
 - [char_type * _M_in_cur](#)
 - [char_type * _M_in_end](#)

- `char_type * _M_out_beg`
- `char_type * _M_out_cur`
- `char_type * _M_out_end`
- `locale _M_buf_locale`
- `locale pubimbue (const locale &__loc)`
- `locale getloc () const`
- `__streambuf_type * pubsetbuf (char_type *__s, streamsize __n)`
- `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `int pubsync ()`

5.365.1 Detailed Description

template<typename _CharT, typename _Traits> class std::basic_filebuf< _CharT, _Traits >

The actual work of input and output (for files).

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Definition at line 67 of file `fstream`.

5.365.2 Member Typedef Documentation

5.365.2.1 **template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_filebuf< _CharT, _Traits >::__streambuf_type**

This is a non-standard type.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 77 of file `fstream`.

5.365.2.2 **template<typename _CharT, typename _Traits> typedef _CharT std::basic_filebuf< _CharT, _Traits >::char_type**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Reimplemented in [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#).

Definition at line 71 of file fstream.

5.365.2.3 template<typename _CharT, typename _Traits> typedef traits_type::int_type std::basic_filebuf<_CharT, _Traits>::int_type

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Reimplemented in [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#).

Definition at line 73 of file fstream.

5.365.2.4 template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_filebuf<_CharT, _Traits>::off_type

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Reimplemented in [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#).

Definition at line 75 of file fstream.

5.365.2.5 template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_filebuf<_CharT, _Traits>::pos_type

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Reimplemented in [__gnu_cxx::enc_filebuf<_CharT>](#), and [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#).

Definition at line 74 of file fstream.

5.365.2.6 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_filebuf<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Reimplemented in `__gnu_cxx::enc_filebuf<_CharT>`, and `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`.

Definition at line 72 of file `fstream`.

5.365.3 Constructor & Destructor Documentation

5.365.3.1 `template<typename _CharT, typename _Traits> std::basic_filebuf< _CharT, _Traits>::basic_filebuf ()`

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

Definition at line 79 of file `fstream.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::_M_buf_locale`.

5.365.3.2 `template<typename _CharT, typename _Traits> virtual std::basic_filebuf<_CharT, _Traits>::~~basic_filebuf () [inline, virtual]`

The destructor closes the file first.

Definition at line 214 of file `fstream`.

5.365.4 Member Function Documentation

5.365.4.1 `template<typename _CharT, typename _Traits> void std::basic_filebuf<_CharT, _Traits>::_M_create_pback () [inline, protected]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 172 of file `fstream`.

5.365.4.2 `template<typename _CharT, typename _Traits> void
std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw
() [inline, protected]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 189 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.4.3 `template<typename _CharT, typename _Traits> void
std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (streamsize
__off) [inline, protected]`

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 390 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.4.4 `template<typename _CharT , typename _Traits > basic_filebuf<
_CharT, _Traits >::_filebuf_type * std::basic_filebuf< _CharT,
_Traits >::close ()`

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::is_open()`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.365.4.5 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eback () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.4.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::egptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.4.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::epptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.365.4.8 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::gbump (int __n)
[inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

5.365.4.9 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf<_CharT, _Traits>::getloc () const
[inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

5.365.4.10 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::gptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.4.11 `template<typename _CharT, typename _Traits> void
std::basic_filebuf< _CharT, _Traits >::imbue (const locale &)
[protected, virtual]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 912 of file fstream.tcc.

References `std::basic_filebuf< _CharT, _Traits >::_M_ext_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_ext_next`, `std::basic_filebuf< _CharT, _Traits >::_M_mode`, `std::basic_filebuf< _CharT, _Traits >::_M_reading`, `std::basic_filebuf< _CharT, _Traits >::_M_set_buffer()`, `std::ios_base::cur`, `std::basic_streambuf< _CharT, _Traits >::eback()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

5.365.4.12 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::in_avail ()
[inline, inherited]`

Looking ahead into the stream.

Definition at line 94 of file fstream.tcc.

References std::basic_filebuf< _CharT, _Traits >::_M_mode, std::basic_filebuf< _CharT, _Traits >::_M_reading, std::basic_filebuf< _CharT, _Traits >::_M_set_buffer(), std::ios_base::ate, std::basic_filebuf< _CharT, _Traits >::close(), std::ios_base::end, and std::basic_filebuf< _CharT, _Traits >::is_open().

5.365.4.15 `template<typename _CharT, typename _Traits> __filebuf_type*
std::basic_filebuf< _CharT, _Traits >::open (const std::string &
__s, ios_base::openmode __mode) [inline]`

Opens an external file.

Parameters

- s* The name of the file.
- mode* The open mode flags.

Returns

this on success, NULL on failure

Definition at line 275 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::open().

5.365.4.16 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::overflow (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

- c* An additional character to consume.

Returns

eof() to indicate failure, something else (usually *c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not eof().

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented in [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#).

Definition at line 746 of file streambuf.

Referenced by [std::basic_streambuf<_CharT, _Traits>::xsputn\(\)](#).

5.365.4.17 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of [gptr\(\)](#), [eback\(\)](#), and [pptr\(\)](#) are the same as for [underflow\(\)](#).

Note

Base class version does nothing, returns eof().

Reimplemented in [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#).

Definition at line 702 of file streambuf.

5.365.4.18 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::pbase () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::sync().

5.365.4.19 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::pbump (int __n)
[inline, protected, inherited]`

Moving the write position.

Parameters

- n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by std::basic_streambuf< _CharT, _Traits >::xsputn().

5.365.4.20 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_streambuf< _CharT, _Traits >::xsputn().

5.365.4.21 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file `streambuf`.

5.365.4.22 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

5.365.4.23 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 244 of file streambuf.

5.365.4.24 `template<typename _CharT, typename _Traits>
__streambuf_type* std::basic_streambuf< _CharT, _Traits
>::pubsetbuf(char_type * __s, streamsize __n) [inline,
inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file streambuf.

5.365.4.25 `template<typename _CharT, typename _Traits> int
std::basic_streambuf< _CharT, _Traits >::pubsync ()
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 249 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

5.365.4.26 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

5.365.4.27 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf< _CharT, _Traits >::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

5.365.4.28 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf< _CharT, _Traits >::seekpos(pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

5.365.4.29 `template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type,_Traits>* std::basic_streambuf<
_CharT,_Traits >::setbuf(char_type*, streamsize) [inline,
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

```
5.365.4.30 template<typename _CharT, typename _Traits > basic_filebuf<
    _CharT, _Traits >::__streambuf_type * std::basic_filebuf<
    _CharT, _Traits >::setbuf ( char_type * __s, streamsize __n )
    [protected, virtual]
```

Manipulates the buffer.

Parameters

s Pointer to a buffer area.

n Size of *s*.

Returns

`this`

If no file has been opened, and both *s* and *n* are zero, then the stream becomes unbuffered. Otherwise, *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 686 of file `fstream.tcc`.

References `std::basic_filebuf< _CharT, _Traits >::_M_buf`, `std::basic_filebuf< _CharT, _Traits >::_M_buf_size`, and `std::basic_filebuf< _CharT, _Traits >::is_open()`.

```
5.365.4.31 template<typename _CharT, typename _Traits> void
    std::basic_streambuf< _CharT, _Traits >::setg ( char_type *
    __gbeg, char_type * __gnext, char_type * __gend ) [inline,
    protected, inherited]
```

Setting the three read area pointers.

Parameters

gbeg A pointer.

gnext A pointer.

gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

5.365.4.32 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setp (char_type
* __pbeg, char_type * __pend) [inline, protected,
inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.

pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `pptr()`

Definition at line 533 of file streambuf.

5.365.4.33 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sgetc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

5.365.4.34 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::sgetn (char_type * __s,
streamsize __n) [inline, inherited]`

Entry point for xsgetn.

Parameters

s A buffer area.

n A count.

Returns xsgetn(s,n). The effect is to fill s[0] through s[n-1] with characters from the input sequence, if possible.

Definition at line 335 of file streambuf.

5.365.4.35 `template<typename _CharT , typename _Traits > streamsize
std::basic_filebuf< _CharT, _Traits >::showmanyc ()
[protected, virtual]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to [underflow\(\)](#) will not return traits::eof() until at least that number of characters have been supplied. If [showmanyc\(\)](#) returns -1, then calls to [underflow\(\)](#) or [uflow\(\)](#) will fail.
[27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return eof() but that they will return immediately.*

The standard adds that *the morphemes of showmanyc are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 178 of file fstream.tcc.

References [std::basic_filebuf< _CharT, _Traits >::M_mode](#), [std::ios_base::binary](#), [std::basic_streambuf< _CharT, _Traits >::egptr\(\)](#), [std::basic_streambuf< _CharT, _Traits >::gpptr\(\)](#), [std::ios_base::in](#), and [std::basic_filebuf< _CharT, _Traits >::is_open\(\)](#).

5.365.4.36 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::snextc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

5.365.4.37 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sputbackc (char_type
__c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

c The character to push back.

Returns

The previous character, if possible.

Similar to `sungetc()`, but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::putback()`.

5.365.4.38 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sputc (char_type __c)
[inline, inherited]`

Entry point for all single-character output functions.

Parameters

c A character to output.

Returns

`c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `c` in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

5.365.4.39 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sputn (const char_type
* __s, streamsize __n) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

`s` A buffer read area.

`n` A count.

One of two public output functions.

Returns `xsputn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.365.4.40 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::stoss () [inline,
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

5.365.4.41 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sungetc () [inline,
inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns [pbackfail\(\)](#). The effect is to *unget* the last character *gotten*.

Definition at line 375 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::unget()`.

5.365.4.42 `template<typename _CharT , typename _Traits > int
std::basic_filebuf< _CharT, _Traits >::sync () [protected,
virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 895 of file fstream.tcc.

References `std::basic_streambuf< _CharT, _Traits >::pbase()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

5.365.4.43 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::uflow () [inline,
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 678 of file `streambuf`.

5.365.4.44 `template<typename _CharT, typename _Traits> basic_filebuf<_CharT, _Traits>::int_type std::basic_filebuf<_CharT, _Traits>::underflow() [protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input `streambuf` can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 204 of file `fstream.tcc`.

References `std::basic_filebuf<_CharT, _Traits>::_M_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback()`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_buf_size`, `std::basic_filebuf<_CharT, _Traits>::_M_ext_next`, `std::basic_filebuf<_CharT, _Traits>::_M_mode`, `std::basic_filebuf<_CharT, _Traits>::_M_reading`, `std::basic_filebuf<_CharT, _Traits>::_M_set_buffer()`, `std::basic_streambuf<_CharT, _Traits>::eback()`, `std::basic_streambuf<_CharT, _Traits>::egptr()`, `std::basic_streambuf<_CharT, _Traits>::gptr()`, `std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::in()`, `std::ios_base::in`, and `std::min()`.

5.365.4.45 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::xsgetn (char_type *
__s, streamsize __n) [protected, virtual, inherited]`

Multiple character extraction.

Parameters

- s* A buffer area.
- n* Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills *s*[0] through *s*[*n*-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either *n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

5.365.4.46 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::xspn (const
char_type * __s, streamsize __n) [protected, virtual,
inherited]`

Multiple character insertion.

Parameters

- s* A buffer area.
- n* Maximum number of characters to write.

Returns

The number of characters written.

Writes *s*[0] through *s*[*n*-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#).

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf< _CharT, _Traits >::epptr()`, `std::min()`, `std::basic_streambuf< _CharT, _Traits >::overflow()`, `std::basic_streambuf< _CharT, _Traits >::pbump()`, and `std::basic_streambuf< _CharT, _Traits >::pptr()`.

5.365.5 Member Data Documentation

5.365.5.1 `template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf` [protected]

Pointer to the beginning of internal buffer.

Definition at line 109 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::setbuf()`.

5.365.5.2 `template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale` [protected, inherited]

Current locale setting.

Definition at line 188 of file `streambuf`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::basic_filebuf()`.

5.365.5.3 `template<typename _CharT, typename _Traits> size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size` [protected]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 116 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::setbuf()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.5.4 `template<typename _CharT, typename _Traits> char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected]`

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to [eback\(\)](#).

Definition at line 151 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.5.5 `template<typename _CharT, typename _Traits> streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size [protected]`

Size of buffer held by `_M_ext_buf`.

Definition at line 156 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.5.6 `template<typename _CharT, typename _Traits> const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected]`

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to [egptr\(\)](#).

Definition at line 163 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.5.7 `template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file `streambuf`.

5.365.5.8 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file `streambuf`.

5.365.5.9 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.365.5.10 `template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits
>::_M_mode [protected]`

Place to stash in || out || in | out settings for current filebuf.

Definition at line 94 of file `fstream`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.365.5.11 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_beg
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

5.365.5.12 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

5.365.5.13 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 185 of file streambuf.

5.365.5.14 `template<typename _CharT, typename _Traits> char_type
std::basic_filebuf< _CharT, _Traits >::_M_pback [protected]`

Necessary bits for putback buffer management.

Note

 pbacks of over one character are not currently supported.

Definition at line 137 of file fstream.

5.365.5.15 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save
[protected]`

Necessary bits for putback buffer management.

Note

 pbacks of over one character are not currently supported.

Definition at line 138 of file fstream.

5.365.5.16 `template<typename _CharT, typename _Traits> char_type*
std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save
[protected]`

Necessary bits for putback buffer management.

Note

 pbacks of over one character are not currently supported.

Definition at line 139 of file fstream.

5.365.5.17 `template<typename _CharT, typename _Traits> bool
std::basic_filebuf< _CharT, _Traits >::_M_pback_init
[protected]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 140 of file fstream.

```
5.365.5.18  template<typename _CharT, typename _Traits> bool
             std::basic_filebuf< _CharT, _Traits >::_M_reading
             [protected]
```

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 128 of file fstream.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::open()`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

The documentation for this class was generated from the following files:

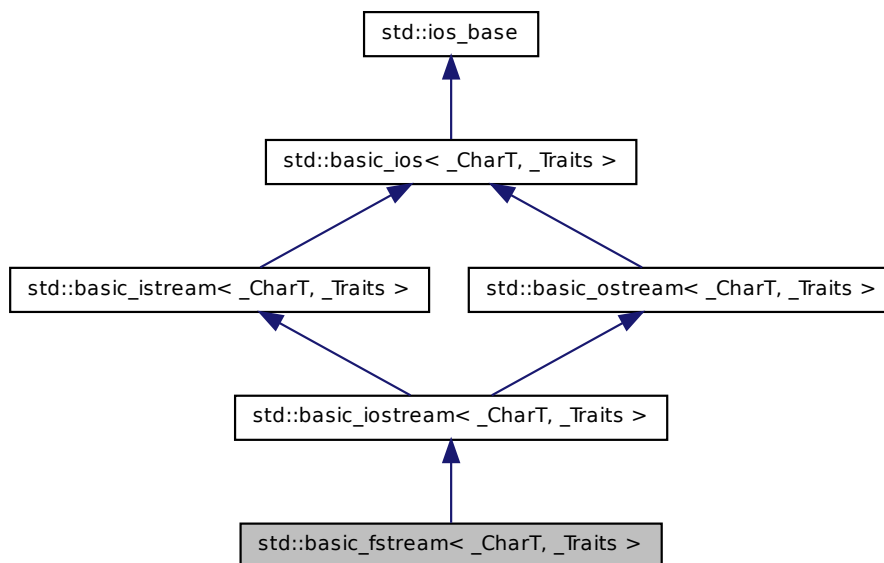
- [fstream](#)
- [fstream.tcc](#)

5.366 `std::basic_fstream< _CharT, _Traits >` Class Template Reference

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for std::basic_fstream< _CharT, _Traits >:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_ios< char_type, traits_type > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`

- typedef `_CharT` [char_type](#)
- typedef `_CharT` [char_type](#)
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(* `event_callback`)(event, `ios_base` &, int)
- typedef `_Ios_Fmtflags` [fmtflags](#)
- typedef `traits_type::int_type` [int_type](#)
- typedef `_Traits::int_type` [int_type](#)
- typedef int `io_state`
- typedef `_Ios_Iostate` [iostate](#)
- typedef `traits_type::off_type` [off_type](#)
- typedef `_Traits::off_type` [off_type](#)
- typedef int `open_mode`
- typedef `_Ios_Openmode` [openmode](#)
- typedef `_Traits::pos_type` [pos_type](#)
- typedef `traits_type::pos_type` [pos_type](#)
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` [seekdir](#)
- typedef `std::streamoff` [streamoff](#)
- typedef `std::streampos` [streampos](#)
- typedef `_Traits` [traits_type](#)
- typedef `_Traits` [traits_type](#)

- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __-`
`num_put_type`

Public Member Functions

- [basic_fstream](#) ()
- [basic_fstream](#) (const char *__s, `ios_base::openmode` __mode=`ios_base::in|ios_base::out`)
- [basic_fstream](#) (const `std::string` &__s, `ios_base::openmode` __mode=`ios_base::in|ios_base::out`)
- [~basic_fstream](#) ()
- const `locale` & [_M_getloc](#) () const
- void [_M_setstate](#) (`iostate` __state)
- bool [bad](#) () const
- void [clear](#) (`iostate` __state=`goodbit`)
- void [close](#) ()
- `basic_ios` & [copyfmt](#) (const `basic_ios` &__rhs)
- bool [eof](#) () const
- `iostate` [exceptions](#) () const
- void [exceptions](#) (`iostate` __except)

- `bool fail () const`
- `char_type fill () const`
- `char_type fill (char_type __ch)`
- `fmtflags flags () const`
- `fmtflags flags (fmtflags __fmtfl)`
- `__ostream_type & flush ()`
- `streamsize gcount () const`
- `template<>`
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type __delim)`
- `template<>`
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)`
- `locale getloc () const`
- `bool good () const`
- `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n)`
- `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
- `template<>`
`basic_istream< char > & ignore (streamsize __n)`
- `template<>`
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
- `locale imbue (const locale &__loc)`
- `bool is_open ()`
- `bool is_open () const`
- `long & iword (int __ix)`
- `char narrow (char_type __c, char __dfault) const`
- `void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`
- `__filebuf_type * rdbuf () const`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`

- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl)
 - void [setstate](#) ([iostate](#) __state)
 - [pos_type](#) [tellp](#) ()
 - [basic_ostream](#)< _CharT, _Traits > * [tie](#) () const
 - [basic_ostream](#)< _CharT, _Traits > * [tie](#) ([basic_ostream](#)< _CharT, _Traits > * __tistr)
 - void [unsetf](#) ([fmtflags](#) __mask)
 - [char_type](#) [widen](#) (char __c) const
 - [streamsize](#) [width](#) ([streamsize](#) __wide)
 - [streamsize](#) [width](#) () const
-
- [__istream_type](#) & [operator](#)>> ([__istream_type](#) &(__pf)(__istream_type &))
 - [__istream_type](#) & [operator](#)>> ([__ios_type](#) &(__pf)(__ios_type &))
 - [__istream_type](#) & [operator](#)>> ([ios_base](#) &(__pf)([ios_base](#) &))

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type [std::basic_istream::sentry](#) with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, [ios_base::badbit](#) will be turned on in the stream's error state without causing an [ios_base::failure](#) to be thrown. The original exception will then be rethrown.

- [__istream_type](#) & [operator](#)>> (bool &__n)
- [__istream_type](#) & [operator](#)>> (short &__n)
- [__istream_type](#) & [operator](#)>> (unsigned short &__n)
- [__istream_type](#) & [operator](#)>> (int &__n)
- [__istream_type](#) & [operator](#)>> (unsigned int &__n)
- [__istream_type](#) & [operator](#)>> (long &__n)
- [__istream_type](#) & [operator](#)>> (unsigned long &__n)
- [__istream_type](#) & [operator](#)>> (long long &__n)
- [__istream_type](#) & [operator](#)>> (unsigned long long &__n)
- [__istream_type](#) & [operator](#)>> (float &__f)
- [__istream_type](#) & [operator](#)>> (double &__f)
- [__istream_type](#) & [operator](#)>> (long double &__f)
- [__istream_type](#) & [operator](#)>> (void *&__p)
- [__istream_type](#) & [operator](#)>> ([__streambuf_type](#) * __sb)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`
- `streamsize readsome (char_type *__s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator void * () const`
- `bool operator! () const`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

Static Public Member Functions

- static bool `sync_with_stdio` (bool __sync=true)
- static int `xalloc` () throw ()

Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`

- static const `openmode` `ate`
- static const `iostate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iostate` `eofbit`
- static const `iostate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

Protected Types

- enum { `_S_local_word_size` }

Protected Member Functions

- void `_M_cache_locale` (const `locale` &__loc)
- void `_M_call_callbacks` (`event` __ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename _ValueT >
 `_istream_type` & `_M_extract` (_ValueT &__v)
- `_Words` & `_M_grow_words` (int __index, bool __iword)

- void **_M_init** () throw ()
- template<typename _ValueT >
 __ostream_type & **_M_insert** (_ValueT __v)
- void **init** (basic_streambuf< _CharT, _Traits > *__sb)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize **_M_gcount**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

Friends

- class **sentry**
- class **sentry**

5.366.1 Detailed Description

template<typename _CharT, typename _Traits> class std::basic_fstream< _CharT, _Traits >

Controlling input and output for files.

This class supports reading from and writing to named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Definition at line 759 of file fstream.

5.366.2 Member Typedef Documentation

5.366.2.1 `template<typename _CharT, typename _Traits> typedef
 ctype<_CharT> std::basic_istream< _CharT, _Traits
 >::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file istream.

5.366.2.2 `template<typename _CharT, typename _Traits> typedef
 ctype<_CharT> std::basic_ostream< _CharT, _Traits
 >::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file ostream.

5.366.2.3 `template<typename _CharT, typename _Traits> typedef
 num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
 > std::basic_istream< _CharT, _Traits >::__num_get_type
 [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file istream.

5.366.2.4 `template<typename _CharT, typename _Traits> typedef
 num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
 > std::basic_ostream< _CharT, _Traits >::__num_put_type
 [inherited]`

These are non-standard types.

Reimplemented in [std::basic_ostream< _CharT, _Traits >](#), [std::basic_ostream< char, _Traits >](#), and [std::basic_ostream< char >](#).

Definition at line 84 of file basic_ios.h.

5.366.2.5 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file ostream.

5.366.2.6 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_istream<_CharT, _Traits>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istream<_CharT, _Traits, _Alloc>](#).

Definition at line 59 of file istream.

5.366.2.7 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_fstream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits>](#).

Definition at line 763 of file fstream.

5.366.2.8 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the [ios_base](#) object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios_base](#) and [basic_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 435 of file [ios_base.h](#).

5.366.2.9 typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]

This is a bitmask type.

_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 254 of file [ios_base.h](#).

5.366.2.10 `template<typename _CharT, typename _Traits> typedef
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istringstream< _CharT, _Traits, _Alloc >](#).

Definition at line 60 of file istream.

5.366.2.11 `template<typename _CharT , typename _Traits > typedef
traits_type::int_type std::basic_fstream< _CharT, _Traits
>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream< _CharT, _Traits >](#).

Definition at line 765 of file fstream.

5.366.2.12 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file ios_base.h.

5.366.2.13 `template<typename _CharT, typename _Traits> typedef
_Traits::off_type std::basic_istream< _CharT, _Traits >::off_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 62 of file `istream`.

5.366.2.14 `template<typename _CharT , typename _Traits > typedef
traits_type::off_type std::basic_fstream< _CharT, _Traits
>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream< _CharT, _Traits >](#).

Definition at line 767 of file `fstream`.

5.366.2.15 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 360 of file `ios_base.h`.

5.366.2.16 `template<typename _CharT , typename _Traits > typedef
traits_type::pos_type std::basic_fstream< _CharT, _Traits
>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 766 of file `fstream`.

5.366.2.17 `template<typename _CharT, typename _Traits> typedef
_Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istringstream< _CharT, _Traits, _Alloc >](#).

Definition at line 61 of file `istream`.

5.366.2.18 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 392 of file `ios_base.h`.

5.366.2.19 `template<typename _CharT , typename _Traits > typedef _Traits
std::basic_fstream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream< _CharT, _Traits >](#).

Definition at line 764 of file `fstream`.

5.366.2.20 `template<typename _CharT, typename _Traits> typedef`
`_Traits std::basic_istream< _CharT, _Traits >::traits_type`
`[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 63 of file `istream`.

5.366.3 Member Enumeration Documentation

5.366.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 418 of file `ios_base.h`.

5.366.4 Constructor & Destructor Documentation

5.366.4.1 `template<typename _CharT , typename _Traits >`
`std::basic_fstream< _CharT, _Traits >::basic_fstream ()`
`[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 786 of file `fstream`.

5.366.4.2 `template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::basic_fstream (const char *
__s, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, explicit]`

Create an input/output file stream.

Parameters

s Null terminated string specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 799 of file `fstream`.

5.366.4.3 `template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::basic_fstream
(const std::string & __s, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, explicit]`

Create an input/output file stream.

Parameters

s Null terminated string specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

Definition at line 814 of file `fstream`.

5.366.4.4 `template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::~~basic_fstream ()
[inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 829 of file `fstream`.

5.366.5 Member Function Documentation

5.366.5.1 `const locale& std::ios_base::_M_getloc () const [inline, inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

5.366.5.2 `template<typename _CharT, typename _Traits> void std::basic_ostream< _CharT, _Traits >::_M_write (const char_type * __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

5.366.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.366.5.4 `template<typename _CharT , typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file basic_ios.tcc.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.366.5.5 `template<typename _CharT , typename _Traits > void
std::basic_fstream< _CharT, _Traits >::close () [inline]`

Close the file.

Calls [std::basic_filebuf::close\(\)](#). If that function fails, failbit is set in the stream's error state.

Definition at line 909 of file fstream.

5.366.5.6 `template<typename _CharT , typename _Traits > basic_ios<
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
const basic_ios< _CharT, _Traits > & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

5.366.5.7 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.366.5.8 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions() const [inline, inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.366.5.9 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions (iostate __except) [inline, inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.366.5.10 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::fail () const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

5.366.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file basic_ios.h.

5.366.5.12 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill () const [inline,
inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.366.5.13 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.366.5.14 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.366.5.15 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & std::basic_ostream< _CharT, _Traits>::flush () [inherited]`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf ()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ()->pubsync ()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::flush()`.

5.366.5.16 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::gcount () const
[inline, inherited]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.366.5.17 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits
>::get (void) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.18 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
char_type & __c) [inherited]`

Simple extraction.

Parameters

`c` The character in which to store data.

Returns

`*this`

Tries to extract a character and store it in `c`. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.19 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::get (char_type * __s, streamsize __n, char_type __delim)`
[*inherited*]

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

delim A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.366.5.20 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

5.366.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

5.366.5.22 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (__streambuf_type & __sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file `istream`.

5.366.5.23 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

s A character array in which to store the data.

n Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

5.366.5.24 `template<typename _CharT, typename _Traits > basic_istream<
 _CharT, _Traits > & std::basic_istream< _CharT, _Traits
 >::getline (char_type * __s, streamsize __n, char_type __delim)
 [inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.366.5.25 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.366.5.26 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::good() const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.366.5.27 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::ignore
(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `n != std::numeric_limits<int>::max()`, `n` characters are extracted

- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.28 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::ignore (streamsize __n) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.366.5.29 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::ignore (streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sputc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.366.5.30 `template<typename _CharT, typename _Traits> locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from [std::ios_base](#).

Definition at line 113 of file basic_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.366.5.31 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.366.5.32 `template<typename _CharT, typename _Traits> bool
std::basic_fstream< _CharT, _Traits >::is_open () [inline]`

Wrapper to test for an open file.

Returns

`rdbuf() -> is_open()`

Definition at line 848 of file fstream.

5.366.5.33 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.366.5.34 `template<typename _CharT, typename _Traits> char
std::basic_ios< _CharT, _Traits >::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

5.366.5.35 `template<typename _CharT, typename _Traits> void
std::basic_fstream<_CharT, _Traits>::open (const char * __s,
ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s, mode)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 869 of file `fstream`.

5.366.5.36 `template<typename _CharT, typename _Traits> void
std::basic_fstream<_CharT, _Traits>::open (const std::string &
__s, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s, mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 890 of file `fstream`.

5.366.5.37 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.366.5.38 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::operator! () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.366.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (unsigned long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file `ostream`.

5.366.5.40 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits >::operator<< (unsigned short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.366.5.41 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.366.5.42 `template<typename _CharT , typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits
>::operator<< (short __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.366.5.43 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 108 of file ostream.

5.366.5.44 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 117 of file ostream.

5.366.5.45 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 127 of file ostream.

5.366.5.46 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.366.5.47 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (bool __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.366.5.48 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int
__n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

5.366.5.49 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.366.5.50 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.366.5.51 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (const void *
__p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.366.5.52 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<(int __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.366.5.53 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.366.5.54 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.366.5.55 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) [inherited]`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.56 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

5.366.5.57 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

5.366.5.58 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file istream.

5.366.5.59 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file istream.

5.366.5.60 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 131 of file istream.

5.366.5.61 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 203 of file istream.

5.366.5.62 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> (unsigned int
& __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 181 of file istream.

5.366.5.63 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> (double &
__f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 207 of file istream.

5.366.5.64 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits >::operator>> (bool & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file istream.

5.366.5.65 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file istream.

5.366.5.66 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (short & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.67 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.68 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (__istream_type &(*)(__istream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 120 of file istream.

5.366.5.69 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

5.366.5.70 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

5.366.5.71 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.72 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (void *& __p) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file `istream`.

5.366.5.73 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek (void) [inherited]`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.74 streamsize std::ios_base::precision () const [inline, inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

5.366.5.75 streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.366.5.76 template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (char_type __c) [inherited]

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.366.5.77 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (char_type __c) [inherited]`

Unextracting a single character.

Parameters

`c` The character to push back into the input stream.

Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

5.366.5.78 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.366.5.79 `template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file basic_ios.tcc.

References `std::basic_ios< _CharT, _Traits >::clear()`.

5.366.5.80 `template<typename _CharT, typename _Traits > __filebuf_type* std::basic_fstream< _CharT, _Traits >::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current [basic_filebuf](#) buffer.

This hides both signatures of [std::basic_ios::rdbuf\(\)](#).

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 840 of file `fstream`.

5.366.5.81 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate() const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See [std::ios_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

5.366.5.82 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read(
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is [good\(\)](#), extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rddbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.366.5.83 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rddbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.366.5.84 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

- __fn* The function to add.
- __index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.366.5.85 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

- off* A file offset object.
- dir* The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.86 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (pos_type __pos) [inherited]`

Changing the current read position.

Parameters

- pos* A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.87 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp(pos_type __pos) [inherited]`

Changing the current write position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.88 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

off A file offset object.
dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() ->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.366.5.89 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.366.5.90 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is [ios_base::adjustfield](#).

Definition at line 594 of file ios_base.h.

5.366.5.91 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file basic_ios.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.366.5.92 `template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.366.5.93 `static bool std::ios_base::sync_with_stdio (bool __sync = true)`
[static, inherited]

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.366.5.94 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (void)` **[inherited]**

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::in`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.366.5.95 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp() [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.366.5.96 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.366.5.97 `template<typename _CharT, typename _Traits>
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (basic_ostream<_CharT, _Traits> * __tistr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tistr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

5.366.5.98 `template<typename _CharT, typename _Traits> basic_istream<
 _CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ungetc (void) [inherited]`

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

5.366.5.99 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

5.366.5.100 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::widen (char __c) const
[inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()) .widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file basic_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

5.366.5.101 `streamsize std::ios_base::width (streamsize __wide) [inline,
inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file ios_base.h.

5.366.5.102 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

5.366.5.103 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write (const char_type * __s, streamsize __n) [inherited]`

Character string insertion.

Parameters

s The array to insert.

n Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

Note

This function is not overloaded on signed `char` and unsigned `char`.

5.366.5.104 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.366.6 Member Data Documentation

5.366.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.366.6.2 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.366.6.3 `const openmode std::ios_base::app [static, inherited]`

Seek to end before each write.

Definition at line 363 of file `ios_base.h`.

5.366.6.4 const openmode std::ios_base::ate [static, inherited]

Open and seek to end immediately after opening.

Definition at line 366 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.366.6.5 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 333 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.366.6.6 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 312 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.366.6.7 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file ios_base.h.

5.366.6.8 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.366.6.9 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.366.6.10 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.366.6.11 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file ios_base.h.

5.366.6.12 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.366.6.13 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter

>::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.366.6.14 `const iostate std::ios_base::failbit` [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.366.6.15 `const fmtflags std::ios_base::fixed` [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.366.6.16 `const fmtflags std::ios_base::floatfield` [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 315 of file ios_base.h.

5.366.6.17 `const iostate std::ios_base::goodbit` [static, inherited]

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),

std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.366.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.366.6.19 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 374 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.366.6.20 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 271 of file ios_base.h.

5.366.6.21 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.366.6.22 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 278 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.366.6.23 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 377 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.366.6.24 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file ios_base.h.

5.366.6.25 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 285 of file ios_base.h.

5.366.6.26 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file ios_base.h.

5.366.6.27 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file ios_base.h.

5.366.6.28 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file ios_base.h.

5.366.6.29 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 299 of file ios_base.h.

5.366.6.30 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for ofstream.

Definition at line 380 of file ios_base.h.

5.366.6.31 const fmtflags std::ios_base::unitbuf [static, inherited]

Flushes output after each output operation.

Definition at line 302 of file ios_base.h.

5.366.6.32 const fmtflags std::ios_base::uppercase [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

The documentation for this class was generated from the following file:

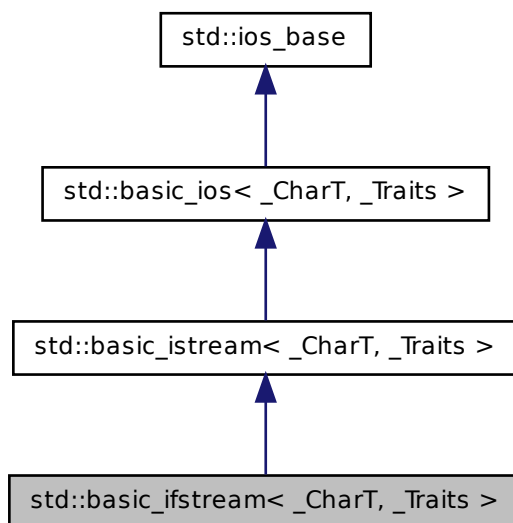
- [fstream](#)

5.367 std::basic_ifstream< _CharT, _Traits > Class Template Reference

Controlling input for files.

This class supports reading from named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ifstream< _CharT, _Traits >`:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `traits_type::off_type off_type`

- typedef int **open_mode**
- typedef `_Ios_Openmode` **openmode**
- typedef `traits_type::pos_type` **pos_type**
- typedef int **seek_dir**
- typedef `_Ios_Seekdir` **seekdir**
- typedef `std::streamoff` **streamoff**
- typedef `std::streampos` **streampos**
- typedef `_Traits` **traits_type**
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> __-`
`num_put_type`

Public Member Functions

- `basic_ifstream()`
- `basic_ifstream(const char * __s, ios_base::openmode __mode=ios_base::in)`
- `basic_ifstream(const std::string & __s, ios_base::openmode __mode=ios_base::in)`
- `~basic_ifstream()`
- `const locale & _M_getloc() const`
- `void _M_setstate(iostate __state)`
- `bool bad() const`
- `void clear(iostate __state=goodbit)`
- `void close()`
- `basic_ios & copyfmt(const basic_ios & __rhs)`
- `bool eof() const`
- `iostate exceptions() const`
- `void exceptions(iostate __except)`
- `bool fail() const`
- `char_type fill() const`
- `char_type fill(char_type __ch)`
- `fmtflags flags() const`
- `fmtflags flags(fmtflags __fmtfl)`
- `streamsize gcount() const`
- `template<>`
`basic_istream<char> & getline(char_type * __s, streamsize __n, char_type`
`__delim)`
- `template<>`
`basic_istream<wchar_t> & getline(char_type * __s, streamsize __n, char_type`
`__delim)`
- `locale getloc() const`
- `bool good() const`

- `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n)`
 - `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
 - `template<>`
`basic_istream< char > & ignore (streamsize __n)`
 - `template<>`
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
 - `locale imbue (const locale &__loc)`
 - `bool is_open ()`
 - `bool is_open () const`
 - `long & iword (int __ix)`
 - `char narrow (char_type __c, char __dfault) const`
 - `void open (const char *__s, ios_base::openmode __mode=ios_base::in)`
 - `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in)`
 - `streamsize precision () const`
 - `streamsize precision (streamsize __prec)`
 - `void *& pword (int __ix)`
 - `__filebuf_type * rdbuf () const`
 - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
 - `iosstate rdstate () const`
 - `void register_callback (event_callback __fn, int __index)`
 - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
 - `fmtflags setf (fmtflags __fmtfl)`
 - `void setstate (iosstate __state)`
 - `basic_ostream< _CharT, _Traits > * tie () const`
 - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
 - `void unsetf (fmtflags __mask)`
 - `char_type widen (char __c) const`
 - `streamsize width () const`
 - `streamsize width (streamsize __wide)`
-
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
 - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
 - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several

effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, [`ios_base::badbit`](#) will be turned on in the stream's error state without causing an [`ios_base::failure`](#) to be thrown. The original exception will then be rethrown.

- [`__istream_type & operator>> \(bool &__n\)`](#)
- [`__istream_type & operator>> \(short &__n\)`](#)
- [`__istream_type & operator>> \(unsigned short &__n\)`](#)
- [`__istream_type & operator>> \(int &__n\)`](#)
- [`__istream_type & operator>> \(unsigned int &__n\)`](#)
- [`__istream_type & operator>> \(long &__n\)`](#)
- [`__istream_type & operator>> \(unsigned long &__n\)`](#)
- [`__istream_type & operator>> \(long long &__n\)`](#)
- [`__istream_type & operator>> \(unsigned long long &__n\)`](#)
- [`__istream_type & operator>> \(float &__f\)`](#)
- [`__istream_type & operator>> \(double &__f\)`](#)
- [`__istream_type & operator>> \(long double &__f\)`](#)
- [`__istream_type & operator>> \(void *&__p\)`](#)
- [`__istream_type & operator>> \(__streambuf_type *__sb\)`](#)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type [`std::basic_ifstream::sentry`](#) with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by [`gcount\(\)`](#).

If an exception is thrown during extraction, [`ios_base::badbit`](#) will be turned on in the stream's error state without causing an [`ios_base::failure`](#) to be thrown. The original exception will then be rethrown.

- [`int_type get \(\)`](#)
- [`__istream_type & get \(char_type &__c\)`](#)
- [`__istream_type & get \(char_type *__s, streamsize __n, char_type __delim\)`](#)
- [`__istream_type & get \(char_type *__s, streamsize __n\)`](#)
- [`__istream_type & get \(__streambuf_type &__sb, char_type __delim\)`](#)
- [`__istream_type & get \(__streambuf_type &__sb\)`](#)
- [`__istream_type & getline \(char_type *__s, streamsize __n, char_type __delim\)`](#)
- [`__istream_type & getline \(char_type *__s, streamsize __n\)`](#)
- [`__istream_type & ignore \(\)`](#)
- [`__istream_type & ignore \(streamsize __n\)`](#)

- [__istream_type](#) & ignore (streamsize __n, int_type __delim)
- [int_type](#) peek ()
- [__istream_type](#) & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- [__istream_type](#) & putback (char_type __c)
- [__istream_type](#) & unget ()
- int sync ()
- [pos_type](#) tellg ()
- [__istream_type](#) & seekg (pos_type)
- [__istream_type](#) & seekg (off_type, ios_base::seekdir)
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) adjustfield
- static const [openmode](#) app
- static const [openmode](#) ate
- static const [iostate](#) badbit
- static const [fmtflags](#) basefield
- static const [seekdir](#) beg
- static const [openmode](#) binary
- static const [fmtflags](#) boolalpha
- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iostate](#) eofbit
- static const [iostate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out

- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- template<typename _ValueT >
 [__istream_type](#) & [_M_extract](#) (_ValueT &__v)
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- void [init](#) ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- [_Callback_list](#) * [_M_callbacks](#)
- const [__ctype_type](#) * [_M_ctype](#)
- [iostate](#) [_M_exception](#)
- [char_type](#) [_M_fill](#)
- bool [_M_fill_init](#)
- [fmtflags](#) [_M_flags](#)
- [streamsize](#) [_M_gcount](#)
- [locale](#) [_M_ios_locale](#)
- [_Words](#) [_M_local_word](#) [[_S_local_word_size](#)]
- const [__num_get_type](#) * [_M_num_get](#)
- const [__num_put_type](#) * [_M_num_put](#)
- [streamsize](#) [_M_precision](#)
- [basic_streambuf](#)< _CharT, _Traits > * [_M_streambuf](#)
- [iostate](#) [_M_streambuf_state](#)

- [basic_ostream](#)< _CharT, _Traits > * _M_tie
- [streamsize](#) _M_width
- _Words * _M_word
- int _M_word_size
- _Words _M_word_zero

Friends

- class `sentry`

5.367.1 Detailed Description

template<typename _CharT, typename _Traits> class std::basic_ifstream< _CharT, _Traits >

Controlling input for files.

This class supports reading from named files, using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Definition at line 418 of file `fstream`.

5.367.2 Member Typedef Documentation

5.367.2.1 **template<typename _CharT, typename _Traits> typedef**
c_type<_CharT> std::basic_istream< _CharT, _Traits
>::__c_type_type [inherited]

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file `istream`.

5.367.2.2 **template<typename _CharT, typename _Traits> typedef**
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_istream< _CharT, _Traits >::__num_get_type
[inherited]

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file `istream`.

5.367.2.3 `template<typename _CharT, typename _Traits> typedef
 num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
 > std::basic_ios<_CharT, _Traits>::__num_put_type
 [inherited]`

These are non-standard types.

Reimplemented in `std::basic_ostream<_CharT, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 84 of file `basic_ios.h`.

5.367.2.4 `template<typename _CharT, typename _Traits> typedef _CharT
 std::basic_ifstream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream<_CharT, _Traits>`.

Definition at line 422 of file `fstream`.

5.367.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
 [inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 435 of file `ios_base.h`.

5.367.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 254 of file ios_base.h.

5.367.2.7 `template<typename _CharT , typename _Traits > typedef
traits_type::int_type std::basic_ifstream< _CharT, _Traits
>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 424 of file fstream.

5.367.2.8 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file `ios_base.h`.

5.367.2.9 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_ifstream<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_istream<_CharT, _Traits>`.

Definition at line 426 of file `fstream`.

5.367.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 360 of file `ios_base.h`.

5.367.2.11 `template<typename _CharT , typename _Traits > typedef traits_type::pos_type std::basic_ifstream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 425 of file `fstream`.

5.367.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 392 of file `ios_base.h`.

5.367.2.13 `template<typename _CharT , typename _Traits > typedef _Traits std::basic_ifstream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 423 of file `fstream`.

5.367.3 Member Enumeration Documentation

5.367.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 418 of file `ios_base.h`.

5.367.4 Constructor & Destructor Documentation

5.367.4.1 `template<typename _CharT, typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream ()
[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 444 of file `fstream`.

5.367.4.2 `template<typename _CharT, typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (const char
* __s, ios_base::openmode __mode = ios_base::in) [inline,
explicit]`

Create an input file stream.

Parameters

s Null terminated string specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

[ios_base::in](#) is automatically included in *mode*.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 458 of file `fstream`.

5.367.4.3 `template<typename _CharT, typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (const
std::string & __s, ios_base::openmode __mode = ios_base::in)
[inline, explicit]`

Create an input file stream.

Parameters

s `std::string` specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

[ios_base::in](#) is automatically included in *mode*.

Definition at line 474 of file `fstream`.

5.367.4.4 `template<typename _CharT, typename _Traits >
std::basic_ifstream< _CharT, _Traits >::~~basic_ifstream ()
[inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 489 of file fstream.

5.367.5 Member Function Documentation

5.367.5.1 `const locale& std::ios_base::_M_getloc () const [inline, inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.367.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.367.5.3 `template<typename _CharT , typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.367.5.4 `template<typename _CharT , typename _Traits > void
std::basic_ifstream< _CharT, _Traits >::close () [inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 567 of file `fstream`.

5.367.5.5 `template<typename _CharT , typename _Traits > basic_ios<
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
const basic_ios< _CharT, _Traits > & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

5.367.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.367.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions() const [inline, inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.367.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions(iostate __except) [inline, inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.367.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits >::fail() const` [`inline`, `inherited`]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.367.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits >::fill(char_type __ch)` [`inline`, `inherited`]

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.367.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill () const [inline,
inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.367.5.12 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file `ios_base.h`.

5.367.5.13 fmtflags std::ios_base::flags () const [inline, inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.367.5.14 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline, inherited]

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.367.5.15 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.367.5.16 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (char_type & __c) [inherited]`

Simple extraction.

Parameters

c The character in which to store data.

Returns

*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.367.5.17 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n, char_type __delim) [inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

delim A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*-1 characters are stored

- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_ifstream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.367.5.18 `template<typename _CharT , typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::sputc()`.

5.367.5.19 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file istream.

5.367.5.20 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file istream.

5.367.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file istream.

Referenced by `std::basic_istream<char>::getline()`.

5.367.5.22 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n, char_type __delim) [inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored

3. $n-1$ characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sputc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.367.5.23 locale `std::ios_base::getloc () const` [inline, inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.367.5.24 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good () const` [inline, inherited]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file basic_ios.h.

5.367.5.25 `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::ignore(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_ifstream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.367.5.26 `template<typename _CharT, typename _Traits> basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::ignore(streamsize __n) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.367.5.27 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.367.5.28 `template<typename _CharT, typename _Traits > locale std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc) [inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

5.367 std::basic_ifstream< _CharT, _Traits > Class Template Reference 1647

Reimplemented from [std::ios_base](#).

Definition at line 113 of file basic_ios.tcc.

References [std::ios_base::getloc\(\)](#), and [std::basic_ios< _CharT, _Traits >::rdbuf\(\)](#).

Referenced by [std::operator<<\(\)](#).

5.367.5.29 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References [std::ios_base::badbit](#), and [std::ios_base::goodbit](#).

5.367.5.30 `template<typename _CharT , typename _Traits > bool
std::basic_ifstream< _CharT, _Traits >::is_open () [inline]`

Wrapper to test for an open file.

Returns

`rdbuf() -> is_open()`

Definition at line 508 of file fstream.

5.367.5.31 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.367.5.32 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

5.367.5.33 `template<typename _CharT , typename _Traits > void
std::basic_ifstream<_CharT, _Traits>::open (const std::string &
__s, ios_base::openmode __mode = ios_base::in) [inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s, mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 549 of file `fstream`.

5.367.5.34 `template<typename _CharT, typename _Traits> void
std::basic_ifstream<_CharT, _Traits>::open (const char * __s,
ios_base::openmode __mode = ios_base::in) [inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s,mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 529 of file `fstream`.

5.367.5.35 `template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits>::operator void * () const [inline,
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.367.5.36 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.367.5.37 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (__ios_type
&(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `istream` header.

Definition at line 124 of file `istream`.

5.367.5.38 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

5.367.5.39 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

5.367.5.40 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 131 of file `istream`.

5.367.5.41 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_ifstream<_CharT, _Traits>::operator>> (float & __f
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file `istream`.

5.367.5.42 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_ifstream<_CharT, _Traits>::operator>> (double &
__f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

5.367.5.43 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_ifstream<_CharT, _Traits>::operator>> (bool & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 167 of file istream.

```
5.367.5.44  template<typename _CharT, typename _Traits> __istream_type&
             std::basic_istream< _CharT, _Traits >::operator>> ( long double
             & __f ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 211 of file istream.

```
5.367.5.45  template<typename _CharT , typename _Traits > basic_istream<
             _CharT, _Traits > & std::basic_istream< _CharT, _Traits
             >::operator>> ( short & __n ) [inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

5.367.5.46 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_ifstream< _CharT, _Traits >::operator>> (void *& __p
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file istream.

5.367.5.47 `template<typename _CharT , typename _Traits > basic_ifstream<
_CharT, _Traits > & std::basic_ifstream< _CharT, _Traits
>::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.48 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

5.367.5.49 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file istream.

5.367.5.50 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.51 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

5.367.5.52 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(&)(__istream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iosmanip` header.

Definition at line 120 of file `istream`.

```
5.367.5.53  template<typename _CharT, typename _Traits> __istream_type&
             std::basic_istream< _CharT, _Traits >::operator>> ( long & __n
             ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file `istream`.

```
5.367.5.54  template<typename _CharT , typename _Traits > basic_istream<
             _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits
             >::peek ( void ) [inherited]
```

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

```
5.367.5.55  streamsize std::ios_base::precision ( streamsize __prec )
             [inline, inherited]
```

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.367.5.56 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

5.367.5.57 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (char_type __c) [inherited]`

Unextracting a single character.

Parameters

c The character to push back into the input stream.

Returns

*this

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sputbackc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sputbackc\(\)](#) fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

5.367.5.58 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.367.5.59 `template<typename _CharT, typename _Traits > __filebuf_type* std::basic_ifstream< _CharT, _Traits >::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Definition at line 500 of file `fstream`.

5.367.5.60 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.367.5.61 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate () const [inline, inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.367.5.62 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read (char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.367.5.63 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream<_CharT, _Traits>::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called *A* here:

- if *A* == -1, sets `eofbit` and extracts no characters
- if *A* == 0, extracts no characters

- if $A > 0$, extracts $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_ifstream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.367.5.64 void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]

Add the callback __fn with parameter __index.

Parameters

__fn The function to add.

__index The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.367.5.65 template<typename _CharT , typename _Traits > basic_ifstream< _CharT, _Traits > & std::basic_ifstream< _CharT, _Traits >::seekg (pos_type __pos) [inherited]

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.66 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff (off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.67 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by std::boolalpha(), std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

5.367.5.68 fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is [ios_base::adjustfield](#).

Definition at line 594 of file ios_base.h.

5.367.5.69 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate __state) [inline, inherited]

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(),

`std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.367.5.70 `template<typename _CharT, typename _Traits> int
std::basic_istream< _CharT, _Traits >::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.367.5.71 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.367.5.72 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg(void) [inherited]`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rddbuf()`.

5.367.5.73 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.367.5.74 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits
>::tie (basic_ostream<_CharT, _Traits> * __tistr) [inline,
inherited]`

Ties this stream to an output stream.

Parameters

tistr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

5.367.5.75 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
(void) [inherited]`

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

5.367.5.76 `void std::ios_base::unsetf (fmtflags __mask) [inline,
inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

5.367.5.77 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::widen (char __c) const
[inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file basic_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

5.367.5.78 `streamsize std::ios_base::width () const [inline,
inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits>::copyfmt(), std::num_put<_CharT, _OutIter>::do_put(), and std::operator>>().

5.367.5.79 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of [width\(\)](#).

Definition at line 652 of file `ios_base.h`.

5.367.5.80 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.367.6 Member Data Documentation

5.367.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.367.6.2 `const fmtflags std::ios_base::adjustfield` [static, inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.367.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 363 of file `ios_base.h`.

5.367.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 366 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.367.6.5 `const iostate std::ios_base::badbit` [static, inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 333 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.367.6.6 `const fmtflags std::ios_base::basefield` [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.367.6.7 `const seekdir std::ios_base::beg` **[static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 395 of file `ios_base.h`.

5.367.6.8 `const openmode std::ios_base::binary` **[static, inherited]**

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 371 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.367.6.9 `const fmtflags std::ios_base::boolalpha` **[static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.367.6.10 `const seekdir std::ios_base::cur` **[static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.367.6.11 `const fmtflags std::ios_base::dec` **[static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file `ios_base.h`.

5.367.6.12 `const seekdir std::ios_base::end` **[static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 401 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.367.6.13 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.367.6.14 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.367.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.367.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 315 of file ios_base.h.

5.367.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.367.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.367.6.19 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 374 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.367.6.20 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 271 of file ios_base.h.

5.367.6.21 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.367.6.22 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 278 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.367.6.23 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 377 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.367.6.24 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file ios_base.h.

5.367.6.25 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 285 of file ios_base.h.

5.367.6.26 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file ios_base.h.

5.367.6.27 `const fmtflags std::ios_base::showpoint` `[static, inherited]`

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file `ios_base.h`.

5.367.6.28 `const fmtflags std::ios_base::showpos` `[static, inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file `ios_base.h`.

5.367.6.29 `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 299 of file `ios_base.h`.

5.367.6.30 `const openmode std::ios_base::trunc` `[static, inherited]`

Open for input. Default for `ofstream`.

Definition at line 380 of file `ios_base.h`.

5.367.6.31 `const fmtflags std::ios_base::unitbuf` `[static, inherited]`

Flushes output after each output operation.

Definition at line 302 of file `ios_base.h`.

5.367.6.32 `const fmtflags std::ios_base::uppercase` `[static, inherited]`

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

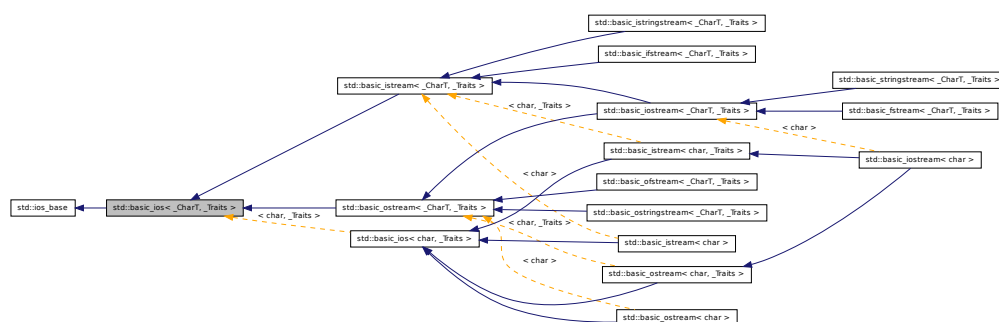
- [fstream](#)

5.368 std::basic_ios< _CharT, _Traits > Class Template Reference

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Inheritance diagram for `std::basic_ios< _CharT, _Traits >`:



Public Types

- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `int` `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `int` `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `int` `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
- typedef `_CharT` `char_type`
- typedef `_Traits::int_type` `int_type`
- typedef `_Traits::pos_type` `pos_type`
- typedef `_Traits::off_type` `off_type`
- typedef `_Traits` `traits_type`
- typedef `c_type< _CharT >` `__c_type_type`

- typedef [num_put](#)< [_CharT](#), [ostreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_put_type](#)
- typedef [num_get](#)< [_CharT](#), [istreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_get_type](#)

Public Member Functions

- [basic_ios](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > *[__sb](#))
- virtual [~basic_ios](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) [__state](#))
- bool [bad](#) () const
- void [clear](#) ([iostate](#) [__state](#)=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &[__rhs](#))
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) [__except](#))
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) [__ch](#))
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) [__fmtfl](#))
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &[__loc](#))
- long & [iword](#) (int [__ix](#))
- [char](#) [narrow](#) ([char_type](#) [__c](#), [char](#) [__dfault](#)) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) [__prec](#))
- void *& [pword](#) (int [__ix](#))
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > *[__sb](#))
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) () const
- [iostate](#) [rdstate](#) () const
- void [register_callback](#) ([event_callback](#) [__fn](#), int [__index](#))
- [fmtflags](#) [setf](#) ([fmtflags](#) [__fmtfl](#), [fmtflags](#) [__mask](#))
- [fmtflags](#) [setf](#) ([fmtflags](#) [__fmtfl](#))
- void [setstate](#) ([iostate](#) [__state](#))
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) () const
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) ([basic_ostream](#)< [_CharT](#), [_Traits](#) > *[__tistr](#))
- void [unsetf](#) ([fmtflags](#) [__mask](#))
- [char_type](#) [widen](#) ([char](#) [__c](#)) const

- [streamsize width](#) () const
- [streamsize width](#) (streamsize __wide)
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static [bool sync_with_stdio](#) (bool __sync=true)
- static [int xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags adjustfield](#)
- static const [openmode app](#)
- static const [openmode ate](#)
- static const [iostate badbit](#)
- static const [fmtflags basefield](#)
- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { `_S_local_word_size` }

Protected Member Functions

- `basic_ios` ()
- void `_M_cache_locale` (const `locale` &__loc)
- void `_M_call_callbacks` (`event` __ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int __index, bool __iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > *__sb)

Protected Attributes

- `_Callback_list` * `_M_callbacks`
- const `__ctype_type` * `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` * `_M_num_get`
- const `__num_put_type` * `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > * `_M_streambuf`
- `iostate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > * `_M_tie`
- `streamsize` `_M_width`
- `_Words` * `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

5.368.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ios< _CharT,
_Traits >
```

Virtual base class for all stream classes.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Definition at line 62 of file `basic_ios.h`.

5.368.2 Member Typedef Documentation

5.368.2.1 `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ios<_CharT, _Traits>::__ctype_type`

These are non-standard types.

Reimplemented in `std::basic_istream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 82 of file `basic_ios.h`.

5.368.2.2 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_get_type`

These are non-standard types.

Reimplemented in `std::basic_istream<_CharT, _Traits>`, `std::basic_istream<char, _Traits>`, and `std::basic_istream<char>`.

Definition at line 86 of file `basic_ios.h`.

5.368.2.3 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_put_type`

These are non-standard types.

Reimplemented in `std::basic_ostream<_CharT, _Traits>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 84 of file `basic_ios.h`.

5.368.2.4 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_ios<_CharT, _Traits>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istream< char, _Traits >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 71 of file `basic_ios.h`.

5.368.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)` **[inherited]**

The type of an event callback function.

Parameters

- event* One of the members of the event enum.
- ios_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 435 of file `ios_base.h`.

5.368.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`

- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 254 of file `ios_base.h`.

5.368.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ios<_CharT, _Traits>::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits>`, `std::basic_iostream<_CharT, _Traits>`, `std::basic_ostream<_CharT, _Traits>`, `std::basic_istreamstream<_CharT, _Traits, _Alloc>`, `std::basic_ostreamstream<_CharT, _Traits, _Alloc>`, `std::basic_stringstream<_CharT, _Traits, _Alloc>`, `std::basic_istream<char, _Traits>`, `std::basic_istream<char>`, `std::basic_iostream<char>`, `std::basic_ostream<char, _Traits>`, and `std::basic_ostream<char>`.

Definition at line 72 of file `basic_ios.h`.

5.368.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 329 of file ios_base.h.

5.368.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, `std::basic_istream< char, _Traits >`, `std::basic_istream< char >`, `std::basic_iostream< char >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 74 of file basic_ios.h.

5.368.2.10 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 360 of file ios_base.h.

5.368.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, `std::basic_istream< char, _Traits >`, `std::basic_istream< char >`, `std::basic_iostream< char >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 73 of file `basic_ios.h`.

5.368.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 392 of file `ios_base.h`.

5.368.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostream< _CharT, _Traits >`, `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, `std::basic_istream< char, _Traits >`, `std::basic_istream< char >`, `std::basic_iostream< char >`, `std::basic_ostream< char, _Traits >`, and `std::basic_ostream< char >`.

Definition at line 75 of file basic_ios.h.

5.368.3 Member Enumeration Documentation

5.368.3.1 enum std::ios_base::event [inherited]

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during [imbue\(\)](#). copyfmt_event is used during copyfmt().

Definition at line 418 of file ios_base.h.

5.368.4 Constructor & Destructor Documentation

5.368.4.1 template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::basic_ios (basic_streambuf< _CharT, _Traits > * __sb) [inline, explicit]

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 260 of file basic_ios.h.

5.368.4.2 template<typename _CharT, typename _Traits> virtual std::basic_ios< _CharT, _Traits >::~~basic_ios () [inline, virtual]

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by [rdbuf\(\)](#).

Definition at line 272 of file basic_ios.h.

5.368.4.3 template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::basic_ios () [inline, protected]

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 450 of file basic_ios.h.

5.368.5 Member Function Documentation

5.368.5.1 `const locale& std::ios_base::_M_getloc () const` [`inline`, `inherited`]

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.368.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad () const` [`inline`]

Fast error checking.

Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.368.5.3 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.368.5.4 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs)`

Copies fields of `__rhs` into this.

Parameters

`__rhs` The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.368.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

5.368.5.6 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::exceptions() const [inline]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.368.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::exceptions (iostate __except) [inline]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.368.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail () const [inline]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.368.5.9 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill () const [inline]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.368.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill (char_type __ch) [inline]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.368.5.11 fmtflags std::ios_base::flags () const [inline, inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.368.5.12 fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline, inherited]

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.368.5.13 locale std::ios_base::getloc () const [inline, inherited]

Locale access.

Returns

A copy of the current locale.

If imbue(loc) has previously been called, then this function returns loc. Otherwise, it returns a copy of std::locale(), the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Traits >::imbue(), std::operator>>(), and std::ws().

5.368.5.14 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.368.5.15 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.368.5.16 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.368.5.17 long& std::ios_base::iword (int __ix) [inline, inherited]

Access to integer array.

Parameters

__ix Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file ios_base.h.

**5.368.5.18 template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline]**

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of *char_type* to a character of *char*, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,dfault)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file basic_ios.h.

5.368.5.19 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * () const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.368.5.20 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! () const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.368.5.21 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

5.368.5.22 `streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file ios_base.h.

5.368.5.23 void*& std::ios_base::pword (int __ix) [inline, inherited]

Access to void pointer array.

Parameters

__ix Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.368.5.24 template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb)

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.368.5.25 `template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,
_Traits>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

5.368.5.26 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate () const [inline]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.368.5.27 void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]

Add the callback __fn with parameter __index.

Parameters

__fn The function to add.

__index The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.368.5.28 fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]

Setting new format flags.

Parameters

__fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by std::boolalpha(), std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

5.368.5.29 fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]

Setting new format flags.

Parameters

__fmtfl Additional flags to set.

__mask The flags mask for *__fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.368.5.30 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.368.5.31 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.368.5.32 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file basic_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.368.5.33 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tiestr) [inline]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file basic_ios.h.

5.368.5.34 `void std::ios_base::unsetf(fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

5.368.5.35 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::widen (char __c) const
[inline]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file basic_ios.h.

Referenced by std::endl(), std::getline(), and std::operator>>().

5.368.5.36 `streamsize std::ios_base::width () const [inline,
inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits>::copyfmt(), std::num_put<_CharT, _OutputIter>::do_put(), and std::operator>>().

5.368.5.37 streamsize std::ios_base::width (streamsize __wide) [inline, inherited]

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of [width\(\)](#).

Definition at line 652 of file ios_base.h.

5.368.5.38 static int std::ios_base::xalloc () throw () [static, inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.368.6 Member Data Documentation**5.368.6.1 const fmtflags std::ios_base::adjustfield [static, inherited]**

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 309 of file ios_base.h.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

5.368.6.2 const openmode std::ios_base::app [static, inherited]

Seek to end before each write.

Definition at line 363 of file ios_base.h.

5.368.6.3 const openmode std::ios_base::ate [static, inherited]

Open and seek to end immediately after opening.

Definition at line 366 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.368.6.4 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 333 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.368.6.5 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 312 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.368.6.6 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file ios_base.h.

5.368.6.7 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.368.6.8 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.368.6.9 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.368.6.10 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file ios_base.h.

5.368.6.11 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.368.6.12 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter

>::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.368.6.13 `const iostate std::ios_base::failbit` **[static, inherited]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.368.6.14 `const fmtflags std::ios_base::fixed` **[static, inherited]**

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.368.6.15 `const fmtflags std::ios_base::floatfield` **[static, inherited]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 315 of file ios_base.h.

5.368.6.16 `const iostate std::ios_base::goodbit` **[static, inherited]**

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),

`std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.368.6.17 `const fmtflags std::ios_base::hex` [**static, inherited**]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.368.6.18 `const openmode std::ios_base::in` [**static, inherited**]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.368.6.19 `const fmtflags std::ios_base::internal` [**static, inherited**]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 271 of file `ios_base.h`.

5.368.6.20 `const fmtflags std::ios_base::left` [**static, inherited**]

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.)

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.368.6.21 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 278 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.368.6.22 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 377 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.368.6.23 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file ios_base.h.

5.368.6.24 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 285 of file ios_base.h.

5.368.6.25 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file ios_base.h.

5.368.6.26 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file ios_base.h.

5.368.6.27 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file ios_base.h.

5.368.6.28 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 299 of file ios_base.h.

5.368.6.29 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for ostream.

Definition at line 380 of file ios_base.h.

5.368.6.30 const fmtflags std::ios_base::unitbuf [static, inherited]

Flushes output after each output operation.

Definition at line 302 of file ios_base.h.

5.368.6.31 const fmtflags std::ios_base::uppercase [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

The documentation for this class was generated from the following files:

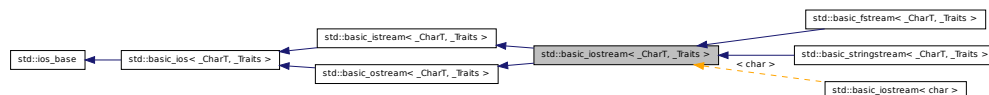
- [basic_ios.h](#)
- [basic_ios.tcc](#)

5.369 std::basic_iostream< _CharT, _Traits > Class Template Reference

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Inheritance diagram for `std::basic_istream<_CharT, _Traits>`:



Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<_CharT, _Traits>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_CharT` `char_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `_Traits::int_type` `int_type`
- typedef `_Traits::int_type` `int_type`
- typedef `int` `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `_Traits::off_type` `off_type`
- typedef `_Traits::off_type` `off_type`
- typedef `int` `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `_Traits::pos_type` `pos_type`
- typedef `_Traits::pos_type` `pos_type`
- typedef `int` `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`

- typedef `std::streampos` `streampos`
- typedef `_Traits traits_type`
- typedef `_Traits traits_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> __-
num_put_type`

Public Member Functions

- `basic_iostream` (`basic_streambuf<_CharT, _Traits> *__sb`)
- virtual `~basic_iostream` ()
- const `locale & _M_getloc` () const
- void `_M_setstate` (`iostate __state`)
- bool `bad` () const
- void `clear` (`iostate __state=goodbit`)
- `basic_ios & copyfmt` (const `basic_ios &__rhs`)
- bool `eof` () const
- `iostate exceptions` () const
- void `exceptions` (`iostate __except`)
- bool `fail` () const
- `char_type fill` () const
- `char_type fill` (`char_type __ch`)
- `fmtflags flags` (`fmtflags __fmtfl`)
- `fmtflags flags` () const
- `__ostream_type & flush` ()
- `streamsize gcount` () const
- template<>
`basic_istream< char> & getline` (`char_type *__s`, `streamsize __n`, `char_type
__delim`)
- template<>
`basic_istream< wchar_t> & getline` (`char_type *__s`, `streamsize __n`, `char_type
__delim`)
- `locale getloc` () const
- bool `good` () const
- template<>
`basic_istream< wchar_t> & ignore` (`streamsize __n`)
- template<>
`basic_istream< wchar_t> & ignore` (`streamsize __n`, `int_type __delim`)
- template<>
`basic_istream< char> & ignore` (`streamsize __n`)
- template<>
`basic_istream< char> & ignore` (`streamsize __n`, `int_type __delim`)
- `locale imbue` (const `locale &__loc`)

- `long & iword (int __ix)`
 - `char narrow (char_type __c, char __dfault) const`
 - `streamsize precision () const`
 - `streamsize precision (streamsize __prec)`
 - `void *& pword (int __ix)`
 - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
 - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
 - `iostate rdstate () const`
 - `void register_callback (event_callback __fn, int __index)`
 - `__ostream_type & seekp (pos_type)`
 - `__ostream_type & seekp (off_type, ios_base::seekdir)`
 - `fmtflags setf (fmtflags __fmtfl)`
 - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
 - `void setstate (iostate __state)`
 - `pos_type tellp ()`
 - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * _ - _tiestr)`
 - `basic_ostream< _CharT, _Traits > * tie () const`
 - `void unsetf (fmtflags __mask)`
 - `char_type widen (char __c) const`
 - `streamsize width (streamsize __wide)`
 - `streamsize width () const`
-
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
 - `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`
 - `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool & __n)`
- `__istream_type & operator>> (short & __n)`

- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type * __sb)`

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`
- `__istream_type & seekg (pos_type)`
- `__istream_type & seekg (off_type, ios_base::seekdir)`
- `operator void * () const`

- `bool operator! () const`
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

Static Public Member Functions

- static bool `sync_with_stdio` (bool __sync=true)
- static int `xalloc` () throw ()

Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `istate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `istate` `eofbit`
- static const `istate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `istate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

Protected Types

- enum { `_S_local_word_size` }

Protected Member Functions

- void **_M_cache_locale** (const [locale](#) &__loc)
- void **_M_call_callbacks** ([event](#) __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
 [__istream_type](#) & **_M_extract** (_ValueT &__v)
- [_Words](#) & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
 [__ostream_type](#) & **_M_insert** (_ValueT __v)
- void **init** ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- [_Callback_list](#) * **_M_callbacks**
- const [__ctype_type](#) * **_M_ctype**
- [iostate](#) **_M_exception**
- [char_type](#) **_M_fill**
- bool **_M_fill_init**
- [fmtflags](#) **_M_flags**
- [streamsize](#) **_M_gcount**
- [locale](#) **_M_ios_locale**
- [_Words](#) **_M_local_word** [[_S_local_word_size](#)]
- const [__num_get_type](#) * **_M_num_get**
- const [__num_put_type](#) * **_M_num_put**
- [streamsize](#) **_M_precision**
- [basic_streambuf](#)< _CharT, _Traits > * **_M_streambuf**
- [iostate](#) **_M_streambuf_state**
- [basic_ostream](#)< _CharT, _Traits > * **_M_tie**
- [streamsize](#) **_M_width**
- [_Words](#) * **_M_word**
- int **_M_word_size**
- [_Words](#) **_M_word_zero**

Friends

- class **sentry**
- class **sentry**

5.369.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_istream< _-
CharT, _Traits >
```

Merging istream and ostream capabilities.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Definition at line 769 of file istream.

5.369.2 Member Typedef Documentation

5.369.2.1 `template<typename _CharT, typename _Traits> typedef`
`ctype<_CharT> std::basic_istream< _CharT, _Traits`
`>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file istream.

5.369.2.2 `template<typename _CharT, typename _Traits> typedef`
`ctype<_CharT> std::basic_ostream< _CharT, _Traits`
`>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file ostream.

5.369.2.3 `template<typename _CharT, typename _Traits> typedef`
`num_get<_CharT, istreambuf_iterator<_CharT, _Traits>`
`> std::basic_istream< _CharT, _Traits >::__num_get_type`
`[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file istream.

5.369.2.4 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file ostream.

5.369.2.5 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ios<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_ostream<_CharT, _Traits>](#), [std::basic_ostream<char, _Traits>](#), and [std::basic_ostream<char>](#).

Definition at line 84 of file basic_ios.h.

5.369.2.6 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_istream<_CharT, _Traits>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc>](#).

Definition at line 59 of file istream.

5.369.2.7 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_iostream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Reimplemented in [std::basic_fstream<_CharT, _Traits>](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc>](#).

Definition at line 777 of file `istream`.

5.369.2.8 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)` `[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 435 of file `ios_base.h`.

5.369.2.9 `typedef _Ios_Fmtflags std::ios_base::fmtflags` `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`

- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 254 of file ios_base.h.

5.369.2.10 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_iostream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream< _CharT, _Traits >`.

Reimplemented in `std::basic_fstream< _CharT, _Traits >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 778 of file istream.

5.369.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_istream< _CharT, _Traits >::int_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istringstream< _CharT, _Traits, _Alloc >`.

Definition at line 60 of file istream.

5.369.2.12 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file `ios_base.h`.

5.369.2.13 `template<typename _CharT, typename _Traits> typedef
_Traits::off_type std::basic_iostream<_CharT, _Traits>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream<_CharT, _Traits>`.

Reimplemented in `std::basic_fstream<_CharT, _Traits>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 780 of file `istream`.

5.369.2.14 `template<typename _CharT, typename _Traits> typedef
_Traits::off_type std::basic_istream<_CharT, _Traits>::off_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, and `std::basic_istream<_CharT, _Traits, _Alloc>`.

Definition at line 62 of file `istream`.

5.369.2.15 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`

- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 360 of file `ios_base.h`.

5.369.2.16 `template<typename _CharT, typename _Traits> typedef
_Traits::pos_type std::basic_iostream< _CharT, _Traits
>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ostream< _CharT, _Traits >`.

Reimplemented in `std::basic_fstream< _CharT, _Traits >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 779 of file `istream`.

5.369.2.17 `template<typename _CharT, typename _Traits> typedef
_Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, and `std::basic_istringstream< _CharT, _Traits, _Alloc >`.

Definition at line 61 of file `istream`.

5.369.2.18 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`

- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 392 of file `ios_base.h`.

5.369.2.19 `template<typename _CharT, typename _Traits> typedef
_Traits std::basic_istream<_CharT, _Traits>::traits_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc>](#).

Definition at line 63 of file `istream`.

5.369.2.20 `template<typename _CharT, typename _Traits> typedef _Traits
std::basic_iostream<_CharT, _Traits>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Reimplemented in [std::basic_fstream<_CharT, _Traits>](#), and [std::basic_stringstream<_CharT, _Traits, _Alloc>](#).

Definition at line 781 of file `istream`.

5.369.3 Member Enumeration Documentation

5.369.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 418 of file `ios_base.h`.

5.369.4 Constructor & Destructor Documentation

5.369.4.1 `template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::basic_istream (
basic_streambuf< _CharT, _Traits > * __sb) [inline,
explicit]`

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

Definition at line 794 of file istream.

5.369.4.2 `template<typename _CharT, typename _Traits> virtual
std::basic_istream< _CharT, _Traits >::~~basic_istream ()
[inline, virtual]`

Destructor does nothing.

Definition at line 801 of file istream.

5.369.5 Member Function Documentation

5.369.5.1 `const locale& std::ios_base::_M_getloc () const [inline,
inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.369.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream< _CharT, _Traits >::_M_write (const char_type
* __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

5.369.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.369.5.4 `template<typename _CharT , typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.369.5.5 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

`__rhs` The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.369.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

5.369.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions (iostate __except) [inline, inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file basic_ios.h.

5.369.5.8 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits >::exceptions () const [inline, inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.369.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.369.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill(char_type __ch) [inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.369.5.11 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill() const [inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.369.5.12 fmtflags std::ios_base::flags() const [inline, inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.369.5.13 fmtflags std::ios_base::flags(fmtflags __fmtfl) [inline, inherited]

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.369.5.14 template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush() [inherited]

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.369.5.15 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream<_CharT, _Traits>::gcount () const
[inline, inherited]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file `istream`.

5.369.5.16 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits
>::get (void) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.17 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n, char_type __delim) [inherited]`

Simple multiple-character extraction.

Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.369.5.18 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type & __c) [inherited]`

Simple extraction.

Parameters

c The character in which to store data.

Returns

*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.369.5.19 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns get(s,n,widen('\n')).

Definition at line 333 of file istream.

5.369.5.20 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

5.369.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (__streambuf_type & __sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file istream.

5.369.5.22 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n, char_type __delim) [inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.369.5.23 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

5.369.5.24 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.369.5.25 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good () const [inline, inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.369.5.26 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.27 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore(streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

5.369 std::basic_istream< _CharT, _Traits > Class Template Reference 1733

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 555 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.369.5.28 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (streamsize __n) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.369.5.29 template<typename _CharT, typename _Traits > locale std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc) [inherited]

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls ios_base::imbue(loc), and if a stream buffer is associated with this stream, calls that buffer's pubimbue(loc).

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from [std::ios_base](#).

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.369.5.30 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<
_CharT, _Traits> * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.369.5.31 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.369.5.32 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

5.369.5.33 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * () const` [**inline**, **inherited**]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

5.369.5.34 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! () const` [**inline**, **inherited**]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

5.369.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (bool __n)` [**inline**, **inherited**]

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.369.5.36 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file ostream.

5.369.5.37 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.369.5.38 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `omanip` header.

Definition at line 127 of file `ostream`.

5.369.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

5.369.5.40 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

5.369.5.41 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (short __n) [inherited]`

Basic arithmetic inserters.

Parameters

`A` variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.369.5.42 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.369.5.43 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (int __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.369.5.44 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long long
__n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.369.5.45 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.369.5.46 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file `ostream`.

5.369.5.47 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file `ostream`.

5.369.5.48 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (const void *
__p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file `ostream`.

5.369.5.49 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) [inherited]`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.50 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 191 of file ostream.

5.369.5.51 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.369.5.52 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (void *& __p
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 215 of file istream.

5.369.5.53 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (
__istream_type &(*)(__istream_type &) __pf) [inline,
inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `omanip` header.

Definition at line 120 of file `istream`.

5.369.5.54 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned int & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

5.369.5.55 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.56 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file istream.

5.369.5.57 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file istream.

5.369.5.58 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 185 of file istream.

5.369.5.59 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (bool & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 167 of file istream.

5.369.5.60 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (unsigned
long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 189 of file istream.

5.369.5.61 `template<typename _CharT, typename _Traits > basic_istream<_CharT, _Traits > & std::basic_istream<_CharT, _Traits >::operator>> (short & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.62 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

5.369.5.63 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 131 of file istream.

5.369.5.64 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (double &
__f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file istream.

5.369.5.65 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (float & __f
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file istream.

5.369.5.66 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits
>::operator>> (int & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.67 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (long double
& __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 211 of file istream.

5.369.5.68 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (unsigned
short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

5.369 std::basic_istream< _CharT, _Traits > Class Template Reference 1749

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

5.369.5.69 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits
>::peek (void) [inherited]`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.70 `streamsize std::ios_base::precision (streamsize __prec)
[inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file ios_base.h.

5.369.5.71 `streamsize std::ios_base::precision () const [inline,
inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

5.369.5.72 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (char_type __c) [inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.369.5.73 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::putback (char_type __c) [inherited]`

Unextracting a single character.

Parameters

c The character to push back into the input stream.

Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

5.369.5.74 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.369.5.75 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::rdbuf () const [inline, inherited]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream<_CharT, _Traits>`, `std::basic_ofstream<_CharT, _Traits>`, `std::basic_fstream<_CharT, _Traits>`, `std::basic_istream<_CharT, _Traits, _Alloc>`, `std::basic_ostringstream<_CharT, _Traits, _Alloc>`, and `std::basic_stringstream<_CharT, _Traits, _Alloc>`.

Definition at line 311 of file `basic_ios.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::flush()`, `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::basic_istream<_CharT, _Traits>::operator>>()`, `std::operator>>()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_ostream<_CharT, _Traits>::put()`, `std::basic_istream<_CharT, _Traits>::putback()`, `std::basic_istream<_CharT, _Traits>::read()`, `std::basic_istream<_CharT, _Traits>::readsome()`, `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::sync()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, `std::basic_istream<_CharT, _Traits>::unget()`, and `std::ws()`.

5.369.5.76 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.369.5.77 `template<typename _CharT, typename _Traits> istate
std::basic_ios< _CharT, _Traits >::rdstate () const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.369.5.78 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.79 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.80 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

- __fn* The function to add.
- __index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.369.5.81 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (pos_type __pos) [inherited]`

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.82 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.83 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.369.5.84 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos) [inherited]`

Changing the current write position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.369.5.85 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file ios_base.h.

5.369.5.86 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.369.5.87 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.369.5.88 `template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

5.369 std::basic_iostream< _CharT, _Traits > Class Template Reference 1759

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.369.5.89 static bool std::ios_base::sync_with_stdio (bool __sync = true) [static, inherited]

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.369.5.90 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tell (void) [inherited]

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

5.369.5.91 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp() [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.369.5.92 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits>* __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

5.369.5.93 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.369.5.94 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::unget (void) [inherited]`

Unextracting the previous character.

Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, `std::basic_ios< _CharT, _Traits>::setstate()`, and `std::basic_streambuf< _CharT, _Traits>::sungetc()`.

5.369.5.95 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.369.5.96 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::widen (char __c) const
[inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()) .widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.369.5.97 `streamsize std::ios_base::width () const [inline,
inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

5.369.5.98 `streamsize std::ios_base::width (streamsize __wide) [inline,
inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.369.5.99 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write (const char_type * __s, streamsize __n) [inherited]`

Character string insertion.

Parameters

s The array to insert.

n Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.369.5.100 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.369.6 Member Data Documentation

5.369.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount` **[protected, inherited]**

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.369.6.2 `const fmtflags std::ios_base::adjustfield` **[static, inherited]**

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.369.6.3 `const openmode std::ios_base::app` **[static, inherited]**

Seek to end before each write.

Definition at line 363 of file `ios_base.h`.

5.369.6.4 `const openmode std::ios_base::ate` **[static, inherited]**

Open and seek to end immediately after opening.

Definition at line 366 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.369.6.5 `const iostate std::ios_base::badbit` **[static, inherited]**

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 333 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.369.6.6 `const fmtflags std::ios_base::basefield` [`static`, `inherited`]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.369.6.7 `const seekdir std::ios_base::beg` [`static`, `inherited`]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file `ios_base.h`.

5.369.6.8 `const openmode std::ios_base::binary` [`static`, `inherited`]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.369.6.9 `const fmtflags std::ios_base::boolalpha` [`static`, `inherited`]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.369.6.10 `const seekdir std::ios_base::cur` [`static`, `inherited`]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.369.6.11 `const fmtflags std::ios_base::dec` [`static`, `inherited`]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file `ios_base.h`.

5.369.6.12 `const seekdir std::ios_base::end` [`static`, `inherited`]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.369.6.13 `const iostate std::ios_base::eofbit` [`static`, `inherited`]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.

5.369.6.14 `const iostate std::ios_base::failbit` [`static`, `inherited`]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.369.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.369.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of set f.

Definition at line 315 of file ios_base.h.

5.369.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.369.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.369.6.19 `const openmode std::ios_base::in` **[static, inherited]**

Open for input. Default for `ifstream` and `fstream`.

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.369.6.20 `const fmtflags std::ios_base::internal` **[static, inherited]**

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 271 of file `ios_base.h`.

5.369.6.21 `const fmtflags std::ios_base::left` **[static, inherited]**

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.369.6.22 `const fmtflags std::ios_base::oct` **[static, inherited]**

Converts integer input or generates integer output in octal base.

Definition at line 278 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.369.6.23 `const openmode std::ios_base::out` **[static, inherited]**

Open for output. Default for `ofstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

5.369 std::basic_iostream< _CharT, _Traits > Class Template Reference 1769

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.369.6.24 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file `ios_base.h`.

5.369.6.25 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 285 of file `ios_base.h`.

5.369.6.26 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 289 of file `ios_base.h`.

5.369.6.27 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated `///` floating-point output.

Definition at line 293 of file `ios_base.h`.

5.369.6.28 const fmtflags std::ios_base::showpos [static, inherited]

Generates a `+` sign in non-negative generated numeric output.

Definition at line 296 of file `ios_base.h`.

5.369.6.29 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 299 of file `ios_base.h`.

5.369.6.30 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for `ofstream`.

Definition at line 380 of file ios_base.h.

5.369.6.31 `const fmtflags std::ios_base::unitbuf` `[static, inherited]`

Flushes output after each output operation.

Definition at line 302 of file ios_base.h.

5.369.6.32 `const fmtflags std::ios_base::uppercase` `[static, inherited]`

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file ios_base.h.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`.

The documentation for this class was generated from the following file:

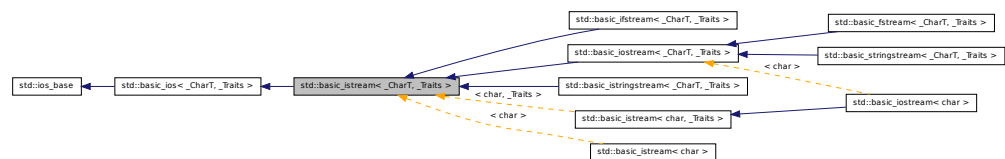
- [istream](#)

5.370 `std::basic_istream< _CharT, _Traits >` Class Template Reference

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic_streambuf](#) to do the actual input.

Inheritance diagram for `std::basic_istream< _CharT, _Traits >`:



Classes

- class [sentry](#)

Performs setup work for input streams.

Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `_CharT` `char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(* `event_callback`)(`event`, `ios_base` &, int)
- typedef `_Ios_Fmtflags` `fmtflags`
- typedef `_Traits::int_type` `int_type`
- typedef int `io_state`
- typedef `_Ios_Iostate` `iostate`
- typedef `_Traits::off_type` `off_type`
- typedef int `open_mode`
- typedef `_Ios_Openmode` `openmode`
- typedef `_Traits::pos_type` `pos_type`
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` `seekdir`
- typedef `std::streamoff` `streamoff`
- typedef `std::streampos` `streampos`
- typedef `_Traits` `traits_type`

- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

Public Member Functions

- `basic_istream` (`__streambuf_type` *__sb)
- virtual `~basic_istream` ()
- const `locale` & `_M_getloc` () const
- void `_M_setstate` (`iostate` __state)
- bool `bad` () const
- void `clear` (`iostate` __state=`goodbit`)
- `basic_ios` & `copyfmt` (const `basic_ios` &__rhs)
- bool `eof` () const
- `iostate` `exceptions` () const

- void [exceptions](#) (iostate __except)
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [fmtflags](#) [flags](#) () const
- [streamsize](#) [gcount](#) () const
- template<>
 [basic_istream](#)< [char](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- template<>
 [basic_istream](#)< [wchar_t](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>
 [basic_istream](#)< [wchar_t](#) > & [ignore](#) ([streamsize](#) __n)
- template<>
 [basic_istream](#)< [wchar_t](#) > & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- template<>
 [basic_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) __n)
- template<>
 [basic_istream](#)< [char](#) > & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- [locale](#) [imbue](#) (const [locale](#) &__loc)
- long & [iword](#) (int __ix)
- [char](#) [narrow](#) ([char_type](#) __c, [char](#) __dfault) const
- [streamsize](#) [precision](#) ([streamsize](#) __prec)
- [streamsize](#) [precision](#) () const
- void *& [pword](#) (int __ix)
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > * __sb)
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) () const
- [iostate](#) [rdstate](#) () const
- void [register_callback](#) ([event_callback](#) __fn, int __index)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl, [fmtflags](#) __mask)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl)
- void [setstate](#) ([iostate](#) __state)
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) () const
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) ([basic_ostream](#)< [_CharT](#), [_Traits](#) > * __tistr)
- void [unsetf](#) ([fmtflags](#) __mask)
- [char_type](#) [widen](#) ([char](#) __c) const
- [streamsize](#) [width](#) ([streamsize](#) __wide)

- `streamsize width () const`
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [int_type](#) [get](#) ()
- [__istream_type](#) & [get](#) ([char_type](#) &__c)
- [__istream_type](#) & [get](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [__istream_type](#) & [get](#) ([char_type](#) *__s, [streamsize](#) __n)
- [__istream_type](#) & [get](#) ([__streambuf_type](#) &__sb, [char_type](#) __delim)
- [__istream_type](#) & [get](#) ([__streambuf_type](#) &__sb)
- [__istream_type](#) & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [__istream_type](#) & [getline](#) ([char_type](#) *__s, [streamsize](#) __n)
- [__istream_type](#) & [ignore](#) ()
- [__istream_type](#) & [ignore](#) ([streamsize](#) __n)
- [__istream_type](#) & [ignore](#) ([streamsize](#) __n, [int_type](#) __delim)
- [int_type](#) [peek](#) ()
- [__istream_type](#) & [read](#) ([char_type](#) *__s, [streamsize](#) __n)
- [streamsize](#) [readsome](#) ([char_type](#) *__s, [streamsize](#) __n)
- [__istream_type](#) & [putback](#) ([char_type](#) __c)
- [__istream_type](#) & [unget](#) ()
- [int](#) [sync](#) ()
- [pos_type](#) [tellg](#) ()
- [__istream_type](#) & [seekg](#) ([pos_type](#))
- [__istream_type](#) & [seekg](#) ([off_type](#), [ios_base::seekdir](#))
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static [bool](#) [sync_with_stdio](#) ([bool](#) __sync=true)
- static [int](#) [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)

- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iostate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

Protected Types

- enum { `_S_local_word_size` }

Protected Member Functions

- void `_M_cache_locale` (const `locale` &__loc)
- void `_M_call_callbacks` (`event` __ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename _ValueT >
 `__istream_type` & `_M_extract` (_ValueT &__v)
- `_Words` & `_M_grow_words` (int __index, bool __iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > *__sb)

Protected Attributes

- `_Callback_list` * `_M_callbacks`
- const `__ctype_type` * `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`

- [fmtflags](#) [_M_flags](#)
- [streamsize](#) [_M_gcount](#)
- [locale](#) [_M_ios_locale](#)
- [_Words](#) [_M_local_word](#) [[_S_local_word_size](#)]
- [const](#) [__num_get_type](#) * [_M_num_get](#)
- [const](#) [__num_put_type](#) * [_M_num_put](#)
- [streamsize](#) [_M_precision](#)
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [_M_streambuf](#)
- [iostate](#) [_M_streambuf_state](#)
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [_M_tie](#)
- [streamsize](#) [_M_width](#)
- [_Words](#) * [_M_word](#)
- [int](#) [_M_word_size](#)
- [_Words](#) [_M_word_zero](#)

Friends

- [class](#) [sentry](#)

5.370.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_istream< _CharT, _Traits >`

Controlling input.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic_streambuf](#) to do the actual input.

Definition at line 55 of file istream.

5.370.2 Member Typedef Documentation

5.370.2.1 `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_istream< _CharT, _Traits >::__ctype_type`

These are non-standard types.

Reimplemented from [std::basic_ios](#)< [_CharT](#), [_Traits](#) >.

Definition at line 71 of file istream.

5.370.2.2 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits> >
std::basic_istream< _CharT, _Traits >::__num_get_type`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file `istream`.

5.370.2.3 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ios< _CharT, _Traits >::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_ostream< _CharT, _Traits >](#), [std::basic_ostream< char, _Traits >](#), and [std::basic_ostream< char >](#).

Definition at line 84 of file `basic_ios.h`.

5.370.2.4 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_istream< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 59 of file `istream`.

5.370.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios_base](#) and [basic_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 435 of file [ios_base.h](#).

5.370.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 254 of file [ios_base.h](#).

5.370.2.7 `template<typename _CharT, typename _Traits> typedef
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 60 of file `istream`.

5.370.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file `ios_base.h`.

5.370.2.9 `template<typename _CharT, typename _Traits> typedef
_Traits::off_type std::basic_istream< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 62 of file `istream`.

5.370.2.10 `typedef _Ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- *app*
- *ate*
- *binary*
- *in*
- *out*
- *trunc*

Definition at line 360 of file *ios_base.h*.

5.370.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istringstream< _CharT, _Traits, _Alloc >](#).

Definition at line 61 of file *istream*.

5.370.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

_Ios_Seekdir is implementation-defined. Defined values of type *seekdir* are:

- *beg*
- *cur*, equivalent to *SEEK_CUR* in the C standard library.
- *end*, equivalent to *SEEK_END* in the C standard library.

Definition at line 392 of file *ios_base.h*.

5.370.2.13 `template<typename _CharT, typename _Traits> typedef _Traits
std::basic_istream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 63 of file `istream`.

5.370.3 Member Enumeration Documentation**5.370.3.1** `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 418 of file `ios_base.h`.

5.370.4 Constructor & Destructor Documentation**5.370.4.1** `template<typename _CharT, typename _Traits> std::basic_istream<
_CharT, _Traits >::basic_istream (__streambuf_type * __sb)
[inline, explicit]`

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 91 of file `istream`.

5.370.4.2 `template<typename _CharT, typename _Traits> virtual
std::basic_istream< _CharT, _Traits >::~~basic_istream ()
[inline, virtual]`

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 101 of file `istream`.

5.370.5 Member Function Documentation

5.370.5.1 `const locale& std::ios_base::_M_getloc () const [inline, inherited]`

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.370.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.370.5.3 `template<typename _CharT , typename _Traits > void std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit) [inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file basic_ios.tcc.

Referenced by std::basic_ios< _CharT, _Traits >::rdbuf().

5.370.5.4 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of __rhs into this.

Parameters

`__rhs` The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 62 of file basic_ios.tcc.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.370.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic_ios.h.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, `std::basic_istream<_CharT, _Traits>::peek()`, `std::basic_istream<_CharT, _Traits>::putback()`, and `std::basic_istream<_CharT, _Traits>::unget()`.

5.370.5.6 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits >::exceptions (iostate __except) [inline, inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.370.5.7 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits >::exceptions () const [inline, inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits >::copyfmt()`.

5.370.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file basic_ios.h.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.370.5.9 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill(char_type __ch) [inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file basic_ios.h.

5.370.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill() const [inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.370.5.11 `fmtflags std::ios_base::flags() const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.370.5.12 `fmtflags std::ios_base::flags(fmtflags __fmtfl) [inline, inherited]`

Setting new format flags all at once.

Parameters

__fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 561 of file `ios_base.h`.

5.370.5.13 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::gcount() const [inline]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.370.5.14 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (void)`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.15 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type & __c)`

Simple extraction.

Parameters

`c` The character in which to store data.

Returns

`*this`

Tries to extract a character and store it in `c`. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.16 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n, char_type __delim)`

Simple multiple-character extraction.

Parameters

- s* Pointer to an array.
- n* Maximum number of characters to store in *s*.
- delim* A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.370.5.17 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (char_type * __s, streamsize __n) [inline]`

Simple multiple-character extraction.

Parameters

- s* Pointer to an array.
n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

5.370.5.18 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb, char_type __delim)`

Extraction into another streambuf.

Parameters

- sb* A streambuf in which to store data.
delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, `failbit` is set in the stream's error state.

Definition at line 356 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::putc()`.

5.370.5.19 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::get (__streambuf_type &
__sb) [inline]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb,widen('\n'))`.

Definition at line 366 of file `istream`.

5.370.5.20 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::getline (char_type * __s, streamsize __n, char_type __delim)`

String extraction.

Parameters

s A character array in which to store the data.

n Maximum number of characters to extract.

delim A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

5.370.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file istream.

Referenced by std::basic_istream< char >::getline().

5.370.5.22 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Traits >::imbue(), std::operator>>(), and std::ws().

5.370.5.23 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.370.5.24 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(void)`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `n != std::numeric_limits<int>::max()`, `n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sputc()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.25 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(streamsize __n)`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.370.5.26 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(streamsize __n, int_type __delim)`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_streambuf< _CharT, _Traits >::sbumpc()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.370.5.27 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.370.5.28 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<
_CharT, _Traits> * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.370.5.29 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.370.5.30 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

5.370.5.31 `template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits>::operator void * () const [inline,
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.370.5.32 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.370.5.33 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

5.370.5.34 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb)`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.35 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::operator>> (short & __n)`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.36 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (
__istream_type &(*)(__istream_type &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

5.370.5.37 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long & __n
) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

5.370.5.38 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long long &
__n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file istream.

5.370.5.39 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (__ios_type
&(*)(__ios_type &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file istream.

5.370.5.40 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (bool & __n
) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

5.370 `std::basic_istream<_CharT, _Traits>` Class Template Reference 1799

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

5.370.5.41 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned long & __n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

5.370.5.42 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (ios_base &(*)(ios_base &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

5.370.5.43 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (unsigned long long & __n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 198 of file istream.

5.370.5.44 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
short & __n) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 174 of file istream.

5.370.5.45 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (void *& __p
) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 215 of file istream.

5.370.5.46 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (float & __f
) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file istream.

5.370.5.47 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (double &
__f) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file istream.

5.370.5.48 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::operator>> (int & __n)`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get<_CharT, _InIter>::get()`, `std::ios_base::goodbit`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.370.5.49 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::operator>> (long double & __f) [inline]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file istream.

5.370.5.50 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek (void)`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.370.5.51 `streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.370.5.52 streamsize std::ios_base::precision () const [inline, inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

5.370.5.53 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (char_type __c)

Unextracting a single character.

Parameters

c The character to push back into the input stream.

Returns

*this

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sputbackc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sputbackc\(\)](#) fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

5.370.5.54 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.370.5.55 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf () const [inline, inherited]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in `std::basic_ifstream< _CharT, _Traits >`, `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_istream< _CharT, _Traits, _Alloc >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, and `std::basic_stringstream< _CharT, _Traits, _Alloc >`.

Definition at line 311 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::imbue(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

5.370.5.56 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > * __sb) [inherited]

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::clear().

5.370.5.57 `template<typename _CharT, typename _Traits> istate
std::basic_ios< _CharT, _Traits >::rdstate () const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See [std::ios_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

5.370.5.58 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
char_type * __s, streamsize __n)`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is [good\(\)](#), extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.59 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n)`

Extraction until the buffer is exhausted, but no more.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rddbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.60 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.370.5.61 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (pos_type __pos)`

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.370.5.62 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (off_type __off, ios_base::seekdir __dir)`

Changing the current read position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.63 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.370.5.64 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.370.5.65 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.370.5.66 `template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void)`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf< _CharT, _Traits >::pubsync()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.370.5.67 `static bool std::ios_base::sync_with_stdio (bool __sync = true)` **[static, inherited]**

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.370.5.68 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits>::pos_type std::basic_istream< _CharT, _Traits>::tellg (void)`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.370.5.69 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file basic_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.370.5.70 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file basic_ios.h.

5.370.5.71 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
(void)`

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sungetc()`.

5.370.5.72 void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.370.5.73 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char __c) const [inline, inherited]

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> > (getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.370.5.74 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.370.5.75 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

5.370.5.76 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.370.6 Member Data Documentation**5.370.6.1 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::_M_gcount` [protected]**

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.370.6.2 `const fmtflags std::ios_base::adjustfield` [static, inherited]

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.370.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 363 of file `ios_base.h`.

5.370.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

Definition at line 366 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.370.6.5 `const iostate std::ios_base::badbit` **[static, inherited]**

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 333 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.370.6.6 `const fmtflags std::ios_base::basefield` **[static, inherited]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.370.6.7 `const seekdir std::ios_base::beg` **[static, inherited]**

Request a seek relative to the beginning of the stream.

Definition at line 395 of file `ios_base.h`.

5.370.6.8 `const openmode std::ios_base::binary` **[static, inherited]**

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.370.6.9 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.370.6.10 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.370.6.11 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file `ios_base.h`.

5.370.6.12 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.370.6.13 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::ws()`.

5.370.6.14 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.370.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.370.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 315 of file ios_base.h.

5.370.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.370.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.370.6.19 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 374 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.370.6.20 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 271 of file ios_base.h.

5.370.6.21 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.370.6.22 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 278 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.370.6.23 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 377 of file ios_base.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.370.6.24 `const fmtflags std::ios_base::right` `[static, inherited]`

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file ios_base.h.

5.370.6.25 `const fmtflags std::ios_base::scientific` `[static, inherited]`

Generates floating-point output in scientific notation.

Definition at line 285 of file ios_base.h.

5.370.6.26 `const fmtflags std::ios_base::showbase` `[static, inherited]`

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file ios_base.h.

5.370.6.27 `const fmtflags std::ios_base::showpoint` `[static, inherited]`

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file ios_base.h.

5.370.6.28 `const fmtflags std::ios_base::showpos` `[static, inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file ios_base.h.

5.370.6.29 `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 299 of file ios_base.h.

5.370.6.30 `const openmode std::ios_base::trunc` [`static`, `inherited`]

Open for input. Default for `ofstream`.

Definition at line 380 of file `ios_base.h`.

5.370.6.31 `const fmtflags std::ios_base::unitbuf` [`static`, `inherited`]

Flushes output after each output operation.

Definition at line 302 of file `ios_base.h`.

5.370.6.32 `const fmtflags std::ios_base::uppercase` [`static`, `inherited`]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

5.371 `std::basic_istream< _CharT, _Traits >::sentry` Class Reference

Performs setup work for input streams.

Public Types

- `typedef __istream_type::__ctype_type __ctype_type`
- `typedef _Traits::int_type __int_type`
- `typedef basic_istream< _CharT, _Traits > __istream_type`
- `typedef basic_streambuf< _CharT, _Traits > __streambuf_type`
- `typedef _Traits traits_type`

Public Member Functions

- `sentry(basic_istream< _CharT, _Traits > &__is, bool __noskipws=false)`
- `operator bool() const`

5.371.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_istream< _-
_CharT, _Traits >::sentry
```

Performs setup work for input streams. Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

Definition at line 633 of file istream.

5.371.2 Member Typedef Documentation

5.371.2.1 `template<typename _CharT, typename _Traits> typedef _Traits
std::basic_istream< _CharT, _Traits >::sentry::traits_type`

Easy access to dependant types.

Definition at line 640 of file istream.

5.371.3 Constructor & Destructor Documentation

5.371.3.1 `template<typename _CharT, typename _Traits> std::basic_istream<
_CharT, _Traits >::sentry::sentry (basic_istream< _CharT, _Traits
> & __is, bool __noskipws = false) [explicit]`

The constructor performs all the work.

Parameters

is The input stream to guard.

noskipws Whether to consume whitespace or not.

If the stream state is good (*is.good()* is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if *noskipws* is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

5.372 `std::basic_istringstream< _CharT, _Traits, _Alloc >` Class Template Reference 1823

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file `istream.tcc`.

References `std::__ctype_abstract_base< _CharT >::is()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.371.4 Member Function Documentation

5.371.4.1 `template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::sentry::operator bool () const` [`inline`, `explicit`]

Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

Definition at line 681 of file `istream`.

The documentation for this class was generated from the following files:

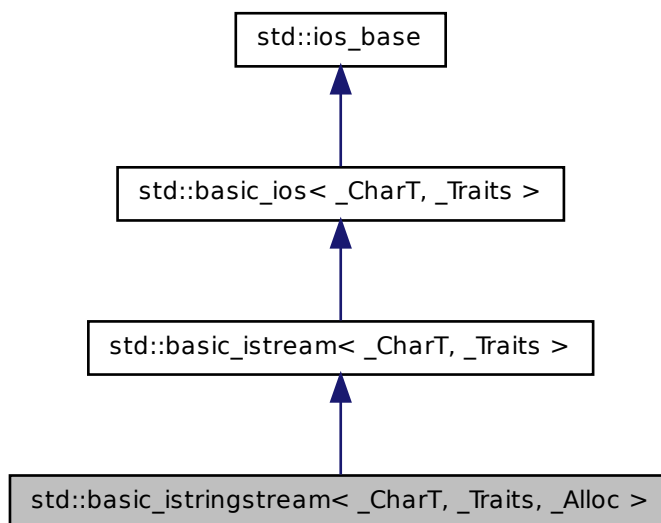
- [istream](#)
- [istream.tcc](#)

5.372 `std::basic_istringstream< _CharT, _Traits, _-Alloc >` Class Template Reference

Controlling input for `std::string`.

This class supports reading from objects of type [std::basic_string](#), using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_istream< _CharT, _Traits, _Alloc >`:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`

- typedef `_Ios_Iostate` [iostate](#)
 - typedef `traits_type::off_type` [off_type](#)
 - typedef `int` [open_mode](#)
 - typedef `_Ios_Openmode` [openmode](#)
 - typedef `traits_type::pos_type` [pos_type](#)
 - typedef `int` [seek_dir](#)
 - typedef `_Ios_Seekdir` [seekdir](#)
 - typedef [std::streamoff](#) [streamoff](#)
 - typedef [std::streampos](#) [streampos](#)
 - typedef `_Traits` [traits_type](#)
-
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __-`
`num_put_type`

Public Member Functions

- [basic_istringstream](#) ([ios_base::openmode](#) __mode=[ios_base::in](#))
- [basic_istringstream](#) (const [__string_type](#) &__str, [ios_base::openmode](#) __mode=[ios_base::in](#))
- [~basic_istringstream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [streamsize](#) [gcount](#) () const
- template<>
[basic_istream](#)< [char](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- template<>
[basic_istream](#)< [wchar_t](#) > & [getline](#) ([char_type](#) *__s, [streamsize](#) __n, [char_type](#) __delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const

- `template<>`
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
 - `template<>`
`basic_istream< char > & ignore (streamsize __n)`
 - `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n)`
 - `template<>`
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
 - `locale imbue (const locale &__loc)`
 - `long & iword (int __ix)`
 - `char narrow (char_type __c, char __dfault) const`
 - `streamsize precision () const`
 - `streamsize precision (streamsize __prec)`
 - `void *& pword (int __ix)`
 - `__stringbuf_type * rdbuf () const`
 - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
 - `iosstate rdstate () const`
 - `void register_callback (event_callback __fn, int __index)`
 - `fmtflags setf (fmtflags __fmtfl)`
 - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
 - `void setstate (iosstate __state)`
 - `void str (const __string_type &__s)`
 - `__string_type str () const`
 - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
 - `basic_ostream< _CharT, _Traits > * tie () const`
 - `void unsetf (fmtflags __mask)`
 - `char_type widen (char __c) const`
 - `streamsize width (streamsize __wide)`
 - `streamsize width () const`
-
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
 - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
 - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

Arithmetic Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`
- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`
- `__istream_type & operator>> (void *&__p)`
- `__istream_type & operator>> (__streambuf_type *__sb)`

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore ()`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `int_type peek ()`
- `__istream_type & read (char_type *__s, streamsize __n)`

- [streamsize](#) [readsome](#) ([char_type](#) *__s, [streamsize](#) __n)
- [__istream_type](#) & [putback](#) ([char_type](#) __c)
- [__istream_type](#) & [unget](#) ()
- [int](#) [sync](#) ()
- [pos_type](#) [tellg](#) ()
- [__istream_type](#) & [seekg](#) ([pos_type](#))
- [__istream_type](#) & [seekg](#) ([off_type](#), [ios_base::seekdir](#))
- [operator void *](#) () const
- [bool operator!](#) () const

Static Public Member Functions

- static [bool](#) [sync_with_stdio](#) ([bool](#) __sync=true)
- static [int](#) [xalloc](#) () [throw](#) ()

Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)

- static const `fmtflags showpoint`
- static const `fmtflags showpos`
- static const `fmtflags skipws`
- static const `openmode trunc`
- static const `fmtflags unitbuf`
- static const `fmtflags uppercase`

Protected Types

- enum { `_S_local_word_size` }

Protected Member Functions

- void `_M_cache_locale` (const `locale` &__loc)
- void `_M_call_callbacks` (`event` __ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- template<typename _ValueT >
 `__istream_type` & `_M_extract` (_ValueT &__v)
- `_Words` & `_M_grow_words` (int __index, bool __iword)
- void `_M_init` () throw ()
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > *__sb)

Protected Attributes

- `_Callback_list` * `_M_callbacks`
- const `__ctype_type` * `_M_ctype`
- `iostate` `_M_exception`
- `char_type` `_M_fill`
- bool `_M_fill_init`
- `fmtflags` `_M_flags`
- `streamsize` `_M_gcount`
- `locale` `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- const `__num_get_type` * `_M_num_get`
- const `__num_put_type` * `_M_num_put`
- `streamsize` `_M_precision`
- `basic_streambuf`< `_CharT`, `_Traits` > * `_M_streambuf`
- `iostate` `_M_streambuf_state`
- `basic_ostream`< `_CharT`, `_Traits` > * `_M_tie`
- `streamsize` `_M_width`
- `_Words` * `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

Friends

- class `sentry`

5.372.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_istream< _CharT, _Traits, _Alloc >`

Controlling input for `std::string`.

This class supports reading from objects of type [std::basic_string](#), using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as `sb`.

Definition at line 256 of file `sstream`.

5.372.2 Member Typedef Documentation

5.372.2.1 `template<typename _CharT, typename _Traits> typedef c_type<_CharT> std::basic_istream< _CharT, _Traits >::__c_type_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios](#)< _CharT, _Traits >.

Definition at line 71 of file `istream`.

5.372.2.2 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_istream< _CharT, _Traits >::__num_get_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios](#)< _CharT, _Traits >.

Definition at line 70 of file `istream`.

5.372.2.3 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_put_type [inherited]`

These are non-standard types.

5.372 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1831

Reimplemented in [std::basic_ostream<_CharT, _Traits>](#), [std::basic_ostream<char, _Traits>](#), and [std::basic_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

5.372.2.4 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _CharT std::basic_istream<_CharT, _Traits, _Alloc>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream<_CharT, _Traits>](#).

Definition at line 260 of file `sstream`.

5.372.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 435 of file `ios_base.h`.

5.372.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`

- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 254 of file ios_base.h.

5.372.2.7 `template<typename _CharT, typename _Traits, typename _Alloc>
 typedef traits_type::int_type std::basic_istream< _CharT,
 _Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 265 of file sstream.

5.372.2.8 `typedef _Ios_Iostate std::ios_base::iostate` **[inherited]**

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

5.372 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1833

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file `ios_base.h`.

5.372.2.9 `template<typename _CharT, typename _Traits, typename _Alloc>`
`typedef traits_type::off_type std::basic_istream<_CharT,`
`_Traits, _Alloc>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream<_CharT, _Traits>](#).

Definition at line 267 of file `sstream`.

5.372.2.10 `typedef _Ios_Openmode std::ios_base::openmode` `[inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 360 of file `ios_base.h`.

5.372.2.11 `template<typename _CharT, typename _Traits, typename _Alloc>
 typedef traits_type::pos_type std::basic_istream< _CharT,
 _Traits, _Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 266 of file sstream.

5.372.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 392 of file `ios_base.h`.

5.372.2.13 `template<typename _CharT, typename _Traits, typename _Alloc>
 typedef _Traits std::basic_istream< _CharT, _Traits, _Alloc
 >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 261 of file sstream.

5.372.3 Member Enumeration Documentation

5.372.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 418 of file `ios_base.h`.

5.372.4 Constructor & Destructor Documentation

5.372.4.1 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_istream< _CharT, _Traits, _Alloc
>::basic_istream (ios_base::openmode __mode = ios_base::in
) [inline, explicit]`

Default constructor starts with an empty string buffer.

Parameters

mode Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes *sb* using *mode*|*in*, and passes &*sb* to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 292 of file `sstream`.

5.372.4.2 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_istream< _CharT, _Traits, _Alloc
>::basic_istream (const __string_type & __str,
ios_base::openmode __mode = ios_base::in) [inline,
explicit]`

Starts with an existing string buffer.

Parameters

str A string to copy as a starting buffer.

mode Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes *sb* using *str* and *mode*|*in*, and passes &*sb* to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 310 of file `sstream`.

5.372.4.3 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_istream< _CharT, _Traits, _Alloc
>::~~basic_istream () [inline]`

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 321 of file sstream.

5.372.5 Member Function Documentation

5.372.5.1 `const locale& std::ios_base::_M_getloc () const [inline,
inherited]`

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::time_put< _CharT, _OutIter >::do_put()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::time_put< _CharT, _OutIter >::put()`.

5.372.5.2 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.372 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 1837

5.372.5.3 `template<typename _CharT, typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.372.5.4 `template<typename _CharT, typename _Traits > basic_ios<
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
const basic_ios< _CharT, _Traits > & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that [rdbuf\(\)](#) and [rdstate\(\)](#) remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

5.372.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::eof () const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.372.5.6 `template<typename _CharT, typename _Traits> iostate
std::basic_ios< _CharT, _Traits >::exceptions () const [inline,
inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.372.5.7 `template<typename _CharT, typename _Traits> void std::basic_ios<
_CharT, _Traits >::exceptions (iostate __except) [inline,
inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

5.372 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference

1839

```
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.372.5.8 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const` `[inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream<_CharT, _Traits>::seekg()`, `std::basic_ostream<_CharT, _Traits>::seekp()`, `std::basic_istream<_CharT, _Traits>::tellg()`, `std::basic_ostream<_CharT, _Traits>::tellp()`, and `std::regex_traits<_Ch_type>::value()`.

5.372.5.9 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill() const` `[inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.372.5.10 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.372.5.11 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.372.5.12 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

5.372 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1841

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.372.5.13 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::gcount () const
[inline, inherited]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.372.5.14 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits
>::get (void) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.372.5.15 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
char_type & __c) [inherited]`

Simple extraction.

Parameters

c The character in which to store data.

Returns

*this

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.372.5.16 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns get(s,n,widen('\n')).

Definition at line 333 of file istream.

5.372.5.17 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

5.372.5.18 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n, char_type __delim)
[inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

delim A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- $n-1$ characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.372.5.19 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb, widen('\n'))`.

Definition at line 366 of file istream.

5.372.5.20 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n, char_type __delim) [inherited]`

String extraction.

5.372 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference

1845

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.
- delim* A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sputc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.372.5.21 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

- s* A character array in which to store the data.
- n* Maximum number of characters to extract.

Returns

*this

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream< char >::getline()`.

5.372.5.22 locale `std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, `std::basic_ios< _CharT, _Traits >::imbue()`, `std::operator>>()`, and `std::ws()`.

5.372.5.23 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good () const [inline, inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.372.5.24 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::ignore (streamsize __n) [inherited]`

Simple extraction.

5.372 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference

1847

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.372.5.25 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::ignore(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if *n* != `std::numeric_limits<int>::max()`, *n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *delim* (in this case, the character is extracted); note that this condition will never occur if *delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.372.5.26 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.372.5.27 `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc) [inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.372 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference **1849**

5.372.5.28 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.372.5.29 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.372.5.30 `template<typename _CharT, typename _Traits> char
std::basic_ios< _CharT, _Traits >::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

5.372.5.31 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * () const` `[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 111 of file `basic_ios.h`.

5.372.5.32 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::operator! () const` `[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

5.372.5.33 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (short & __n)` `[inherited]`

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

5.372 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1851

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

5.372.5.34 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file istream.

5.372.5.35 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file istream.

5.372.5.36 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 174 of file istream.

```
5.372.5.37  template<typename _CharT, typename _Traits> __istream_type&
             std::basic_istream< _CharT, _Traits >::operator>> ( void *& __p
             ) [inline, inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

Definition at line 215 of file istream.

```
5.372.5.38  template<typename _CharT , typename _Traits > basic_istream<
             _CharT, _Traits > & std::basic_istream< _CharT, _Traits
             >::operator>> ( int & __n ) [inherited]
```

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to parse the input data.

5.372 `std::basic_istream< _CharT, _Traits, _Alloc >` Class Template Reference 1853

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.372.5.39 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.372.5.40 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file `istream`.

5.372.5.41 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(*)(__istream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 120 of file `istream`.

5.372.5.42 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 124 of file `istream`.

5.372.5.43 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 131 of file `istream`.

5.372 std::basic_istream<_CharT, _Traits, _Alloc > Class Template Reference 1855

5.372.5.44 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

5.372.5.45 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (unsigned long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file istream.

5.372.5.46 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits >::operator>> (double & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

5.372.5.47 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long long &
__n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

5.372.5.48 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (bool & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

`A` variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream`.

5.372 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1857

5.372.5.49 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file `istream`.

5.372.5.50 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (void) [inherited]`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.372.5.51 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

5.372.5.52 `streamsize std::ios_base::precision (streamsize __prec)
[inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.372.5.53 `template<typename _CharT, typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::putback (char_type __c) [inherited]`

Unextracting a single character.

Parameters

c The character to push back into the input stream.

Returns

*this

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sputbackc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sputbackc\(\)](#) fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::eof(), std::ios_base::goodbit, std::basic_ios<

5.372 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1859

`_CharT, _Traits >::rdbuf()`, `std::basic_ios<_CharT, _Traits >::setstate()`, and `std::basic_streambuf<_CharT, _Traits >::sputbackc()`.

Referenced by `std::operator>>()`.

5.372.5.54 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.372.5.55 `template<typename _CharT, typename _Traits, typename _Alloc> __stringbuf_type* std::basic_istream<_CharT, _Traits, _Alloc>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 332 of file `sstream`.

5.372.5.56 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.372.5.57 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate() const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.372.5.58 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> &std::basic_istream<_CharT, _Traits>::read(
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.372.5.59 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

s A character array.

n Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.372.5.60 `void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.372.5.61 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

`off` A file offset object.

`dir` The direction in which to seek.

Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

5.372 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1863

Definition at line 870 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.372.5.62 template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (pos_type __pos) [inherited]

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

5.372.5.63 fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.372.5.64 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.372.5.65 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate __state) [inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See `std::ios_base::iostate` for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream<`

5.372 std::basic_istream< _CharT, _Traits, _Alloc > Class Template Reference 1865

_CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

5.372.5.66 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_istream< _CharT, _Traits, _Alloc >::str (
const __string_type & __s) [inline]`

Setting a new buffer.

Parameters

s The string to use as a new sequence.

Calls `rdbuf() -> str(s)`.

Definition at line 350 of file sstream.

5.372.5.67 `template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_istream< _CharT, _Traits, _Alloc
>::str () const [inline]`

Copying out the string buffer.

Returns

`rdbuf() -> str()`

Definition at line 340 of file sstream.

5.372.5.68 `template<typename _CharT , typename _Traits > int
std::basic_istream< _CharT, _Traits >::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.372.5.69 `static bool std::ios_base::sync_with_stdio (bool __sync = true)`
[static, inherited]

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.372.5.70 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg (void)` **[inherited]**

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

5.372 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1867

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.372.5.71 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.372.5.72 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

5.372.5.73 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
(void) [inherited]`

Unextracting the previous character.

Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf() -> sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 744 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sungetc()`.

5.372.5.74 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.372.5.75 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen (char __c) const [inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

5.372 `std::basic_istream<_CharT, _Traits, _Alloc>` Class Template Reference 1869

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.372.5.76 `streamsize std::ios_base::width () const` `[inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _Outiter>::do_put()`, and `std::operator>>()`.

5.372.5.77 `streamsize std::ios_base::width (streamsize __wide)` `[inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.372.5.78 `static int std::ios_base::xalloc () throw ()` `[static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iwor`d and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iwor`d and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iwor`d and `pword` arrays.

5.372.6 Member Data Documentation
**5.372.6.1 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::_M_gcount` [protected,
inherited]**

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsomewhat()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.372.6.2 `const fmtflags std::ios_base::adjustfield` [static, inherited]

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.372.6.3 `const openmode std::ios_base::app` [static, inherited]

Seek to end before each write.

Definition at line 363 of file `ios_base.h`.

5.372.6.4 `const openmode std::ios_base::ate` [static, inherited]

Open and seek to end immediately after opening.

5.372 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference 1871

Definition at line 366 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.372.6.5 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 333 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.372.6.6 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 312 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.372.6.7 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file ios_base.h.

5.372.6.8 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file ios_base.h.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.372.6.9 `const fmtflags std::ios_base::boolalpha` **[static, inherited]**

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.372.6.10 `const seekdir std::ios_base::cur` **[static, inherited]**

Request a seek relative to the current position within the sequence.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.372.6.11 `const fmtflags std::ios_base::dec` **[static, inherited]**

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file `ios_base.h`.

5.372.6.12 `const seekdir std::ios_base::end` **[static, inherited]**

Request a seek relative to the current end of the sequence.

Definition at line 401 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.372.6.13 `const iostate std::ios_base::eofbit` **[static, inherited]**

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits`

5.372 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference 1873

>::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.372.6.14 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.372.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.372.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of set f.

Definition at line 315 of file ios_base.h.

5.372.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(),

`std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsomewhat()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.372.6.18 `const fmtflags std::ios_base::hex [static, inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.372.6.19 `const openmode std::ios_base::in [static, inherited]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.372.6.20 `const fmtflags std::ios_base::internal [static, inherited]`

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 271 of file `ios_base.h`.

5.372.6.21 `const fmtflags std::ios_base::left [static, inherited]`

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.372.6.22 `const fmtflags std::ios_base::oct [static, inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 278 of file `ios_base.h`.

5.372 `std::basic_istringstream< _CharT, _Traits, _Alloc >` Class Template Reference 1875

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.372.6.23 `const openmode std::ios_base::out` [`static`, `inherited`]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.372.6.24 `const fmtflags std::ios_base::right` [`static`, `inherited`]

Adds fill characters on the left (initial positions) of certain `///` generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file `ios_base.h`.

5.372.6.25 `const fmtflags std::ios_base::scientific` [`static`, `inherited`]

Generates floating-point output in scientific notation.

Definition at line 285 of file `ios_base.h`.

5.372.6.26 `const fmtflags std::ios_base::showbase` [`static`, `inherited`]

Generates a prefix indicating the numeric base of generated integer `///` output.

Definition at line 289 of file `ios_base.h`.

5.372.6.27 `const fmtflags std::ios_base::showpoint` [`static`, `inherited`]

Generates a decimal-point character unconditionally in generated `///` floating-point output.

Definition at line 293 of file `ios_base.h`.

5.372.6.28 `const fmtflags std::ios_base::showpos` [`static`, `inherited`]

Generates a `+` sign in non-negative generated numeric output.

Definition at line 296 of file `ios_base.h`.

5.372.6.29 `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 299 of file `ios_base.h`.

5.372.6.30 `const openmode std::ios_base::trunc` `[static, inherited]`

Open for input. Default for `ofstream`.

Definition at line 380 of file `ios_base.h`.

5.372.6.31 `const fmtflags std::ios_base::unitbuf` `[static, inherited]`

Flushes output after each output operation.

Definition at line 302 of file `ios_base.h`.

5.372.6.32 `const fmtflags std::ios_base::uppercase` `[static, inherited]`

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

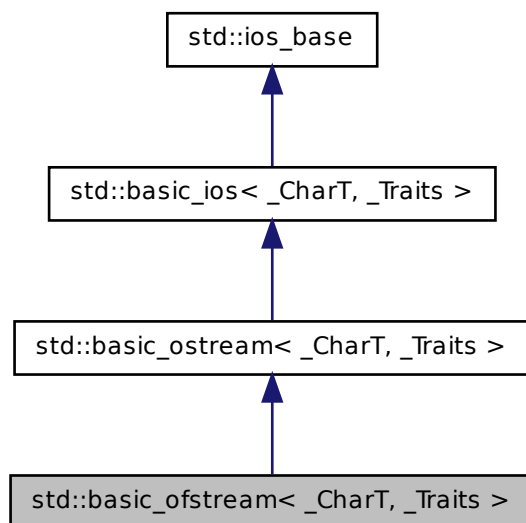
- [sstream](#)

5.373 `std::basic_ofstream<_CharT, _Traits>` Class Template Reference

Controlling output for files.

This class supports reading from named files, using the inherited functions from [std::basic_ostream](#). To control the associated sequence, an instance of [std::basic_filebuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for std::basic_ofstream< _CharT, _Traits >:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< char_type, traits_type > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `traits_type::off_type off_type`

- typedef int **open_mode**
 - typedef _Ios_Openmode [openmode](#)
 - typedef traits_type::pos_type [pos_type](#)
 - typedef int **seek_dir**
 - typedef _Ios_Seekdir [seekdir](#)
 - typedef [std::streamoff](#) **streamoff**
 - typedef [std::streampos](#) **streampos**
 - typedef _Traits [traits_type](#)
-
- typedef [num_get](#)< _CharT, [istreambuf_iterator](#)< _CharT, _Traits > > [__num_get_type](#)

Public Member Functions

- [basic_ofstream](#) ()
- [basic_ofstream](#) (const char *__s, [ios_base::openmode](#) __mode=[ios_base::out|ios_base::trunc](#))
- [basic_ofstream](#) (const [std::string](#) &__s, [ios_base::openmode](#) __mode=[ios_base::out|ios_base::trunc](#))
- [~basic_ofstream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- void [close](#) ()
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [__ostream_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &__loc)
- bool [is_open](#) ()
- bool [is_open](#) () const
- long & [iword](#) (int __ix)
- char [narrow](#) ([char_type](#) __c, char __dfault) const

- `void open (const char *__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)`
- `void open (const std::string &__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)`
- `streamsize precision () const`
- `streamsize precision (streamsize __prec)`
- `void *& pword (int __ix)`
- `__filebuf_type * rdbuf () const`
- `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> *__sb)`
- `iosstate rdstate () const`
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `__ostream_type & seekp (pos_type)`
- `fmtflags setf (fmtflags __fmtfl)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `void setstate (iosstate __state)`
- `pos_type tellp ()`
- `basic_ostream<_CharT, _Traits> * tie () const`
- `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> * __tistr)`
- `void unsetf (fmtflags __mask)`
- `char_type widen (char __c) const`
- `streamsize width (streamsize __wide)`
- `streamsize width () const`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`

- [__ostream_type](#) & [operator<<](#) (unsigned long __n)
- [__ostream_type](#) & [operator<<](#) (bool __n)
- [__ostream_type](#) & [operator<<](#) (short __n)
- [__ostream_type](#) & [operator<<](#) (unsigned short __n)
- [__ostream_type](#) & [operator<<](#) (int __n)
- [__ostream_type](#) & [operator<<](#) (unsigned int __n)
- [__ostream_type](#) & [operator<<](#) (long long __n)
- [__ostream_type](#) & [operator<<](#) (unsigned long long __n)
- [__ostream_type](#) & [operator<<](#) (double __f)
- [__ostream_type](#) & [operator<<](#) (float __f)
- [__ostream_type](#) & [operator<<](#) (long double __f)
- [__ostream_type](#) & [operator<<](#) (const void *__p)
- [__ostream_type](#) & [operator<<](#) ([__streambuf_type](#) *__sb)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type [std::basic_ostream::sentry](#). This has several effects, concluding with the setting of a status flag; see the [sentry documentation](#) for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, [ios_base::badbit](#) will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [__ostream_type](#) & [put](#) ([char_type](#) __c)
- void [_M_write](#) (const [char_type](#) *__s, [streamsize](#) __n)
- [__ostream_type](#) & [write](#) (const [char_type](#) *__s, [streamsize](#) __n)
- [operator void *](#) () const
- bool [operator!](#) () const

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)

- static const [seekdir beg](#)
- static const [openmode binary](#)
- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- template<typename _ValueT >
 [__ostream_type](#) & [_M_insert](#) (_ValueT __v)
- void [init](#) ([basic_streambuf](#)<_CharT, _Traits> *__sb)

Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

Friends

- `class sentry`

5.373.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ofstream< _CharT, _Traits >`

Controlling output for files.

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 586 of file `fstream`.

5.373.2 Member Typedef Documentation

5.373.2.1 `template<typename _CharT, typename _Traits> typedef
 ctype<_CharT> std::basic_ofstream<_CharT, _Traits
 >::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file `ostream`.

5.373.2.2 `template<typename _CharT, typename _Traits> typedef
 num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
 > std::basic_ios<_CharT, _Traits>::__num_get_type
 [inherited]`

These are non-standard types.

Reimplemented in [std::basic_istream<_CharT, _Traits>](#), [std::basic_istream<char, _Traits>](#), and [std::basic_istream<char>](#).

Definition at line 86 of file `basic_ios.h`.

5.373.2.3 `template<typename _CharT, typename _Traits> typedef
 num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
 > std::basic_ofstream<_CharT, _Traits>::__num_put_type
 [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file `ostream`.

5.373.2.4 `template<typename _CharT, typename _Traits> typedef _CharT
 std::basic_ofstream<_CharT, _Traits>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ofstream<_CharT, _Traits>](#).

Definition at line 590 of file `fstream`.

5.373.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)` [*inherited*]

The type of an event callback function.

Parameters

- event* One of the members of the event enum.
- ios_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 435 of file `ios_base.h`.

5.373.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [*inherited*]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 254 of file `ios_base.h`.

5.373.2.7 `template<typename _CharT , typename _Traits > typedef traits_type::int_type std::basic_ofstream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Definition at line 592 of file `fstream`.

5.373.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file `ios_base.h`.

5.373.2.9 `template<typename _CharT , typename _Traits > typedef traits_type::off_type std::basic_ofstream< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Definition at line 594 of file `fstream`.

5.373.2.10 `typedef _Ios_Openmode std::ios_base::openmode` **[inherited]**

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 360 of file `ios_base.h`.

5.373.2.11 `template<typename _CharT , typename _Traits > typedef traits_type::pos_type std::basic_ofstream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Definition at line 593 of file `fstream`.

5.373.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir` **[inherited]**

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 392 of file ios_base.h.

5.373.2.13 `template<typename _CharT , typename _Traits > typedef _Traits
std::basic_ofstream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Definition at line 591 of file fstream.

5.373.3 Member Enumeration Documentation

5.373.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during [imbue\(\)](#). copyfmt_event is used during copyfmt().

Definition at line 418 of file ios_base.h.

5.373.4 Constructor & Destructor Documentation

5.373.4.1 `template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream ()
[inline]`

Default constructor.

Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 612 of file fstream.

5.373.4.2 `template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (const char
* __s, ios_base::openmode __mode = ios_base::out|ios_base::trunc)
[inline, explicit]`

Create an output file stream.

Parameters

s Null terminated string specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

[ios_base::out](#)|[ios_base::trunc](#) is automatically included in *mode*.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 627 of file `fstream`.

```
5.373.4.3 template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream
( const std::string & __s, ios_base::openmode __mode =
ios_base::out|ios_base::trunc ) [inline, explicit]
```

Create an output file stream.

Parameters

s `std::string` specifying the filename.

mode Open file in specified mode (see [std::ios_base](#)).

[ios_base::out](#)|[ios_base::trunc](#) is automatically included in *mode*.

Definition at line 645 of file `fstream`.

```
5.373.4.4 template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream ( )
[inline]
```

The destructor does nothing.

The file is closed by the `filebuf` object, not the formatting stream.

Definition at line 660 of file `fstream`.

5.373.5 Member Function Documentation

```
5.373.5.1 const locale& std::ios_base::_M_getloc ( ) const [inline,
inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

5.373.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream< _CharT, _Traits >::M_write (const char_type
* __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

5.373.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.373.5.4 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.373.5.5 `template<typename _CharT, typename _Traits> void
std::basic_ofstream<_CharT, _Traits>::close () [inline]`

Close the file.

Calls [std::basic_filebuf::close\(\)](#). If that function fails, `failbit` is set in the stream's error state.

Definition at line 740 of file `fstream`.

5.373.5.6 `template<typename _CharT, typename _Traits> basic_ios<
_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that [rdbuf\(\)](#) and [rdstate\(\)](#) remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

5.373 std::basic_ofstream< _CharT, _Traits > Class Template Reference 1891

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), and std::ios_base::width().

5.373.5.7 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof() const [inline, inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic_ios.h.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), and std::basic_istream< _CharT, _Traits >::unget().

5.373.5.8 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions() const [inline, inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.373.5.9 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions(iostate __except) [inline, inherited]

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 247 of file `basic_ios.h`.

5.373.5.10 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::fail () const [inline,
inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.373.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file basic_ios.h.

5.373.5.12 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill () const [inline,
inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.373.5.13 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.373.5.14 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.373.5.15 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.373.5.16 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.373.5.17 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 170 of file basic_ios.h.

5.373.5.18 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from [std::ios_base](#).

Definition at line 113 of file basic_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.373.5.19 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.373.5.20 `template<typename _CharT, typename _Traits> bool
std::basic_ofstream<_CharT, _Traits>::is_open () [inline]`

Wrapper to test for an open file.

Returns

`rdbuf() -> is_open()`

Definition at line 679 of file fstream.

5.373.5.21 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.373.5.22 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

5.373.5.23 `template<typename _CharT, typename _Traits> void
std::basic_ofstream<_CharT, _Traits>::open (const char * __s,
ios_base::openmode __mode = ios_base::out | ios_base::trunc)
[inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using `std::string` to hold the filename, you must use `.c_str()` before passing it to this constructor.

Definition at line 700 of file `fstream`.

5.373.5.24 `template<typename _CharT, typename _Traits> void
std::basic_ofstream<_CharT, _Traits>::open (const std::string &
__s, ios_base::openmode __mode = ios_base::out | ios_base::trunc
) [inline]`

Opens an external file.

Parameters

s The name of the file.

mode The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out|trunc)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 721 of file `fstream`.

5.373.5.25 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.373.5.26 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.373.5.27 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

5.373.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

5.373 std::basic_ofstream< _CharT, _Traits > Class Template Reference 1899

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

```
5.373.5.29  template<typename _CharT, typename _Traits > basic_ofstream<
               _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits
               >::operator<<( __streambuf_type * __sb ) [inherited]
```

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

```
5.373.5.30  template<typename _CharT, typename _Traits> __ostream_type&
               std::basic_ofstream< _CharT, _Traits >::operator<<( ios_base
               &(*)(ios_base &) __pf ) [inline, inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file ostream.

5.373.5.31 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.373.5.32 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.373.5.33 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.373.5.34 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ofstream< _CharT, _Traits >::operator<< (long __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.373.5.35 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ofstream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.373.5.36 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (const void *
__p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.373.5.37 `template<typename _CharT , typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits
>::operator<< (int __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.373.5.38 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file ostream.

5.373.5.39 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ofstream< _CharT, _Traits >::operator<< (bool __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.373.5.40 `template<typename _CharT, typename _Traits> basic_ofstream<
_CharT, _Traits> & std::basic_ofstream< _CharT, _Traits
>::operator<< (short __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.373.5.41 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.373.5.42 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.373.5.43 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline, inherited]`

Interface for manipulators.

5.373 std::basic_ofstream<_CharT, _Traits> Class Template Reference 1905

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

5.373.5.44 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

5.373.5.45 `streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

5.373.5.46 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::put (char_type __c) [inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.373.5.47 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.373.5.48 `template<typename _CharT, typename _Traits> __filebuf_type* std::basic_ofstream<_CharT, _Traits>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 671 of file fstream.

5.373.5.49 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.373.5.50 `template<typename _CharT, typename _Traits> iostate std::basic_ios<_CharT, _Traits>::rdstate () const [inline, inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.373.5.51 `void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

- __fn* The function to add.
- __index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.373.5.52 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

- off* A file offset object.
- dir* The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.373.5.53 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos) [inherited]`

Changing the current write position.

Parameters

- pos* A file position object.

Returns

*this

5.373 std::basic_ofstream< _CharT, _Traits > Class Template Reference 1909

If `fail()` is not true, calls `rdbuf() ->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.373.5.54 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file ios_base.h.

5.373.5.55 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.373.5.56 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.373.5.57 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.373.5.58 `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>::pos_type std::basic_ofstream<_CharT, _Traits>::tellp() [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.373.5.59 `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ofstream<_CharT, _Traits>* __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

5.373.5.60 `template<typename _CharT, typename _Traits> basic_ofstream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.373.5.61 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.373.5.62 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen (char __c) const [inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.373.5.63 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.373.5.64 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

5.373.5.65 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ofstream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inherited]`

Character string insertion.

Parameters

s The array to insert.

n Maximum number of characters to insert.

Returns

`*this`

Characters are copied from *s* and inserted into the stream until one of the following happens:

- n characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.373.5.66 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.373.6 Member Data Documentation

5.373.6.1 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.373.6.2 `const openmode std::ios_base::app [static, inherited]`

Seek to end before each write.

Definition at line 363 of file `ios_base.h`.

5.373.6.3 const openmode std::ios_base::ate [static, inherited]

Open and seek to end immediately after opening.

Definition at line 366 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.373.6.4 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 333 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.373.6.5 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 312 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.373.6.6 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file ios_base.h.

5.373.6.7 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.373.6.8 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract bool in alphabetic rather than numeric format.

Definition at line 257 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.373.6.9 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.373.6.10 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file ios_base.h.

5.373.6.11 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.373.6.12 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter

>::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.373.6.13 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.373.6.14 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.373.6.15 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 315 of file ios_base.h.

5.373.6.16 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),

`std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.373.6.17 `const fmtflags std::ios_base::hex` [**static, inherited**]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.373.6.18 `const openmode std::ios_base::in` [**static, inherited**]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.373.6.19 `const fmtflags std::ios_base::internal` [**static, inherited**]

Adds fill characters at a designated internal point in certain `///` generated output, or identical to `right` if no such point is `///` designated.

Definition at line 271 of file `ios_base.h`.

5.373.6.20 `const fmtflags std::ios_base::left` [**static, inherited**]

Adds fill characters on the right (final positions) of certain `///` generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.373.6.21 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 278 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.373.6.22 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 377 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.373.6.23 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file ios_base.h.

5.373.6.24 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 285 of file ios_base.h.

5.373.6.25 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file ios_base.h.

5.373.6.26 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file ios_base.h.

5.373.6.27 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file `ios_base.h`.

5.373.6.28 `const fmtflags std::ios_base::skipws` `[static, inherited]`

Skips leading white space before certain input operations.

Definition at line 299 of file `ios_base.h`.

5.373.6.29 `const openmode std::ios_base::trunc` `[static, inherited]`

Open for input. Default for `ofstream`.

Definition at line 380 of file `ios_base.h`.

5.373.6.30 `const fmtflags std::ios_base::unitbuf` `[static, inherited]`

Flushes output after each output operation.

Definition at line 302 of file `ios_base.h`.

5.373.6.31 `const fmtflags std::ios_base::uppercase` `[static, inherited]`

Replaces certain lowercase letters with their uppercase equivalents `///` in generated output.

Definition at line 306 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

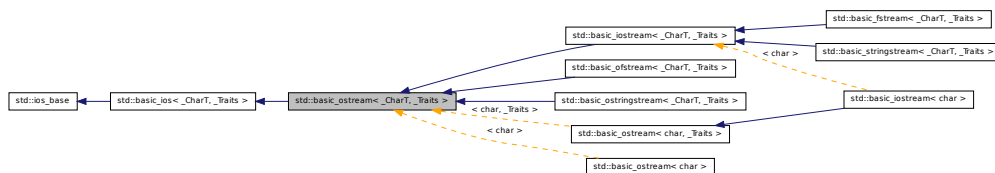
- [fstream](#)

5.374 `std::basic_ostream<_CharT, _Traits>` Class Template Reference

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic_streambuf](#) to do the actual output.

Inheritance diagram for std::basic_ostream< _CharT, _Traits >:



Classes

- class `sentry`
Performs setup work for output streams.

Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `_Traits::int_type int_type`
- typedef `int io_state`
- typedef `_Ios_Iostate iostate`
- typedef `_Traits::off_type off_type`
- typedef `int open_mode`
- typedef `_Ios_Openmode openmode`
- typedef `_Traits::pos_type pos_type`
- typedef `int seek_dir`
- typedef `_Ios_Seekdir seekdir`
- typedef `std::streamoff streamoff`
- typedef `std::streampos streampos`
- typedef `_Traits traits_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`

Public Member Functions

- [basic_ostream](#) ([__streambuf_type](#) *__sb)
- virtual [~basic_ostream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- void [exceptions](#) ([iostate](#) __except)
- [iostate](#) [exceptions](#) () const
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [__ostream_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &__loc)
- long & [iword](#) (int __ix)
- char [narrow](#) ([char_type](#) __c, char __dfault) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) __prec)
- void *& [pword](#) (int __ix)
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) () const
- [basic_streambuf](#)< [_CharT](#), [_Traits](#) > * [rdbuf](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > * __sb)
- [iostate](#) [rdstate](#) () const
- void [register_callback](#) ([event_callback](#) __fn, int __index)
- [__ostream_type](#) & [seekp](#) ([pos_type](#))
- [__ostream_type](#) & [seekp](#) ([off_type](#), [ios_base::seekdir](#))
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl, [fmtflags](#) __mask)
- void [setstate](#) ([iostate](#) __state)
- [pos_type](#) [tellp](#) ()
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) ([basic_ostream](#)< [_CharT](#), [_Traits](#) > * __tistr)
- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) () const
- void [unsetf](#) ([fmtflags](#) __mask)
- [char_type](#) [widen](#) (char __c) const
- [streamsize](#) [width](#) () const

- `streamsize width (streamsize __wide)`
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`

- void [_M_write](#) (const [char_type](#) *__s, [streamsize](#) __n)
- [__ostream_type](#) & [write](#) (const [char_type](#) *__s, [streamsize](#) __n)
- [operator void *](#) () const
- bool [operator!](#) () const

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const [locale](#) &__loc)
- void **_M_call_callbacks** ([event](#) __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- [_Words](#) & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
[__ostream_type](#) & **_M_insert** (_ValueT __v)
- void **init** ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- [_Callback_list](#) * **_M_callbacks**
- const [__ctype_type](#) * **_M_ctype**
- [iostate](#) **_M_exception**
- [char_type](#) **_M_fill**
- bool **_M_fill_init**
- [fmtflags](#) **_M_flags**
- [locale](#) **_M_ios_locale**
- [_Words](#) **_M_local_word** [[_S_local_word_size](#)]
- const [__num_get_type](#) * **_M_num_get**
- const [__num_put_type](#) * **_M_num_put**
- [streamsize](#) **_M_precision**
- [basic_streambuf](#)< _CharT, _Traits > * **_M_streambuf**
- [iostate](#) **_M_streambuf_state**
- [basic_ostream](#)< _CharT, _Traits > * **_M_tie**
- [streamsize](#) **_M_width**
- [_Words](#) * **_M_word**
- int **_M_word_size**
- [_Words](#) **_M_word_zero**

Friends

- class **sentry**

5.374.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_ostream< _-  
CharT, _Traits >
```

Controlling output.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from [basic_streambuf](#) to do the actual output.

Definition at line 55 of file ostream.

5.374.2 Member Typedef Documentation

5.374.2.1 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_ostream< _CharT, _Traits
>::__ctype_type`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 71 of file ostream.

5.374.2.2 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_ios< _CharT, _Traits >::__num_get_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_istream< _CharT, _Traits >](#), [std::basic_istream< char, _Traits >](#), and [std::basic_istream< char >](#).

Definition at line 86 of file basic_ios.h.

5.374.2.3 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> >
std::basic_ostream< _CharT, _Traits >::__num_put_type`

These are non-standard types.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 70 of file ostream.

5.374.2.4 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_ostream< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ofstream< _CharT, _Traits >](#), [std::basic_fstream< _CharT, _Traits >](#), [std::basic_iostream< _CharT, _Traits >](#), [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#), [std::basic_stringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_istream< char >](#).

Definition at line 59 of file ostream.

5.374.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int) [inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the [ios_base](#) object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several [ios_base](#) and [basic_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 435 of file ios_base.h.

5.374.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`

- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 254 of file `ios_base.h`.

5.374.2.7 `template<typename _CharT, typename _Traits> typedef _Traits::int_type std::basic_ostream< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ofstream< _CharT, _Traits >](#), [std::basic_fstream< _CharT, _Traits >](#), [std::basic_iostream< _CharT, _Traits >](#), [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#), [std::basic_stringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_istream< char >](#).

Definition at line 60 of file `ostream`.

5.374.2.8 `typedef _Ios_Iostate std::ios_base::iostate` [inherited]

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file `ios_base.h`.

5.374.2.9 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_ostream< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ofstream< _CharT, _Traits >](#), [std::basic_fstream< _CharT, _Traits >](#), [std::basic_iostream< _CharT, _Traits >](#), [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#), [std::basic_stringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_istream< char >](#).

Definition at line 62 of file `ostream`.

5.374.2.10 `typedef _Ios_Openmode std::ios_base::openmode` [inherited]

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`

- out
- trunc

Definition at line 360 of file ios_base.h.

5.374.2.11 `template<typename _CharT, typename _Traits> typedef _Traits::pos_type std::basic_ostream< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, `std::basic_stringstream< _CharT, _Traits, _Alloc >`, and `std::basic_istream< char >`.

Definition at line 61 of file ostream.

5.374.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 392 of file ios_base.h.

5.374.2.13 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_ostream< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_ios< _CharT, _Traits >`.

Reimplemented in `std::basic_ofstream< _CharT, _Traits >`, `std::basic_fstream< _CharT, _Traits >`, `std::basic_iostream< _CharT, _Traits >`, `std::basic_ostringstream<`

[_CharT, _Traits, _Alloc >](#), [std::basic_stringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_istream< char >](#).

Definition at line 63 of file ostream.

5.374.3 Member Enumeration Documentation

5.374.3.1 enum std::ios_base::event [inherited]

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during [imbue\(\)](#). copyfmt_event is used during copyfmt().

Definition at line 418 of file ios_base.h.

5.374.4 Constructor & Destructor Documentation

5.374.4.1 template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits >::basic_ostream (__streambuf_type * __sb) [inline, explicit]

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 82 of file ostream.

5.374.4.2 template<typename _CharT, typename _Traits> virtual std::basic_ostream< _CharT, _Traits >::~~basic_ostream () [inline, virtual]

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 91 of file ostream.

5.374.5 Member Function Documentation

5.374.5.1 const locale& std::ios_base::_M_getloc () const [inline, inherited]

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

5.374.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream<_CharT, _Traits>::_M_write (const char_type
* __s, streamsize __n) [inline]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file `ostream`.

5.374.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits>::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.374.5.4 `template<typename _CharT, typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

5.374.5.5 `template<typename _CharT, typename _Traits > basic_ios<
_CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
const basic_ios< _CharT, _Traits > & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios< _CharT, _Traits >::tie()`, and `std::ios_base::width()`.

5.374.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::eof () const [inline, inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file basic_ios.h.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.374.5.7 `template<typename _CharT, typename _Traits> iostate
std::basic_ios< _CharT, _Traits >::exceptions () const [inline,
inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic_ios.h.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.374.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios<
_CharT, _Traits >::exceptions (iostate __except) [inline,
inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

```

#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}

```

Definition at line 247 of file basic_ios.h.

5.374.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in [fail\(\)](#) is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file basic_ios.h.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.374.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill() const [inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.374.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.374.5.12 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.374.5.13 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.374.5.14 `template<typename _CharT, typename _Traits> basic_ostream<
_CharT, _Traits> & std::basic_ostream< _CharT, _Traits>::flush
()`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.374.5.15 `locale std::ios_base::getloc() const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by `std::basic_ios< _CharT, _Traits>::copyfmt()`, `std::money_put< _CharT, _OutIter>::do_put()`, `std::basic_ios< _CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.374.5.16 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.374.5.17 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.374.5.18 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.374.5.19 long& std::ios_base::iword (int __ix) [inline, inherited]

Access to integer array.

Parameters

__ix Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file ios_base.h.

5.374.5.20 template<typename _CharT, typename _Traits> char std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char __dfault) const [inline, inherited]

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of *char_type* to a character of *char*, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,dfault)
```

Additional 110n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file basic_ios.h.

5.374.5.21 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits >::operator void * () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.374.5.22 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! () const [inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.374.5.23 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

5.374.5.24 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

5.374 std::basic_ostream<_CharT, _Traits> Class Template Reference 1941

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.374.5.25 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (__streambuf_type * __sb)`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.374.5.26 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file `ostream`.

5.374.5.27 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<<(int __n)`

Basic arithmetic inserters.

Parameters

`A` variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.374.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(__ostream_type &(*)(__ostream_type &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

5.374.5.29 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<<(unsigned int __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

5.374.5.30 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file ostream.

5.374.5.31 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.374.5.32 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.374.5.33 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (bool __n)
[inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.374.5.34 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (const void *
__p) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.374.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (double __f) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.374.5.36 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) __pf) [inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `io manip` header.

Definition at line 127 of file ostream.

5.374.5.37 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (short __n)`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References [std::ios_base::basefield](#), [std::ios_base::flags\(\)](#), [std::ios_base::hex](#), and [std::ios_base::oct](#).

5.374.5.38 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.374.5.39 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.374.5.40 `streamsize std::ios_base::precision (streamsize __prec)
[inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

5.374.5.41 `streamsize std::ios_base::precision () const [inline,
inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

5.374.5.42 `template<typename _CharT , typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
char_type __c)`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.374.5.43 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

`__ix` Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.374.5.44 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb) [inherited]`

Changing the underlying buffer.

Parameters

`sb` The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

5.374 std::basic_ostream< _CharT, _Traits > Class Template Reference 1949

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::clear().

5.374.5.45 `template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT,
_Traits>::rdbuf() const [inline, inherited]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), [std::basic_ofstream< _CharT, _Traits >](#), [std::basic_fstream< _CharT, _Traits >](#), [std::basic_istream< _CharT, _Traits, _Alloc >](#), [std::basic_ostringstream< _CharT, _Traits, _Alloc >](#), and [std::basic_stringstream< _CharT, _Traits, _Alloc >](#).

Definition at line 311 of file basic_ios.h.

Referenced by [std::basic_ostream< _CharT, _Traits >::flush\(\)](#), [std::basic_istream< _CharT, _Traits >::get\(\)](#), [std::basic_istream< _CharT, _Traits >::getline\(\)](#), [std::getline\(\)](#), [std::basic_istream< _CharT, _Traits >::ignore\(\)](#), [std::basic_ios< _CharT, _Traits >::imbue\(\)](#), [std::basic_ostream< _CharT, _Traits >::operator<<\(\)](#), [std::basic_istream< _CharT, _Traits >::operator>>\(\)](#), [std::operator>>\(\)](#), [std::basic_istream< _CharT, _Traits >::peek\(\)](#), [std::basic_ostream< _CharT, _Traits >::put\(\)](#), [std::basic_istream< _CharT, _Traits >::putback\(\)](#), [std::basic_istream< _CharT, _Traits >::read\(\)](#), [std::basic_istream< _CharT, _Traits >::readsome\(\)](#), [std::basic_istream< _CharT, _Traits >::seekg\(\)](#), [std::basic_ostream< _CharT, _Traits >::seekp\(\)](#), [std::basic_istream< _CharT, _Traits >::sync\(\)](#), [std::basic_istream< _CharT, _Traits >::tellg\(\)](#), [std::basic_ostream< _CharT, _Traits >::tellp\(\)](#), [std::basic_istream< _CharT, _Traits >::unget\(\)](#), and [std::ws\(\)](#).

5.374.5.46 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate() const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See [std::ios_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

5.374.5.47 `void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.374.5.48 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits> & std::basic_ostream< _CharT, _Traits>::seekp (pos_type __pos)`

Changing the current write position.

Parameters

`pos` A file position object.

Returns

`*this`

If `fail()` is not true, calls `rdbuf() -> pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits>::rdbuf()`, and `std::basic_ios< _CharT, _Traits>::setstate()`.

5.374.5.49 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (off_type __off, ios_base::seekdir __dir)`

Changing the current write position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf() -> pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.374.5.50 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file ios_base.h.

5.374.5.51 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.374.5.52 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file basic_ios.h.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.374.5.53 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.374.5.54 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits>::pos_type std::basic_ostream< _CharT, _Traits>::tellp ()`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits>::rdbuf()`.

5.374.5.55 `template<typename _CharT, typename _Traits> basic_ostream< _CharT, _Traits>* std::basic_ios< _CharT, _Traits>::tie () const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits>::copyfmt()`.

5.374.5.56 `template<typename _CharT, typename _Traits>
 basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (basic_ostream<_CharT, _Traits> * __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file `basic_ios.h`.

5.374.5.57 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.374.5.58 `template<typename _CharT, typename _Traits> char_type
 std::basic_ios<_CharT, _Traits>::widen (char __c) const
 [inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

5.374 std::basic_ostream<_CharT, _Traits> Class Template Reference 1955

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.374.5.59 streamsize std::ios_base::width () const [inline, inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutT, _Traits>::do_put()`, and `std::operator>>()`.

5.374.5.60 streamsize std::ios_base::width (streamsize __wide) [inline, inherited]

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.374.5.61 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::write (const char_type * __s, streamsize __n)

Character string insertion.

Parameters

- s* The array to insert.
- n* Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.374.5.62 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.374.6 Member Data Documentation

5.374.6.1 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

5.374.6.2 const openmode std::ios_base::app [static, inherited]

Seek to end before each write.

Definition at line 363 of file ios_base.h.

5.374.6.3 const openmode std::ios_base::ate [static, inherited]

Open and seek to end immediately after opening.

Definition at line 366 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.374.6.4 const iostate std::ios_base::badbit [static, inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 333 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.374.6.5 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 312 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.374.6.6 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file ios_base.h.

5.374.6.7 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.374.6.8 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.374.6.9 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.374.6.10 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file ios_base.h.

5.374.6.11 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.374.6.12 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.374.6.13 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.374.6.14 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.374.6.15 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 315 of file ios_base.h.

5.374.6.16 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.374.6.17 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.374.6.18 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 374 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.374.6.19 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 271 of file ios_base.h.

5.374.6.20 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.374.6.21 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 278 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.374.6.22 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 377 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.374.6.23 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file ios_base.h.

5.374.6.24 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 285 of file ios_base.h.

5.374.6.25 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file ios_base.h.

5.374.6.26 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file ios_base.h.

5.374.6.27 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file ios_base.h.

5.374.6.28 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 299 of file ios_base.h.

5.374.6.29 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for ofstream.

Definition at line 380 of file ios_base.h.

5.374.6.30 const fmtflags std::ios_base::unitbuf [static, inherited]

Flushes output after each output operation.

Definition at line 302 of file ios_base.h.

5.374.6.31 const fmtflags std::ios_base::uppercase [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file ios_base.h.

Referenced by std::num_put<_CharT, _OutIter >::do_put().

The documentation for this class was generated from the following files:

- [ostream](#)
- [ostream.tcc](#)

5.375 `std::basic_ostream< _CharT, _Traits >::sentry` Class Reference

Performs setup work for output streams.

Public Member Functions

- `sentry` (`basic_ostream< _CharT, _Traits > &__os`)
- `~sentry` ()
- `operator bool` () const

5.375.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::basic_ostream< _CharT, _Traits >::sentry`

Performs setup work for output streams. Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

Definition at line 377 of file ostream.

5.375.2 Constructor & Destructor Documentation

5.375.2.1 `template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::sentry (
basic_ostream< _CharT, _Traits > & __os) [explicit]`

The constructor performs preparatory work.

Parameters

os The output stream to guard.

If the stream state is good (*os.good()* is true), then if the stream is tied to another output stream, *is.tie()* -> `flush()` is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file ostream.tcc.

5.375.2.2 `template<typename _CharT, typename _Traits>`
`std::basic_ostream< _CharT, _Traits >::sentry::~sentry ()`
`[inline]`

Possibly flushes the stream.

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor calls `flush()` on the output stream.

Definition at line 405 of file `ostream`.

References `std::uncaught_exception()`.

5.375.3 Member Function Documentation

5.375.3.1 `template<typename _CharT, typename _Traits>`
`std::basic_ostream< _CharT, _Traits >::sentry::operator bool ()`
`const [inline, explicit]`

Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

Definition at line 426 of file `ostream`.

The documentation for this class was generated from the following files:

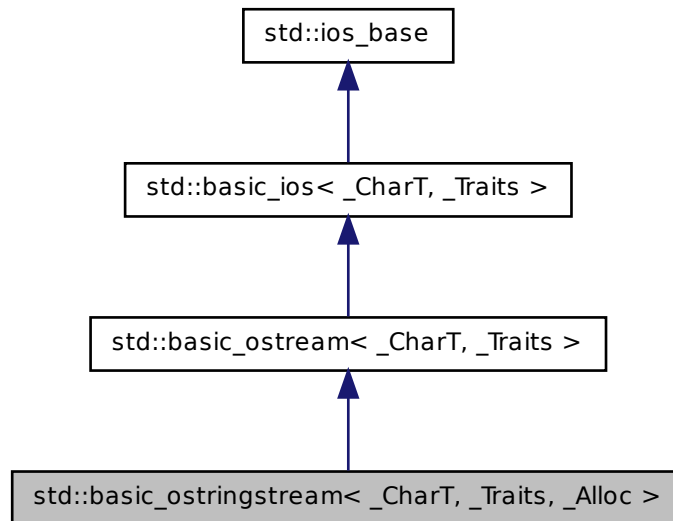
- [ostream](#)
- [ostream.tcc](#)

5.376 `std::basic_ostringstream< _CharT, _Traits, _-` `Alloc >` Class Template Reference

Controlling output for `std::string`.

This class supports writing to objects of type [std::basic_string](#), using the inherited functions from [std::basic_ostream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_ostringstream< _CharT, _Traits, _Alloc >`:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< char_type, traits_type > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback)(event, ios_base &, int)`
- typedef `_Ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `int io_state`

- typedef `_Ios_Iostate` [iostate](#)
 - typedef `traits_type::off_type` [off_type](#)
 - typedef `int` **`open_mode`**
 - typedef `_Ios_Openmode` [openmode](#)
 - typedef `traits_type::pos_type` [pos_type](#)
 - typedef `int` **`seek_dir`**
 - typedef `_Ios_Seekdir` [seekdir](#)
 - typedef `std::streamoff` **`streamoff`**
 - typedef `std::streampos` **`streampos`**
 - typedef `_Traits` [traits_type](#)
-
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __-`
`num_get_type`

Public Member Functions

- [basic_ostringstream](#) (`ios_base::openmode __mode=ios_base::out`)
- [basic_ostringstream](#) (`const __string_type &__str, ios_base::openmode __-`
`mode=ios_base::out`)
- [~basic_ostringstream](#) ()
- `const locale & _M_getloc` () const
- `void _M_setstate` (`iostate __state`)
- `bool bad` () const
- `void clear` (`iostate __state=goodbit`)
- [basic_ios](#) & [copyfmt](#) (`const basic_ios &__rhs`)
- `bool eof` () const
- [iostate exceptions](#) () const
- `void exceptions` (`iostate __except`)
- `bool fail` () const
- [char_type fill](#) () const
- [char_type fill](#) (`char_type __ch`)
- [fmtflags flags](#) (`fmtflags __fmtfl`)
- [fmtflags flags](#) () const
- [__ostream_type & flush](#) ()
- [locale getloc](#) () const
- `bool good` () const
- [locale imbue](#) (`const locale &__loc`)
- `long & iword` (`int __ix`)
- `char narrow` (`char_type __c, char __dfault`) const
- [streamsize precision](#) () const
- [streamsize precision](#) (`streamsize __prec`)
- `void *& pword` (`int __ix`)

- `__stringbuf_type * rdbuf ()` const
- `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
- `ios_base::rdstate ()` const
- `void register_callback (event_callback __fn, int __index)`
- `__ostream_type & seekp (pos_type)`
- `__ostream_type & seekp (off_type, ios_base::seekdir)`
- `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
- `fmtflags setf (fmtflags __fmtfl)`
- `void setstate (ios_base::iostate __state)`
- `void str (const __string_type & __s)`
- `__string_type str ()` const
- `pos_type tellp ()`
- `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tistr)`
- `basic_ostream< _CharT, _Traits > * tie ()` const
- `void unsetf (fmtflags __mask)`
- `char_type widen (char __c)` const
- `streamsize width ()` const
- `streamsize width (streamsize __wide)`

- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
- `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
- `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`

- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`
- `operator void * () const`
- `bool operator! () const`

Static Public Member Functions

- static bool `sync_with_stdio` (bool __sync=true)
- static int `xalloc` () throw ()

Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`
- static const `seekdir cur`
- static const `fmtflags dec`

- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- template<typename _ValueT >
[__ostream_type](#) & [_M_insert](#) (_ValueT __v)
- void [init](#) ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

Friends

- `class sentry`

5.376.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class
std::basic_ostringstream<_CharT, _Traits, _Alloc>
```

Controlling output for `std::string`.

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 366 of file `sstream`.

5.376.2 Member Typedef Documentation

5.376.2.1 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_ostream<_CharT, _Traits
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file `ostream`.

5.376.2.2 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_ios<_CharT, _Traits>::__num_get_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_istream<_CharT, _Traits>](#), [std::basic_istream<char, _Traits>](#), and [std::basic_istream<char>](#).

Definition at line 86 of file `basic_ios.h`.

5.376.2.3 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file `ostream`.

5.376.2.4 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _CharT std::basic_ostringstream<_CharT, _Traits, _Alloc
>::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Definition at line 370 of file `sstream`.

5.376.2.5 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`
[inherited]

The type of an event callback function.

Parameters

- event* One of the members of the event enum.
- ios_base* Reference to the `ios_base` object.
- int* The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 435 of file `ios_base.h`.

5.376.2.6 `typedef _Ios_Fmtflags std::ios_base::fmtflags` **[inherited]**

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1973

- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 254 of file ios_base.h.

5.376.2.7 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::int_type std::basic_ostringstream< _CharT,
_Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Definition at line 375 of file sstream.

5.376.2.8 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 329 of file ios_base.h.

5.376.2.9 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::off_type std::basic_ostringstream< _CharT,
_Traits, _Alloc >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Definition at line 377 of file sstream.

5.376.2.10 `typedef _Ios_Openmode std::ios_base::openmode` **[inherited]**

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 360 of file `ios_base.h`.

5.376.2.11 `template<typename _CharT, typename _Traits, typename _Alloc>` `typedef traits_type::pos_type std::basic_ostringstream<_CharT,` `_Traits, _Alloc>::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream<_CharT, _Traits>](#).

Definition at line 376 of file sstream.

5.376.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir` **[inherited]**

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1975

Definition at line 392 of file ios_base.h.

5.376.2.13 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _Traits std::basic_ostringstream< _CharT, _Traits, _Alloc
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ostream< _CharT, _Traits >](#).

Definition at line 371 of file sstream.

5.376.3 Member Enumeration Documentation

5.376.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during [imbue\(\)](#). copyfmt_event is used during copyfmt().

Definition at line 418 of file ios_base.h.

5.376.4 Constructor & Destructor Documentation

5.376.4.1 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc
>::basic_ostringstream (ios_base::openmode __mode =
ios_base::out) [inline, explicit]`

Default constructor starts with an empty string buffer.

Parameters

mode Whether the buffer can read, or write, or both.

[ios_base::out](#) is automatically included in *mode*.

Initializes sb using mode|out, and passes &sb to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 402 of file sstream.

5.376.4.2 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc>::basic_ostringstream (const __string_type & __str, ios_base::openmode __mode = ios_base::out) [inline, explicit]`

Starts with an existing string buffer.

Parameters

str A string to copy as a starting buffer.

mode Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes *sb* using *str* and `mode|out`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 420 of file `sstream`.

5.376.4.3 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc>::~~basic_ostringstream () [inline]`

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 431 of file `sstream`.

5.376.5 Member Function Documentation

5.376.5.1 `const locale& std::ios_base::_M_getloc () const [inline, inherited]`

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get<`

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1977

_CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

5.376.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream< _CharT, _Traits >::_M_write (const char_type
* __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

*this

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file ostream.

5.376.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits >::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file basic_ios.h.

5.376.5.4 `template<typename _CharT, typename _Traits > void
std::basic_ios< _CharT, _Traits >::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.376.5.5 `template<typename _CharT, typename _Traits> basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

`__rhs` The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.376.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::eof() const [inline, inherited]`

Fast error checking.

Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 180 of file `basic_ios.h`.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1979

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), and std::basic_istream< _CharT, _Traits >::unget().

5.376.5.7 `template<typename _CharT, typename _Traits> iostate
std::basic_ios< _CharT, _Traits >::exceptions () const [inline,
inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

5.376.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios<
_CharT, _Traits >::exceptions (iostate __except) [inline,
inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");
```

```

std::cerr << "Setting badbit\n";
f.setstate (std::ios_base::badbit);

std::cerr << "Setting exception mask\n";
f.exceptions (std::ios_base::badbit);
}

```

Definition at line 247 of file `basic_ios.h`.

5.376.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const` `[inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.376.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill() const` `[inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1981

5.376.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios< _CharT, _Traits >::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.376.5.12 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.376.5.13 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline, inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.376.5.14 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.376.5.15 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.376.5.16 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::good () const [inline, inherited]`

Fast error checking.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1983

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 170 of file basic_ios.h.

5.376.5.17 `template<typename _CharT , typename _Traits > locale
std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)
[inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Reimplemented from [std::ios_base](#).

Definition at line 113 of file basic_ios.tcc.

References `std::ios_base::getloc()`, and `std::basic_ios< _CharT, _Traits >::rdbuf()`.

Referenced by `std::operator<<()`.

5.376.5.18 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::init (basic_streambuf<
_CharT, _Traits > * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file basic_ios.tcc.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.376.5.19 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

`__ix` Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file `ios_base.h`.

5.376.5.20 `template<typename _CharT, typename _Traits> char
std::basic_ios< _CharT, _Traits >::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

`c` The character to narrow.

`dfault` The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 420 of file `basic_ios.h`.

5.376.5.21 `template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits >::operator void * () const [inline,
inherited]`

The quick-and-easy status check.

5.376 std::basic_ostringstream<_CharT, _Traits, _Alloc> Class Template Reference 1985

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.376.5.22 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.376.5.23 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits>::operator<< (long __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file `ostream`.

5.376.5.24 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream<_CharT, _Traits>::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.376.5.25 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (
__ostream_type &(*)(__ostream_type &) __pf) [inline,
inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `omanip` header.

Definition at line 108 of file ostream.

5.376.5.26 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.376.5.27 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (bool __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1987

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.376.5.28 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (const void * __p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.376.5.29 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 117 of file ostream.

5.376.5.30 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) [inherited]`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ostream<_CharT, _Traits>::rdbuf()`, and `std::basic_ostream<_CharT, _Traits>::setstate()`.

5.376.5.31 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file ostream.

5.376.5.32 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (unsigned short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1989

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.376.5.33 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<<(int __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.376.5.34 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<(ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "`std::cout << std::endl`". For more information, see the `omanip` header.

Definition at line 127 of file ostream.

5.376.5.35 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<(unsigned int __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 191 of file ostream.

```
5.376.5.36  template<typename _CharT, typename _Traits> __ostream_type&  
             std::basic_ostream< _CharT, _Traits >::operator<< ( unsigned  
             long __n ) [inline, inherited]
```

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 169 of file ostream.

```
5.376.5.37  template<typename _CharT , typename _Traits > basic_ostream<  
             _CharT, _Traits > & std::basic_ostream< _CharT, _Traits  
             >::operator<< ( short __n ) [inherited]
```

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1991

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.376.5.38 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (float __f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.376.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.376.5.40 `streamsize std::ios_base::precision (streamsize __prec) [inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file `ios_base.h`.

5.376.5.41 `streamsize std::ios_base::precision () const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, and `std::operator<<()`.

5.376.5.42 `template<typename _CharT, typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (char_type __c) [inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

**this*

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1993

5.376.5.43 void*& std::ios_base::pword (int __ix) [inline, inherited]

Access to void pointer array.

Parameters

__ix Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.376.5.44 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > * __sb) [inherited]

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::clear()`.

5.376.5.45 `template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type* std::basic_ostringstream<_CharT, _Traits,
_Alloc>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Reimplemented from `std::basic_ios<_CharT, _Traits>`.

Definition at line 442 of file `sstream`.

5.376.5.46 `template<typename _CharT, typename _Traits> iostate
std::basic_ios<_CharT, _Traits>::rdstate () const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 127 of file `basic_ios.h`.

5.376.5.47 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

`__fn` The function to add.

`__index` The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template
Reference **1995**

5.376.5.48 `template<typename _CharT, typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp
(pos_type __pos) [inherited]`

Changing the current write position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.376.5.49 `template<typename _CharT, typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp
(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.376.5.50 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask)`
[inline, inherited]

Setting new format flags.

Parameters

fmtfl Additional flags to set.
mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.376.5.51 `fmtflags std::ios_base::setf (fmtflags __fmtfl)` **[inline, inherited]**

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.376.5.52 `template<typename _CharT, typename _Traits> void`
`std::basic_ios< _CharT, _Traits >::setstate (iostate __state)`
[inline, inherited]

Sets additional flags in the error state.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 1997

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.376.5.53 `template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_ostringstream< _CharT, _Traits, _Alloc
>::str () const [inline]`

Copying out the string buffer.

Returns

`rddbuf ()->str ()`

Definition at line 450 of file `sstream`.

Referenced by `std::operator<<()`.

5.376.5.54 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_ostringstream< _CharT, _Traits, _Alloc >::str (
const __string_type & __s) [inline]`

Setting a new buffer.

Parameters

s The string to use as a new sequence.

Calls `rddbuf ()->str (s)`.

Definition at line 460 of file `sstream`.

5.376.5.55 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.376.5.56 `template<typename _CharT, typename _Traits> basic_ostream<
_CharT, _Traits>::pos_type std::basic_ostream< _CharT, _Traits
>::tellp () [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios< _CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios< _CharT, _Traits>::rddbuf()`.

5.376.5.57 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits
>::tie () const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

5.376 std::basic_ostringstream<_CharT, _Traits, _Alloc> Class Template Reference 1999

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 285 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

```
5.376.5.58 template<typename _CharT, typename _Traits>
    basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>
    >::tie ( basic_ostream<_CharT, _Traits> * __tiestr ) [inline,
    inherited]
```

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 297 of file `basic_ios.h`.

```
5.376.5.59 void std::ios_base::unsetf ( fmtflags __mask ) [inline,
    inherited]
```

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

```
5.376.5.60 template<typename _CharT, typename _Traits> char_type
    std::basic_ios<_CharT, _Traits>::widen ( char __c ) const
    [inline, inherited]
```

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file `basic_ios.h`.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.376.5.61 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.376.5.62 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 2001

5.376.5.63 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inherited]`

Character string insertion.

Parameters

- s* The array to insert.
- n* Maximum number of characters to insert.

Returns

*this

Characters are copied from *s* and inserted into the stream until one of the following happens:

- *n* characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.376.5.64 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.376.6 Member Data Documentation

5.376.6.1 `const fmtflags std::ios_base::adjustfield` `[static, inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.376.6.2 `const openmode std::ios_base::app` `[static, inherited]`

Seek to end before each write.

Definition at line 363 of file `ios_base.h`.

5.376.6.3 `const openmode std::ios_base::ate` `[static, inherited]`

Open and seek to end immediately after opening.

Definition at line 366 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.376.6.4 `const iostate std::ios_base::badbit` `[static, inherited]`

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 333 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.376.6.5 const fmtflags std::ios_base::basefield [static, inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.376.6.6 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file `ios_base.h`.

5.376.6.7 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). `/// This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.`

Definition at line 371 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`.

5.376.6.8 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

5.376.6.9 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_istream< _CharT, _Traits >::tellg()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.376.6.10 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file ios_base.h.

5.376.6.11 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.376.6.12 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsomewhat(), and std::ws().

5.376.6.13 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.376.6.14 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.376 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference 2005

5.376.6.15 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 315 of file `ios_base.h`.

5.376.6.16 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 344 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.376.6.17 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.376.6.18 const openmode std::ios_base::in [static, inherited]

Open for input. Default for `ifstream` and `fstream`.

Definition at line 374 of file `ios_base.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.376.6.19 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to `right` if no such point is /// designated.

Definition at line 271 of file `ios_base.h`.

5.376.6.20 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

5.376.6.21 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 278 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::operator<<()`.

5.376.6.22 const openmode std::ios_base::out [static, inherited]

Open for output. Default for `ofstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_ostream<_CharT, _Traits>::seekp()`, and `std::basic_ostream<_CharT, _Traits>::tellp()`.

5.376.6.23 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file `ios_base.h`.

5.376.6.24 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 285 of file `ios_base.h`.

5.376.6.25 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file ios_base.h.

5.376.6.26 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file ios_base.h.

5.376.6.27 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file ios_base.h.

5.376.6.28 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 299 of file ios_base.h.

5.376.6.29 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for ofstream.

Definition at line 380 of file ios_base.h.

5.376.6.30 const fmtflags std::ios_base::unitbuf [static, inherited]

Flushes output after each output operation.

Definition at line 302 of file ios_base.h.

5.376.6.31 const fmtflags std::ios_base::uppercase [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

The documentation for this class was generated from the following file:

- [sstream](#)

5.377 `std::basic_regex< _Ch_type, _Rx_traits >` Class Template Reference

Public Types

- typedef [regex_constants::syntax_option_type](#) **flag_type**
- typedef `_Rx_traits::locale_type` **locale_type**
- typedef `_Rx_traits::string_type` **string_type**
- typedef `_Ch_type` **value_type**

Public Member Functions

- [basic_regex](#) ()
- [basic_regex](#) (const `_Ch_type` *__p, `flag_type` __f=[regex_constants::ECMAScript](#))
- [basic_regex](#) (const [basic_regex](#) &__rhs)
- [basic_regex](#) ([initializer_list](#)< `_Ch_type` > __l, `flag_type` __f=[regex_constants::ECMAScript](#))
- [basic_regex](#) (const [basic_regex](#) &&__rhs)
- [basic_regex](#) (const `_Ch_type` *__p, `std::size_t` __len, `flag_type` __f)
- `template<typename _Ch_traits, typename _Ch_alloc >`
[basic_regex](#) (const [std::basic_string](#)< `_Ch_type`, `_Ch_traits`, `_Ch_alloc` > &__s, `flag_type` __f=[regex_constants::ECMAScript](#))
- `template<typename _InputIterator >`
[basic_regex](#) (_InputIterator __first, _InputIterator __last, `flag_type` __f=[regex_constants::ECMAScript](#))
- `~basic_regex` ()
- `const __regex::__AutomatonPtr & _M_get_automaton` () const
- `template<typename _InputIterator >`
[basic_regex](#) & [assign](#) (_InputIterator __first, _InputIterator __last, `flag_type` __flags=[regex_constants::ECMAScript](#))
- [basic_regex](#) & [assign](#) (const `_Ch_type` *__p, `flag_type` __flags=[regex_constants::ECMAScript](#))
- [basic_regex](#) & [assign](#) (const `_Ch_type` *__p, `std::size_t` __len, `flag_type` __flags)
- `template<typename _Ch_traits, typename _Allocator >`
[basic_regex](#) & [assign](#) (const [basic_string](#)< `_Ch_type`, `_Ch_traits`, `_Allocator` > &__s, `flag_type` __f=[regex_constants::ECMAScript](#))

- `basic_regex` & `assign` (`initializer_list`<_Ch_type> __l, `flag_type` __f=`regex_constants::ECMAScript`)
- `basic_regex` & `assign` (`const basic_regex` &__rhs)
- `basic_regex` & `assign` (`basic_regex` &&__rhs)
- `flag_type` `flags` () `const`
- `locale_type` `getloc` () `const`
- `locale_type` `imbue` (`locale_type` __loc)
- `unsigned int` `mark_count` () `const`
- `basic_regex` & `operator=` (`const _Ch_type` *__p)
- `basic_regex` & `operator=` (`const basic_regex` &__rhs)
- `basic_regex` & `operator=` (`basic_regex` &&__rhs)
- `template`<`typename` _Ch_type, `traits` , `typename` _Allocator >
`basic_regex` & `operator=` (`const basic_string`<_Ch_type, _Ch_type, traits, _Allocator> &__s)
- `void` `swap` (`basic_regex` &__rhs)

Static Public Attributes

Constants

std [28.8.1](1)

Todo

These should be constexpr.

- `static const` `regex_constants::syntax_option_type` `icase`
- `static const` `regex_constants::syntax_option_type` `nosubs`
- `static const` `regex_constants::syntax_option_type` `optimize`
- `static const` `regex_constants::syntax_option_type` `collate`
- `static const` `regex_constants::syntax_option_type` `ECMAScript`
- `static const` `regex_constants::syntax_option_type` `basic`
- `static const` `regex_constants::syntax_option_type` `extended`
- `static const` `regex_constants::syntax_option_type` `awk`
- `static const` `regex_constants::syntax_option_type` `grep`
- `static const` `regex_constants::syntax_option_type` `egrep`

Protected Attributes

- `__regex::AutomatonPtr` `_M_automaton`
- `flag_type` `_M_flags`
- `_Rx_traits` `_M_traits`

5.377.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> class std::basic_regex< _Ch_type, _Rx_traits >
```

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 340 of file `regex.h`.

5.377.2 Constructor & Destructor Documentation

5.377.2.1 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits >::basic_regex () [inline]`

Constructs a basic regular expression that does not match any character sequence.

Definition at line 382 of file `regex.h`.

5.377.2.2 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (const _Ch_type * __p, flag_type __f = regex_constants::ECMAScript) [inline, explicit]`

Constructs a basic regular expression from the sequence `[p, p + char_traits<_Ch_type>::length(p))` interpreted according to the flags in `f`.

Parameters

p A pointer to the start of a C-style null-terminated string containing a regular expression.

f Flags indicating the syntax rules and options.

Exceptions

regex_error if *p* is not a valid regular expression.

Definition at line 400 of file `regex.h`.

5.377 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference 2011

5.377.2.3 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex (const _Ch_type * __p, std::size_t __len, flag_type
__f) [inline]`

Constructs a basic regular expression from the sequence [p, p + len) interpreted according to the flags in f.

Parameters

- p* A pointer to the start of a string containing a regular expression.
- len* The length of the string containing the regular expression.
- f* Flags indicating the syntax rules and options.

Exceptions

- regex_error* if p is not a valid regular expression.

Definition at line 418 of file regex.h.

5.377.2.4 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex (const basic_regex< _Ch_type, _Rx_traits > &
__rhs) [inline]`

Copy-constructs a basic regular expression.

Parameters

- rhs* A regex object.

Definition at line 428 of file regex.h.

5.377.2.5 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex (const basic_regex< _Ch_type, _Rx_traits > &&
__rhs) [inline]`

Move-constructs a basic regular expression.

Parameters

- rhs* A regex object.

Definition at line 438 of file regex.h.

5.377.2.6 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> template<typename _Ch_traits ,
 typename _Ch_alloc > std::basic_regex< _Ch_type, _Rx_traits
 >::basic_regex (const std::basic_string< _Ch_type, _Ch_traits,
 _Ch_alloc > & __s, flag_type __f= regex_constants::ECMAScript)
 [inline, explicit]`

Constructs a basic regular expression from the string *s* interpreted according to the flags in *f*.

Parameters

s A string containing a regular expression.

f Flags indicating the syntax rules and options.

Exceptions

regex_error if *s* is not a valid regular expression.

Definition at line 454 of file regex.h.

5.377.2.7 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> template<typename _InputIterator
 > std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 _InputIterator __first, _InputIterator __last, flag_type __f=
 regex_constants::ECMAScript) [inline]`

Constructs a basic regular expression from the range [*first*, *last*) interpreted according to the flags in *f*.

Parameters

first The start of a range containing a valid regular expression.

last The end of a range containing a valid regular expression.

f The format flags of the regular expression.

Exceptions

regex_error if [*first*, *last*) is not a valid regular expression.

Definition at line 476 of file regex.h.

5.377.2.8 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::basic_regex (initializer_list<_Ch_type> __l, flag_type __f =
regex_constants::ECMAScript) [inline]`

Constructs a basic regular expression from an initializer list.

Parameters

l The initializer list.

f The format flags of the regular expression.

Exceptions

[*regex_error*](#) if *l* is not a valid regular expression.

Definition at line 490 of file regex.h.

5.377.2.9 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits
>::~~basic_regex () [inline]`

Destroys a basic regular expression.

Definition at line 500 of file regex.h.

5.377.3 Member Function Documentation

5.377.3.1 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits>::assign (const basic_regex< _Ch_type,
_Rx_traits> & __rhs) [inline]`

the real assignment operator.

Parameters

rhs Another regular expression object.

Definition at line 546 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::swap().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::operator=().

5.377.3.2 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::assign (const _Ch_type * __p, std::size_t
 __len, flag_type __flags) [inline]`

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

Parameters

- p* A pointer to a C-style string containing a regular expression pattern.
- len* The length of the regular expression pattern string.
- flags* Syntax option flags.

Exceptions

- [*regex_error*](#) if *p* does not contain a valid regular expression pattern interpreted according to *flags*. If [*regex_error*](#) is thrown, **this* remains unchanged.

Definition at line 598 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

5.377.3.3 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> template<typename _Ch_type, traits_type,
 typename _Allocator > basic_regex& std::basic_regex< _Ch_type,
 _Rx_traits >::assign (const basic_string< _Ch_type, _Ch_type_traits,
 _Allocator > & __s, flag_type __f = regex_constants::ECMAScript
) [inline]`

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

Parameters

- s* A string containing a regular expression pattern.
- flags* Syntax option flags.

Exceptions

- [*regex_error*](#) if *p* does not contain a valid regular expression pattern interpreted according to *flags*. If [*regex_error*](#) is thrown, **this* remains unchanged.

Definition at line 614 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

5.377.3.4 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> template<typename _InputIterator >
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (
_InputIterator __first, _InputIterator __last, flag_type __flags =
regex_constants::ECMAScript) [inline]`

Assigns a new regular expression to a regex object.

Parameters

first The start of a range containing a valid regular expression.

last The end of a range containing a valid regular expression.

flags Syntax option flags.

Exceptions

[*regex_error*](#) if p does not contain a valid regular expression pattern interpreted according to flags. If [*regex_error*](#) is thrown, the object remains unchanged.

Definition at line 637 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign().

5.377.3.5 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits >::assign (initializer_list< _Ch_type > __l,
flag_type __f= regex_constants::ECMAScript) [inline]`

Assigns a new regular expression to a regex object.

Parameters

l An initializer list representing a regular expression.

flags Syntax option flags.

Exceptions

[*regex_error*](#) if l does not contain a valid regular expression pattern interpreted according to flags. If [*regex_error*](#) is thrown, the object remains unchanged.

Definition at line 652 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign().

5.377.3.6 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::assign (basic_regex< _Ch_type, _Rx_traits
 > && __rhs) [inline]`

The move-assignment operator.

Parameters

rhs Another regular expression object.

Definition at line 559 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

5.377.3.7 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> basic_regex& std::basic_regex<
 _Ch_type, _Rx_traits >::assign (const _Ch_type * __p, flag_type
 __flags = regex_constants::ECMAScript) [inline]`

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

Parameters

p A pointer to a C-style null-terminated string containing a regular expression pattern.

flags Syntax option flags.

Exceptions

[*regex_error*](#) if *p* does not contain a valid regular expression pattern interpreted according to *flags*. If [*regex_error*](#) is thrown, *this remains unchanged.

Definition at line 580 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

5.377.3.8 `template<typename _Ch_type, typename _Rx_traits =
 regex_traits<_Ch_type>> flag_type std::basic_regex< _Ch_type,
 _Rx_traits >::flags () const [inline]`

Gets the flags used to construct the regular expression or in the last call to [`assign\(\)`](#).

Definition at line 670 of file regex.h.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::operator=()`.

5.377.3.9 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> locale_type std::basic_regex< _Ch_type,
_Rx_traits >::getloc () const [inline]`

Gets the locale currently imbued in the regular expression object.

Definition at line 688 of file regex.h.

5.377.3.10 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> locale_type std::basic_regex< _Ch_type,
_Rx_traits >::imbue (locale_type __loc) [inline]`

Imbues the regular expression object with the given locale.

Parameters

loc A locale.

Definition at line 680 of file regex.h.

5.377.3.11 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> unsigned int std::basic_regex<
_Ch_type, _Rx_traits >::mark_count () const [inline]`

Gets the number of marked subexpressions within the regular expression.

Definition at line 662 of file regex.h.

5.377.3.12 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits >::operator= (basic_regex< _Ch_type,
_Rx_traits > && __rhs) [inline]`

Move-assigns one regular expression to another.

Definition at line 514 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

5.377.3.13 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits >::operator= (const basic_regex< _Ch_type,
_Rx_traits > & __rhs) [inline]`

Assigns one regular expression to another.

Definition at line 507 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

5.377.3.14 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> template<typename _Ch_type, _Rx_traits,
typename _Allocator > basic_regex& std::basic_regex< _Ch_type,
_Rx_traits >::operator= (const basic_string< _Ch_type,
_Ch_type, _Allocator > & __s) [inline]`

Replaces a regular expression with a new one constructed from a string.

Parameters

A pointer to a string containing a regular expression.

Definition at line 536 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::flags()`.

5.377.3.15 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> basic_regex& std::basic_regex<
_Ch_type, _Rx_traits >::operator= (const _Ch_type * __p)
[inline]`

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

Parameters

A pointer to the start of a null-terminated C-style string containing a regular expression.

Definition at line 525 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::basic_regex< _Ch_type, _Rx_traits >::flags()`.

5.377.3.16 `template<typename _Ch_type, typename _Rx_traits =
regex_traits<_Ch_type>> void std::basic_regex< _Ch_type,
_Rx_traits >::swap (basic_regex< _Ch_type, _Rx_traits > &
__rhs) [inline]`

Swaps the contents of two regular expression objects.

Parameters

rhs Another regular expression object.

Definition at line 698 of file regex.h.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, and `std::swap()`.

The documentation for this class was generated from the following file:

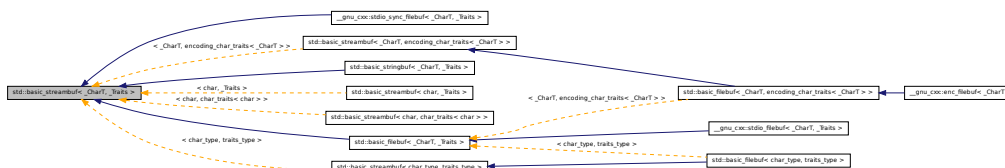
- `regex.h`

5.378 `std::basic_streambuf<_CharT, _Traits>` Class Template Reference

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Inheritance diagram for `std::basic_streambuf<_CharT, _Traits>`:



Public Types

- typedef _CharT **char_type**
- typedef _Traits **traits_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::pos_type **pos_type**
- typedef traits_type::off_type **off_type**
- typedef **basic_streambuf**< **char_type**, **traits_type** > **__streambuf_type**

Public Member Functions

- `streamsize in_avail ()`
- `int_type sbumpc ()`

- [int_type sgetc](#) ()
- [streamsize sgetn](#) ([char_type](#) *__s, [streamsize](#) __n)
- [int_type snextc](#) ()
- [int_type sputbackc](#) ([char_type](#) __c)
- [int_type sputc](#) ([char_type](#) __c)
- [streamsize sputn](#) (const [char_type](#) *__s, [streamsize](#) __n)
- void [stoss](#) ()
- [int_type sungetc](#) ()

Protected Member Functions

- [basic_streambuf](#) ()
 - void [gbump](#) (int __n)
 - virtual void [imbue](#) (const [locale](#) &)
 - virtual [int_type overflow](#) ([int_type](#)=[traits_type::eof](#)())
 - virtual [int_type pbackfail](#) ([int_type](#)=[traits_type::eof](#)())
 - void [pbump](#) (int __n)
 - virtual [pos_type seekoff](#) ([off_type](#), [ios_base::seekdir](#), [ios_base::openmode](#)=[ios_base::in](#)|[ios_base::out](#))
 - virtual [pos_type seekpos](#) ([pos_type](#), [ios_base::openmode](#)=[ios_base::in](#)|[ios_base::out](#))
 - virtual [basic_streambuf](#)<[char_type](#), [_Traits](#)> * [setbuf](#) ([char_type](#) *, [streamsize](#))
 - void [setg](#) ([char_type](#) *__gbeg, [char_type](#) *__gnext, [char_type](#) *__gend)
 - void [setp](#) ([char_type](#) *__pbeg, [char_type](#) *__pend)
 - virtual [streamsize showmanyc](#) ()
 - virtual int [sync](#) ()
 - virtual [int_type uflow](#) ()
 - virtual [int_type underflow](#) ()
 - virtual [streamsize xsgetn](#) ([char_type](#) *__s, [streamsize](#) __n)
 - virtual [streamsize xsputn](#) (const [char_type](#) *__s, [streamsize](#) __n)
-
- [char_type](#) * [eback](#) () const
 - [char_type](#) * [gptr](#) () const
 - [char_type](#) * [egptr](#) () const
-
- [char_type](#) * [pbase](#) () const
 - [char_type](#) * [pptr](#) () const
 - [char_type](#) * [epptr](#) () const

Friends

- template<bool _IsMove, typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)
 - streamsize __copy_streambufs_eof (__streambuf_type *, __streambuf_type *, bool &)
 - class basic_ios< char_type, traits_type >
 - class basic_istream< char_type, traits_type >
 - class basic_ostream< char_type, traits_type >
 - template<typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)
 - template<typename _CharT2, typename _Traits2, typename _Alloc >
basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)
 - class istreambuf_iterator< char_type, traits_type >
 - template<typename _CharT2, typename _Traits2, typename _Alloc >
basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)
 - template<typename _CharT2, typename _Traits2 >
basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2, _Traits2 > &, _CharT2 *)
 - class ostreambuf_iterator< char_type, traits_type >
-
- char_type * _M_in_beg
 - char_type * _M_in_cur
 - char_type * _M_in_end
 - char_type * _M_out_beg
 - char_type * _M_out_cur
 - char_type * _M_out_end
 - locale _M_buf_locale
 - virtual ~basic_streambuf ()
 - locale pubimbue (const locale &__loc)
 - locale getloc () const
 - __streambuf_type * pubsetbuf (char_type *__s, streamsize __n)
 - pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
 - pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
 - int pubsync ()

5.378.1 Detailed Description

```
template<typename _CharT, typename _Traits> class std::basic_streambuf< _  
_CharT, _Traits >
```

The actual work of input and output (interface).

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output. Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:
 - The controlled input sequence can be not readable.
 - The controlled output sequence can be not writable.
 - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
 - The controlled sequences can support operations *directly* to or from associated sequences.
 - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.
2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
 - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
 - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
 - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
 - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.

- If *xnext* is not a null pointer and *xnext* < *xend* for an output sequence, then a *write position* is available. In this case, **xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
- If *xnext* is not a null pointer and *xbeg* < *xnext* for an input sequence, then a *putback position* is available. In this case, *xnext*[-1] shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
- If *xnext* is not a null pointer and *xnext* < *xend* for an input sequence, then a *read position* is available. In this case, **xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

Definition at line 115 of file `streambuf`.

5.378.2 Member Typedef Documentation

5.378.2.1 `template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_streambuf<_CharT, _Traits >::__streambuf_type`

This is a non-standard type.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 133 of file `streambuf`.

5.378.2.2 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_streambuf< _CharT, _Traits >::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 124 of file `streambuf`.

5.378.2.3 `template<typename _CharT, typename _Traits> typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 126 of file streambuf.

5.378.2.4 `template<typename _CharT, typename _Traits> typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 128 of file streambuf.

5.378.2.5 `template<typename _CharT, typename _Traits> typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::enc_filebuf< _CharT >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 127 of file streambuf.

5.378.2.6 `template<typename _CharT, typename _Traits> typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::enc_filebuf< _CharT >`, `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 125 of file streambuf.

5.378.3 Constructor & Destructor Documentation

5.378.3.1 `template<typename _CharT, typename _Traits> virtual std::basic_streambuf< _CharT, _Traits >::~~basic_streambuf () [inline, virtual]`

Destructor deallocates no buffer space.

Definition at line 193 of file streambuf.

5.378.3.2 `template<typename _CharT, typename _Traits> std::basic_streambuf< _CharT, _Traits >::basic_streambuf () [inline, protected]`

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the [basic_streambuf](#) class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

Definition at line 441 of file streambuf.

5.378.4 Member Function Documentation

5.378.4.1 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eback () const
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.378.4.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::egptr () const
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.378.4.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::epptr () const
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by std::basic_streambuf< _CharT, _Traits >::xsputn().

5.378.4.4 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::gbump (int __n)
[inline, protected]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

5.378.4.5 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::getloc () const
[inline]`

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent loc is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

5.378.4.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::gptr () const
[inline, protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.378.4.7 `template<typename _CharT, typename _Traits> virtual void
std::basic_streambuf< _CharT, _Traits >::imbue (const locale &)
[inline, protected, virtual]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 554 of file streambuf.

5.378.4.8 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::in_avail () [inline]`

Looking ahead into the stream.

Returns

The number of characters available.

5.378 std::basic_streambuf<_CharT, _Traits> Class Template Reference 2029

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 262 of file streambuf.

5.378.4.9 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::overflow (int_type =
traits_type::eof()) [inline, protected, virtual]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

c An additional character to consume.

Returns

`eof()` to indicate failure, something else (usually *c*, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns `eof()`.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 746 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xsputn()`.

5.378.4.10 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail (int_type =
traits_type::eof()) [inline, protected, virtual]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 702 of file streambuf.

5.378.4.11 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::pbase () const
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits>::sync()`.

5.378.4.12 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::pbump (int __n)
[inline, protected]`

Moving the write position.

Parameters

n The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf<_CharT, _Traits>::xspn()`.

5.378.4.13 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pptr () const
[inline, protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.378.4.14 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file streambuf.

5.378.4.15 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file `streambuf`.

```
5.378.4.16  template<typename _CharT, typename _Traits> pos_type
             std::basic_streambuf< _CharT, _Traits >::pubseekpos ( pos_type
             __sp, ios_base::openmode __mode = ios_base::in | ios_base::out )
             [inline]
```

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

```
5.378.4.17  template<typename _CharT, typename _Traits>
             __streambuf_type* std::basic_streambuf< _CharT, _Traits
             >::pubsetbuf ( char_type * __s, streamsize __n ) [inline]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

```
5.378.4.18  template<typename _CharT, typename _Traits> int
             std::basic_streambuf< _CharT, _Traits >::pubsync ( )
             [inline]
```

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived imbue(loc).

Definition at line 249 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::sync().

5.378.4.19 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sbumpc() [inline]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::istreambuf_iterator< _CharT, _Traits >::operator++().

5.378.4.20 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf< _CharT, _Traits >::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file streambuf.

5.378.4.21 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf< _CharT, _Traits >::seekpos (pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 592 of file `streambuf`.

5.378.4.22 `template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type,_Traits>* std::basic_streambuf<
_CharT, _Traits >::setbuf (char_type *, streamsize) [inline,
protected, virtual]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns `this`.

Definition at line 569 of file `streambuf`.

5.378.4.23 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setg (char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline,
protected]`

Setting the three read area pointers.

Parameters

gbeg A pointer.

gnext A pointer.

gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

5.378.4.24 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setp (char_type *
__pbeg, char_type * __pend) [inline, protected]`

Setting the three write area pointers.

Parameters

pbeg A pointer.

pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file streambuf.

5.378.4.25 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sgetc () [inline]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

5.378.4.26 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::sgetn (char_type * __s,
streamsize __n) [inline]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file `streambuf`.

5.378.4.27 `template<typename _CharT, typename _Traits> virtual streamsize
std::basic_streambuf< _CharT, _Traits >::showmanyc ()
[inline, protected, virtual]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
[27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 627 of file `streambuf`.

5.378.4.28 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::snextc () [inline]`

Getting the next character.

Returns

The next character, or eof.

5.378 std::basic_streambuf<_CharT, _Traits> Class Template Reference 2037

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 276 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.378.4.29 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputbackc (char_type
__c) [inline]`

Pushing characters back into the input stream.

Parameters

`c` The character to push back.

Returns

The previous character, if possible.

Similar to `sungetc()`, but `c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `c`.

Definition at line 350 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

5.378.4.30 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputc (char_type __c)
[inline]`

Entry point for all single-character output functions.

Parameters

`c` A character to output.

Returns

`c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `c` in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file streambuf.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, and `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

5.378.4.31 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::sputn (const char_type
* __s, streamsize __n) [inline]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file streambuf.

5.378.4.32 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::stossc () [inline]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file streambuf.

5.378.4.33 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sungetc () [inline]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `ebackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 375 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

5.378.4.34 `template<typename _CharT, typename _Traits> virtual int
std::basic_streambuf< _CharT, _Traits >::sync (void)
[inline, protected, virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 605 of file streambuf.

5.378.4.35 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::uflow () [inline,
protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#).

Definition at line 678 of file streambuf.

5.378.4.36 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::underflow ()
[inline, protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 665 of file streambuf.

5.378.4.37 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf< _CharT, _Traits >::xsgetn (char_type *
__s, streamsize __n) [protected, virtual]`

Multiple character extraction.

Parameters

s A buffer area.

n Maximum number of characters to assign.

Returns

The number of characters assigned.

5.378 std::basic_streambuf<_CharT, _Traits> Class Template Reference 2041

Fills $s[0]$ through $s[n-1]$ with characters from the input sequence, as if by `sbumpc()`. Stops when either n characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

```
5.378.4.38  template<typename _CharT, typename _Traits> streamsize
               std::basic_streambuf<_CharT, _Traits>::xsputn( const char_type
               * __s, streamsize __n ) [protected, virtual]
```

Multiple character insertion.

Parameters

s A buffer area.

n Maximum number of characters to write.

Returns

The number of characters written.

Writes $s[0]$ through $s[n-1]$ to the output sequence, as if by `sputc()`. Stops when either n characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.378.5 Member Data Documentation

```
5.378.5.1  template<typename _CharT, typename _Traits> locale
               std::basic_streambuf<_CharT, _Traits>::_M_buf_locale
               [protected]
```

Current locale setting.

Definition at line 188 of file streambuf.

Referenced by `std::basic_filebuf<_CharT, _Traits >::basic_filebuf()`.

5.378.5.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits >::_M_in_beg
[protected]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get == input == read`
- `put == output == write`

Definition at line 180 of file streambuf.

5.378.5.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits >::_M_in_cur
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file streambuf.

5.378.5.4 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits >::_M_in_end
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.378.5.5 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_beg
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

5.378.5.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_cur
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

5.378.5.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

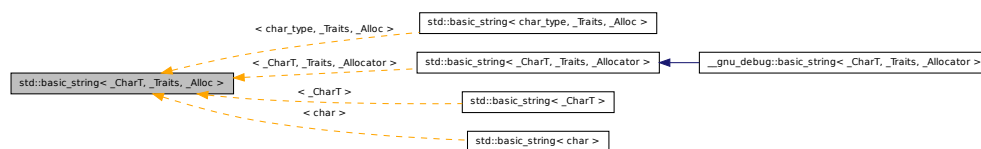
The documentation for this class was generated from the following files:

- [streambuf](#)
- [streambuf.tcc](#)

5.379 `std::basic_string< _CharT, _Traits, _Alloc >` Class Template Reference

Managing sequences of characters and character-like objects.

Inheritance diagram for `std::basic_string< _CharT, _Traits, _Alloc >`:

**Public Types**

- typedef `_Alloc` **allocator_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic_string >` **const_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const_pointer**
- typedef `_CharT_alloc_type::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**

- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_CharT_alloc_type::size_type` **size_type**
- typedef `_Traits` **traits_type**
- typedef `_Traits::char_type` **value_type**

Public Member Functions

- `basic_string()`
- `basic_string(const _Alloc &__a)`
- `basic_string(const basic_string &__str, size_type __pos, size_type __n=npos)`
- `basic_string(size_type __n, _CharT __c, const _Alloc &__a=_Alloc())`
- `basic_string(basic_string &&__str)`
- `basic_string(const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)`
- `basic_string(initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())`
- template<class _InputIterator >
`basic_string(_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())`
- `basic_string(const basic_string &__str)`
- `basic_string(const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())`
- `basic_string(const _CharT *__s, const _Alloc &__a=_Alloc())`
- `~basic_string()`
- template<typename _InIterator >
`_CharT * _S_construct(_InIterator __beg, _InIterator __end, const _Alloc &__a, forward_iterator_tag)`
- `basic_string & append(const basic_string &__str)`
- `basic_string & append(const basic_string &__str, size_type __pos, size_type __n)`
- `basic_string & append(const _CharT *__s, size_type __n)`
- `basic_string & append(const _CharT *__s)`
- `basic_string & append(size_type __n, _CharT __c)`
- `basic_string & append(initializer_list< _CharT > __l)`
- template<class _InputIterator >
`basic_string & append(_InputIterator __first, _InputIterator __last)`
- `basic_string & assign(const _CharT *__s)`
- `basic_string & assign(size_type __n, _CharT __c)`
- template<class _InputIterator >
`basic_string & assign(_InputIterator __first, _InputIterator __last)`
- `basic_string & assign(initializer_list< _CharT > __l)`
- `basic_string & assign(const basic_string &__str)`
- `basic_string & assign(basic_string &&__str)`
- `basic_string & assign(const basic_string &__str, size_type __pos, size_type __n)`

- [basic_string](#) & [assign](#) (const _CharT *__s, size_type __n)
- const_reference [at](#) (size_type __n) const
- reference [at](#) (size_type __n)
- reference [back](#) ()
- const_reference [back](#) () const
- iterator [begin](#) ()
- const_iterator [begin](#) () const
- const _CharT * [c_str](#) () const
- size_type [capacity](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [cend](#) () const
- void [clear](#) ()
- int [compare](#) (size_type __pos, size_type __n1, const _CharT *__s) const
- int [compare](#) (const _CharT *__s) const
- int [compare](#) (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- int [compare](#) (size_type __pos, size_type __n, const [basic_string](#) &__str) const
- int [compare](#) (size_type __pos1, size_type __n1, const [basic_string](#) &__str, size_type __pos2, size_type __n2) const
- int [compare](#) (const [basic_string](#) &__str) const
- size_type [copy](#) (_CharT *__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator [crbegin](#) () const
- const_reverse_iterator [crend](#) () const
- const _CharT * [data](#) () const
- bool [empty](#) () const
- iterator [end](#) ()
- const_iterator [end](#) () const
- [basic_string](#) & [erase](#) (size_type __pos=0, size_type __n=[npos](#))
- iterator [erase](#) (iterator __position)
- iterator [erase](#) (iterator __first, iterator __last)
- size_type [find](#) (_CharT __c, size_type __pos=0) const
- size_type [find](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find](#) (const [basic_string](#) &__str, size_type __pos=0) const
- size_type [find](#) (const _CharT *__s, size_type __pos=0) const
- size_type [find_first_not_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find_first_not_of](#) (_CharT __c, size_type __pos=0) const
- size_type [find_first_not_of](#) (const [basic_string](#) &__str, size_type __pos=0) const
- size_type [find_first_not_of](#) (const _CharT *__s, size_type __pos=0) const
- size_type [find_first_of](#) (_CharT __c, size_type __pos=0) const
- size_type [find_first_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find_first_of](#) (const _CharT *__s, size_type __pos=0) const

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2047

- size_type [find_first_of](#) (const [basic_string](#) &__str, size_type __pos=0) const
- size_type [find_last_not_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [find_last_not_of](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type [find_last_not_of](#) (const [basic_string](#) &__str, size_type __pos=[npos](#)) const
- size_type [find_last_not_of](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const [basic_string](#) &__str, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- size_type [find_last_of](#) (const _CharT *__s, size_type __pos, size_type __n) const
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const
- void [insert](#) (iterator __p, size_type __n, _CharT __c)
- template<class _InputIterator>
void [insert](#) (iterator __p, _InputIterator __beg, _InputIterator __end)
- void [insert](#) (iterator __p, [initializer_list](#)<_CharT> __l)
- [basic_string](#) & [insert](#) (size_type __pos1, const [basic_string](#) &__str, size_type __pos2, size_type __n)
- [basic_string](#) & [insert](#) (size_type __pos, const _CharT *__s, size_type __n)
- [basic_string](#) & [insert](#) (size_type __pos, const _CharT *__s)
- iterator [insert](#) (iterator __p, _CharT __c)
- [basic_string](#) & [insert](#) (size_type __pos, size_type __n, _CharT __c)
- [basic_string](#) & [insert](#) (size_type __pos1, const [basic_string](#) &__str)
- size_type [length](#) () const
- size_type [max_size](#) () const
- [basic_string](#) & [operator+=](#) ([initializer_list](#)<_CharT> __l)
- [basic_string](#) & [operator+=](#) (const [basic_string](#) &__str)
- [basic_string](#) & [operator+=](#) (_CharT __c)
- [basic_string](#) & [operator+=](#) (const _CharT *__s)
- [basic_string](#) & [operator=](#) (const _CharT *__s)
- [basic_string](#) & [operator=](#) (_CharT __c)
- [basic_string](#) & [operator=](#) ([initializer_list](#)<_CharT> __l)
- [basic_string](#) & [operator=](#) ([basic_string](#) &&__str)
- [basic_string](#) & [operator=](#) (const [basic_string](#) &__str)
- const_reference [operator\[\]](#) (size_type __pos) const
- reference [operator\[\]](#) (size_type __pos)
- void [push_back](#) (_CharT __c)
- [reverse_iterator](#) [rbegin](#) ()
- const [reverse_iterator](#) [rbegin](#) () const
- [reverse_iterator](#) [rend](#) ()

- [const_reverse_iterator rend](#) () const
- [basic_string & replace](#) (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- [basic_string & replace](#) (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const [basic_string](#) &__str)
- template<class _InputIterator >
[basic_string & replace](#) (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- [basic_string & replace](#) (size_type __pos, size_type __n, const [basic_string](#) &__str)
- [basic_string & replace](#) (size_type __pos, size_type __n1, const _CharT *__s)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)
- [basic_string & replace](#) (size_type __pos1, size_type __n1, const [basic_string](#) &__str, size_type __pos2, size_type __n2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const _CharT *__s)
- [basic_string & replace](#) (iterator __i1, iterator __i2, [initializer_list](#)< _CharT > __l)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)
- [basic_string & replace](#) (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)
- void [reserve](#) (size_type __res_arg=0)
- void [resize](#) (size_type __n)
- void [resize](#) (size_type __n, _CharT __c)
- size_type [rfind](#) (_CharT __c, size_type __pos=[npos](#)) const
- size_type [rfind](#) (const _CharT *__s, size_type __pos, size_type __n) const
- size_type [rfind](#) (const [basic_string](#) &__str, size_type __pos=[npos](#)) const
- size_type [rfind](#) (const _CharT *__s, size_type __pos=[npos](#)) const
- void [shrink_to_fit](#) ()
- size_type [size](#) () const
- [basic_string substr](#) (size_type __pos=0, size_type __n=[npos](#)) const
- void [swap](#) ([basic_string](#) &__s)

Static Public Attributes

- static const size_type [npos](#)

5.379.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_string< _CharT, _Traits, _Alloc >`

Managing sequences of characters and character-like objects. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and array access are supported.

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Documentation? What's that? Nathan Myers <ncm@cantrip.org>.

A string looks like this:

	<code>[_Rep]</code>
	<code>_M_length</code>
<code>[basic_string<char_type>]</code>	<code>_M_capacity</code>
<code>_M_dataplus</code>	<code>_M_refcount</code>
<code>_M_p -----></code>	<code>unnamed array of char_type</code>

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_data()`, and `string::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 105 of file `basic_string.h`.

5.379.2 Constructor & Destructor Documentation

5.379.2.1 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string ()
[inline]`

Default constructor creates an empty string.

Definition at line 430 of file `basic_string.h`.

5.379.2.2 `template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const
_Alloc & __a) [explicit]`

Construct an empty string using allocator *a*.

Definition at line 178 of file `basic_string.tcc`.

5.379.2.3 `template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const
basic_string< _CharT, _Traits, _Alloc > & __str)`

Construct string with copy of value of *str*.

Parameters

str Source string.

Definition at line 170 of file `basic_string.tcc`.

5.379.2.4 `template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const
basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos,
size_type __n = npos)`

Construct string as copy of a substring.

Parameters

str Source string.

pos Index of first character to copy from.

n Number of characters to copy (default remainder).

Definition at line 184 of file `basic_string.tcc`.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2051

5.379.2.5 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const
basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos,
size_type __n, const _Alloc & __a)`

Construct string as copy of a substring.

Parameters

- str* Source string.
- pos* Index of first character to copy from.
- n* Number of characters to copy.
- a* Allocator to use.

Definition at line 194 of file basic_string.tcc.

5.379.2.6 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const
_CharT * __s, size_type __n, const _Alloc & __a = _Alloc())`

Construct string initialized by a character array.

Parameters

- s* Source character array.
- n* Number of characters to copy.
- a* Allocator to use (default is default allocator).

NB: *s* must have at least *n* characters, `'\0'` has no special meaning.

Definition at line 206 of file basic_string.tcc.

5.379.2.7 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const
_CharT * __s, const _Alloc & __a = _Alloc())`

Construct string as copy of a C string.

Parameters

- s* Source C string.
- a* Allocator to use (default is default allocator).

Definition at line 213 of file basic_string.tcc.

5.379.2.8 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
size_type __n, _CharT __c, const _Alloc & __a = _Alloc())`

Construct string as multiple characters.

Parameters

- n* Number of characters.
- c* Character to use.
- a* Allocator to use (default is default allocator).

Definition at line 220 of file basic_string.tcc.

5.379.2.9 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
basic_string< _CharT, _Traits, _Alloc > && __str) [inline]`

Move construct string.

Parameters

- str* Source string.

The newly-created string contains the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 500 of file basic_string.h.

5.379.2.10 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
initializer_list< _CharT > __l, const _Alloc & __a = _Alloc())`

Construct string from an initializer list.

Parameters

- l* `std::initializer_list` of characters.
- a* Allocator to use (default is default allocator).

Definition at line 235 of file basic_string.tcc.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2053

5.379.2.11 `template<typename _CharT , typename _Traits , typename
_Alloc> template<typename _InputIterator > std::basic_string<
_CharT, _Traits, _Alloc >::basic_string (_InputIterator __beg,
_InputIterator __end, const _Alloc & __a = _Alloc())`

Construct string as copy of a range.

Parameters

beg Start of range.

end End of range.

a Allocator to use (default is default allocator).

Definition at line 228 of file basic_string.tcc.

5.379.2.12 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~~basic_string ()
[inline]`

Destroy the string instance.

Definition at line 531 of file basic_string.h.

5.379.3 Member Function Documentation

5.379.3.1 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::append (const basic_string< _CharT,
_Traits, _Alloc > & __str, size_type __pos, size_type __n)`

Append a substring.

Parameters

str The string to append.

pos Index of the first character of str to append.

n The number of characters to append.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) if *pos* is not a valid index.

This function appends n characters from *str* starting at *pos* to this string. If n is larger than the number of available characters in *str*, the remainder of *str* is appended.

Definition at line 342 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.379.3.2 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<
_CharT, _Traits, _Alloc>::append (const _CharT * __s, size_type
__n)`

Append a C substring.

Parameters

- s The C string to append.
- n The number of characters to append.

Returns

Reference to this string.

Definition at line 298 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.379.3.3 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::append
(const _CharT * __s) [inline]`

Append a C string.

Parameters

- s The C string to append.

Returns

Reference to this string.

Definition at line 995 of file `basic_string.h`.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

5.379.3.4 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::append (size_type __n, _CharT __c)`

Append multiple characters.

Parameters

n The number of characters to append.

c The character to use.

Returns

Reference to this string.

Appends *n* copies of *c* to this string.

Definition at line 281 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

5.379.3.5 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append
(initializer_list< _CharT > __l) [inline]`

Append an [initializer_list](#) of characters.

Parameters

l The [initializer_list](#) of characters to append.

Returns

Reference to this string.

Definition at line 1019 of file `basic_string.h`.

Referenced by `std::basic_string< char >::append()`.

5.379.3.6 `template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator > basic_string& std::basic_string<
_CharT, _Traits, _Alloc >::append (_InputIterator __first,
_InputIterator __last) [inline]`

Append a range of characters.

Parameters

first Iterator referencing the first character to append.

last Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 1033 of file basic_string.h.

5.379.3.7 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::append (const basic_string< _CharT,
_Traits, _Alloc > & __str)`

Append a string to this string.

Parameters

str The string to append.

Returns

Reference to this string.

Definition at line 325 of file basic_string.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

Referenced by `std::collate< _CharT >::do_transform()`, `std::operator+()`, `std::operator>>()`, and `std::basic_string< _CharT, _Traits, _Alloc >::resize()`.

5.379.3.8 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
basic_string< _CharT, _Traits, _Alloc > && __str) [inline]`

Set value to contents of another string.

Parameters

str Source string to use.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2057

Returns

Reference to this string.

This function sets this string to the exact contents of *str*. *str* is a valid, but unspecified string.

Definition at line 1068 of file basic_string.h.

```
5.379.3.9 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::assign (
const basic_string<_CharT, _Traits, _Alloc> & __str, size_type
__pos, size_type __n ) [inline]
```

Set value to a substring of a string.

Parameters

str The string to use.

pos Index of the first character of *str*.

n Number of characters to use.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) if *pos* is not a valid index.

This function sets this string to the substring of *str* consisting of *n* characters at *pos*. If *n* is larger than the number of available characters in *str*, the remainder of *str* is used.

Definition at line 1088 of file basic_string.h.

Referenced by std::basic_string<char>::assign().

```
5.379.3.10 template<typename _CharT, typename _Traits, typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<
_CharT, _Traits, _Alloc>::assign ( const _CharT * __s, size_type
__n )
```

Set value to a C substring.

Parameters

s The C string to use.

n Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first *n* characters of *s*. If *n* is larger than the number of available characters in *s*, the remainder of *s* is used.

Definition at line 259 of file basic_string.tcc.

References std::basic_string<_CharT, _Traits, _Alloc >::size().

5.379.3.11 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign
(const _CharT * __s) [inline]`

Set value to contents of a C string.

Parameters

s The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of *s*. The data is copied, so there is no dependence on *s* once the function returns.

Definition at line 1116 of file basic_string.h.

5.379.3.12 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign
(size_type __n, _CharT __c) [inline]`

Set value to multiple characters.

Parameters

n Length of the resulting string.

c The character to use.

Returns

Reference to this string.

This function sets the value of this string to *n* copies of character *c*.

Definition at line 1132 of file basic_string.h.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2059

5.379.3.13 `template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator > basic_string& std::basic_string<
_CharT, _Traits, _Alloc >::assign (_InputIterator __first,
_InputIterator __last) [inline]`

Set value to a range of characters.

Parameters

first Iterator referencing the first character to append.

last Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 1145 of file basic_string.h.

5.379.3.14 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign
(initializer_list< _CharT > __l) [inline]`

Set value to an [initializer_list](#) of characters.

Parameters

l The [initializer_list](#) of characters to assign.

Returns

Reference to this string.

Definition at line 1155 of file basic_string.h.

Referenced by std::basic_string< char >::assign().

5.379.3.15 `template<typename _CharT , typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::assign (const basic_string< _CharT,
_Traits, _Alloc > & __str)`

Set value to contents of another string.

Parameters

str Source string to use.

Returns

Reference to this string.

Definition at line 243 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::get_allocator()`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::str()`.

5.379.3.16 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::at (
size_type __n) const [inline]`

Provides access to the data contained in the string.

Parameters

n The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws [*out_of_range*](#) if the check fails.

Definition at line 854 of file `basic_string.h`.

5.379.3.17 `template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::at (
size_type __n) [inline]`

Provides access to the data contained in the string.

Parameters

n The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws [out_of_range](#) if the check fails. Success results in unsharing the string.

Definition at line 907 of file `basic_string.h`.

5.379.3.18 `template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back ()
[inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 883 of file `basic_string.h`.

5.379.3.19 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back
() const [inline]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 891 of file `basic_string.h`.

5.379.3.20 `template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ()
[inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 598 of file `basic_string.h`.

Referenced by `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

5.379.3.21 `template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 609 of file `basic_string.h`.

5.379.3.22 `template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::c_str (
) const [inline]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1764 of file basic_string.h.

Referenced by `std::collate< _CharT >::do_compare()`, `std::money_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::collate< _CharT >::do_transform()`, and `std::basic_filebuf< char_type, traits_type >::open()`.

5.379.3.23 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity (
) const [inline]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 766 of file basic_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, and `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`.

5.379.3.24 `template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin (
) const [inline]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 673 of file basic_string.h.

5.379.3.25 `template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend (
) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 681 of file basic_string.h.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2063

5.379.3.26 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::clear ()
[inline]`

Erases the string, making it empty.

Definition at line 793 of file basic_string.h.

Referenced by std::basic_stringbuf<_CharT, _Traits, _Alloc>::setbuf().

5.379.3.27 `template<typename _CharT, typename _Traits, typename _Alloc
> int std::basic_string<_CharT, _Traits, _Alloc>::compare (
size_type __pos, size_type __n1, const _CharT * __s, size_type
__n2) const`

Compare substring against a character array.

Parameters

pos1 Index of first character of substring.

n1 Number of characters in substring.

s character array to compare against.

n2 Number of characters of *s*.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form a string from the first *n2* characters of *s*. Returns an integer < 0 if this substring is ordered before the string from *s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *s*. Determines the effective length *r1* of the strings to compare as the smallest of the length of the substring and *n2*. The function then compares the two strings by calling traits::compare(substring.data(),s,r1). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: *s* must have at least *n2* characters, '\0' has no special meaning.

Definition at line 980 of file basic_string.tcc.

References std::min().

5.379.3.28 `template<typename _CharT, typename _Traits, typename _Alloc
> int std::basic_string<_CharT, _Traits, _Alloc>::compare (
size_type __pos, size_type __n, const basic_string<_CharT,
_Traits, _Alloc> & __str) const`

Compare substring to a string.

Parameters

pos Index of first character of substring.

n Number of characters in substring.

str String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and *str.size()*. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 916 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::compare()`, `std::basic_string<_CharT, _Traits, _Alloc >::data()`, `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc >::size()`.

5.379.3.29 `template<typename _CharT, typename _Traits, typename _Alloc
> int std::basic_string<_CharT, _Traits, _Alloc >::compare (
size_type __pos1, size_type __n1, const basic_string<_CharT,
_Traits, _Alloc > & __str, size_type __pos2, size_type __n2) const`

Compare substring to a substring.

Parameters

pos1 Index of first character of substring.

n1 Number of characters in substring.

str String to compare against.

pos2 Index of first character of substring of *str*.

n2 Number of characters in substring of *str*.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *n1* characters starting at *pos1*. Form the substring of *str* from the *n2* characters starting at *pos2*. Returns an integer < 0 if this substring is ordered before the substring of *str*, 0 if their values are

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2065

equivalent, or > 0 if this substring is ordered after the substring of *str*. Determines the effective length *rlen* of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 931 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::min()`.

```
5.379.3.30  template<typename _CharT, typename _Traits, typename _Alloc>
               int std::basic_string< _CharT, _Traits, _Alloc >::compare (
               const basic_string< _CharT, _Traits, _Alloc > & __str ) const
               [inline]
```

Compare to a string.

Parameters

str String to compare against.

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before *str*, 0 if their values are equivalent, or > 0 if this string is ordered after *str*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2171 of file `basic_string.h`.

Referenced by `std::sub_match< _Bi_iter >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

```
5.379.3.31  template<typename _CharT, typename _Traits , typename _Alloc
               > int std::basic_string< _CharT, _Traits, _Alloc >::compare (
               size_type __pos, size_type __n1, const _CharT * __s ) const
```

Compare substring to a C string.

Parameters

pos Index of first character of substring.

nl Number of characters in substring.

s C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *nl* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *s*. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 964 of file `basic_string.tcc`.

References `std::min()`.

5.379.3.32 `template<typename _CharT, typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (const
_CharT * __s) const`

Compare to a C string.

Parameters

s C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *s*, 0 if their values are equivalent, or > 0 if this string is ordered after *s*. Determines the effective length *rlen* of the strings to compare as the smallest of `size()` and the length of a string constructed from *s*. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 949 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::length()`, `std::min()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference

5.379.3.33 `template<typename _CharT, typename _Traits, typename
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::copy (_CharT * __s,
size_type __n, size_type __pos = 0) const`

Copy substring into C string.

Parameters

- s* C string to copy value into.
- n* Number of characters to copy.
- pos* Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

[*std::out_of_range*](#) If *pos* > *size()*.

Copies up to *n* characters starting at *pos* into the C string *s*. If *pos* is greater than *size()*, *out_of_range* is thrown.

Definition at line 723 of file `basic_string.tcc`.

5.379.3.34 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc
>::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 690 of file `basic_string.h`.

5.379.3.35 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc
>::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 699 of file `basic_string.h`.

5.379.3.36 `template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::data (
) const [inline]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1774 of file basic_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `std::collate< _CharT >::do_compare()`, `std::collate< _CharT >::do_transform()`, `std::basic_string< char >::find()`, `std::basic_string< char >::find_first_not_of()`, `std::basic_string< char >::find_first_of()`, `std::basic_string< char >::find_last_not_of()`, `std::basic_string< char >::find_last_of()`, `std::basic_string< char >::rfind()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`, and `std::regex_traits< _Ch_type >::transform()`.

5.379.3.37 `template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string< _CharT, _Traits, _Alloc >::empty () const
[inline]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 801 of file basic_string.h.

Referenced by `std::operator>>()`.

5.379.3.38 `template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::end ()
[inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 617 of file basic_string.h.

Referenced by `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

5.379.3.39 `template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end (
) const [inline]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 628 of file basic_string.h.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2069

5.379.3.40 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::erase (
size_type __pos = 0, size_type __n = npos) [inline]`

Remove characters.

Parameters

- pos* Index of first character to remove (default 0).
n Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Removes *n* characters from this string starting at *pos*. The length of the string is reduced by *n*. If there are < *n* characters to remove, the remainder of the string is truncated. If *p* is beyond end of string, [*out_of_range*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1345 of file basic_string.h.

Referenced by `std::getline()`, `std::operator>>()`, and `std::basic_string< _CharT, _Traits, _Alloc >::resize()`.

5.379.3.41 `template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
iterator __position) [inline]`

Remove one character.

Parameters

- position* Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1361 of file basic_string.h.

5.379.3.42 `template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::iterator
std::basic_string< _CharT, _Traits, _Alloc >::erase (iterator
__first, iterator __last)`

Remove a range of characters.

Parameters

first Iterator referencing the first character to remove.

last Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 391 of file basic_string.tcc.

5.379.3.43 `template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::find (const _CharT *
__s, size_type __pos, size_type __n) const`

Find position of a C substring.

Parameters

s C string to locate.

pos Index of character to search from.

n Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 737 of file basic_string.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::find()`, and `std::basic_string< _CharT, _Traits, _Alloc >::find_first_of()`.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2071

5.379.3.44 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (const
basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos =
0) const [inline]`

Find position of a string.

Parameters

str String to locate.

pos Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for value of *str* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1809 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find()`.

5.379.3.45 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find (const
_CharT * __s, size_type __pos = 0) const [inline]`

Find position of a C string.

Parameters

s C string to locate.

pos Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *pos*, searches forward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1823 of file `basic_string.h`.

5.379.3.46 `template<typename _CharT, typename _Traits, typename
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::find (_CharT __c,
size_type __pos = 0) const`

Find position of a character.

Parameters

c Character to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 760 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::find()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.379.3.47 `template<typename _CharT, typename _Traits, typename
_Alloc> basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of(
_CharT __c, size_type __pos = 0) const`

Find position of a different character.

Parameters

c Character to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 864 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.379.3.48 `template<typename _CharT, typename _Traits, typename
_Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc
>::find_first_not_of(const basic_string<_CharT, _Traits, _Alloc
> & __str, size_type __pos = 0) const [inline]`

Find position of a character not in string.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2073

Parameters

str String containing characters to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2033 of file basic_string.h.

Referenced by std::basic_string< char >::find_first_not_of().

```
5.379.3.49  template<typename _CharT, typename _Traits , typename
             _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
             std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
             const _CharT * __s, size_type __pos, size_type __n ) const
```

Find position of a character not in C substring.

Parameters

s C string containing characters to avoid.

pos Index of character to search from.

n Number of characters from *s* to consider.

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 852 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

```
5.379.3.50  template<typename _CharT, typename _Traits, typename
             _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc
             >::find_first_not_of ( const _CharT * __s, size_type __pos = 0 )
             const [inline]
```

Find position of a character not in C string.

Parameters

s C string containing characters to avoid.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2062 of file basic_string.h.

```
5.379.3.51  template<typename _CharT, typename _Traits, typename _Alloc>
              size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of
              ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type
                __pos = 0 ) const  [inline]
```

Find position of a character of string.

Parameters

str String containing characters to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1911 of file basic_string.h.

Referenced by std::basic_string< char >::find_first_of().

```
5.379.3.52  template<typename _CharT, typename _Traits , typename
              _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
              std::basic_string< _CharT, _Traits, _Alloc >::find_first_of ( const
              _CharT * __s, size_type __pos, size_type __n ) const
```

Find position of a character of C substring.

Parameters

s String containing characters to locate.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2075

pos Index of character to search from.

n Number of characters from *s* to search for.

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 816 of file basic_string.tcc.

References std::basic_string<_CharT, _Traits, _Alloc>::find(), and std::basic_string<_CharT, _Traits, _Alloc>::size().

5.379.3.53 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of
(const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character of C string.

Parameters

s String containing characters to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 1939 of file basic_string.h.

5.379.3.54 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of
(_CharT __c, size_type __pos = 0) const [inline]`

Find position of a character.

Parameters

c Character to locate.

pos Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *pos*, searches forward for the character *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `find(c, pos)`.

Definition at line 1958 of file `basic_string.h`.

```
5.379.3.55  template<typename _CharT, typename _Traits, typename
              _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
              std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
              const _CharT * __s, size_type __pos, size_type __n ) const
```

Find last position of a character not in C substring.

Parameters

s C string containing characters to avoid.

pos Index of character to search back from.

n Number of characters from *s* to consider.

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 875 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

```
5.379.3.56  template<typename _CharT, typename _Traits, typename
              _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc
              >::find_last_not_of ( const _CharT * __s, size_type __pos = npow )
              const [inline]
```

Find last position of a character not in C string.

Parameters

s C string containing characters to avoid.

pos Index of character to search back from (default end).

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2077

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2121 of file basic_string.h.

5.379.3.57 `template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
_CharT __c, size_type __pos = npos) const`

Find last position of a different character.

Parameters

c Character to avoid.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character other than *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 896 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

5.379.3.58 `template<typename _CharT, typename _Traits, typename
_Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (const basic_string< _CharT, _Traits, _Alloc >
& __str, size_type __pos = npos) const [inline]`

Find last position of a character not in string.

Parameters

str String containing characters to avoid.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for a character not contained in *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2092 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_not_of()`.

5.379.3.59 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of
(_CharT __c, size_type __pos = npow) const [inline]`

Find last position of a character.

Parameters

c Character to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Note: equivalent to `rfind(c, pos)`.

Definition at line 2019 of file `basic_string.h`.

5.379.3.60 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of
(const basic_string< _CharT, _Traits, _Alloc > & __str, size_type
__pos = npow) const [inline]`

Find last position of a character of string.

Parameters

str String containing characters to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *str* within this string. If found, returns the index where it was found. If not found, returns *npos*.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

Definition at line 1972 of file basic_string.h.

Referenced by std::basic_string< char >::find_last_of().

5.379.3.61 `template<typename _CharT, typename _Traits, typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (const
_CharT * __s, size_type __pos, size_type __n) const`

Find last position of a character of C substring.

Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from.
- n* Number of characters from *s* to search for.

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the first *n* characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 831 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

5.379.3.62 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of
(const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a character of C string.

Parameters

- s* C string containing characters to locate.
- pos* Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for one of the characters of *s* within this string. If found, returns the index where it was found. If not found, returns *npos*.

Definition at line 2000 of file basic_string.h.

5.379.3.63 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::front
() const [inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 875 of file basic_string.h.

5.379.3.64 `template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::front ()
[inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 867 of file basic_string.h.

5.379.3.65 `template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string< _CharT, _Traits, _Alloc
>::get_allocator () const [inline]`

Return copy of allocator used to construct this string.

Definition at line 1781 of file basic_string.h.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, and `std::basic_string< _CharT, _Traits, _Alloc >::swap()`.

5.379.3.66 `template<typename _CharT, typename _Traits , typename _Alloc
> basic_string< _CharT, _Traits, _Alloc > & std::basic_string<
_CharT, _Traits, _Alloc >::insert (size_type __pos, const _CharT
* __s, size_type __n)`

Insert a C substring.

Parameters

pos Iterator referencing location in string to insert at.

s The C string to insert.

n The number of characters to insert.

Returns

Reference to this string.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2081

Exceptions

std::length_error If new length exceeds `max_size()`.

std::out_of_range If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If *pos* is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 360 of file `basic_string.tcc`.

```
5.379.3.67  template<typename _CharT, typename _Traits, typename _Alloc>
            iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p, _CharT __c ) [inline]
```

Insert one character.

Parameters

p Iterator referencing position in string to insert at.

c The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error If new length exceeds `max_size()`.

Inserts character *c* at position referenced by *p*. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If *p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1321 of file `basic_string.h`.

```
5.379.3.68  template<typename _CharT, typename _Traits, typename _Alloc>
            void std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator
            __p, size_type __n, _CharT __c ) [inline]
```

Insert multiple characters.

Parameters

p Iterator referencing location in string to insert at.

n Number of characters to insert

c The character to insert.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Inserts *n* copies of character *c* starting at the position referenced by iterator *p*. If adding characters causes the length to exceed `max_size()`, [*length_error*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1172 of file `basic_string.h`.

```
5.379.3.69  template<typename _CharT, typename _Traits, typename _Alloc>
            basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
                size_type __pos1, const basic_string< _CharT, _Traits, _Alloc > &
                __str ) [inline]
```

Insert value of a string.

Parameters

pos1 Iterator referencing location in string to insert at.

str The string to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Inserts value of *str* starting at *pos1*. If adding characters causes the length to exceed `max_size()`, [*length_error*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1218 of file `basic_string.h`.

Referenced by `std::basic_string< char >::insert()`.

```
5.379.3.70  template<typename _CharT, typename _Traits, typename _Alloc>
            void std::basic_string< _CharT, _Traits, _Alloc >::insert ( iterator
                __p, initializer_list< _CharT > __l ) [inline]
```

Insert an [*initializer_list*](#) of characters.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2083

Parameters

- p* Iterator referencing location in string to insert at.
- l* The [initializer_list](#) of characters to insert.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Definition at line 1199 of file `basic_string.h`.

```
5.379.3.71  template<typename _CharT, typename _Traits, typename _Alloc>
            template<class _InputIterator > void std::basic_string< _CharT,
            _Traits, _Alloc >::insert ( iterator __p, _InputIterator __beg,
            _InputIterator __end ) [inline]
```

Insert a range of characters.

Parameters

- p* Iterator referencing location in string to insert at.
- beg* Start of range.
- end* End of range.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed `max_size()`, [length_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1188 of file `basic_string.h`.

```
5.379.3.72  template<typename _CharT, typename _Traits, typename _Alloc>
            basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos, size_type __n, _CharT __c ) [inline]
```

Insert multiple characters.

Parameters

- pos* Index in string to insert at.
- n* Number of characters to insert
- c* The character to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Inserts *n* copies of character *c* starting at index *pos*. If adding characters causes the length to exceed `max_size()`, [*length_error*](#) is thrown. If *pos* > `length()`, [*out_of_range*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1304 of file `basic_string.h`.

5.379.3.73 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
size_type __pos1, const basic_string< _CharT, _Traits, _Alloc > &
__str, size_type __pos2, size_type __n) [inline]`

Insert a substring.

Parameters

pos1 Iterator referencing location in string to insert at.

str The string to insert.

pos2 Start of characters in *str* to insert.

n Number of characters to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos1* > `size()` or *pos2* > *str.size()*.

Starting at *pos1*, insert *n* character of *str* beginning with *pos2*. If adding characters causes the length to exceed `max_size()`, [*length_error*](#) is thrown. If *pos1* is beyond the end of this string or *pos2* is beyond the end of *str*, [*out_of_range*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1240 of file `basic_string.h`.

Referenced by `std::basic_string< char >::insert()`.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2085

5.379.3.74 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
size_type __pos, const _CharT * __s) [inline]`

Insert a C string.

Parameters

pos Iterator referencing location in string to insert at.

s The C string to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

Inserts the first *n* characters of *s* starting at *pos*. If adding characters causes the length to exceed `max_size()`, [*length_error*](#) is thrown. If *pos* is beyond `end()`, [*out_of_range*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1281 of file `basic_string.h`.

5.379.3.75 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::length ()
const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 714 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::collate< _CharT >::do_compare()`, `std::collate< _CharT >::do_transform()`, and `std::regex_traits< _Ch_type >::length()`.

5.379.3.76 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::max_size () const [inline]`

Returns the [*size\(\)*](#) of the largest possible string.

Definition at line 719 of file `basic_string.h`.

Referenced by `std::getline()`, and `std::operator>>()`.

5.379.3.77 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator+=(const basic_string< _CharT, _Traits, _Alloc > &
__str) [inline]`

Append a string to this string.

Parameters

str The string to append.

Returns

Reference to this string.

Definition at line 922 of file basic_string.h.

5.379.3.78 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator+=(const _CharT * __s) [inline]`

Append a C string.

Parameters

s The C string to append.

Returns

Reference to this string.

Definition at line 931 of file basic_string.h.

5.379.3.79 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator+=(_CharT __c) [inline]`

Append a character.

Parameters

c The character to append.

Returns

Reference to this string.

Definition at line 940 of file basic_string.h.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2087

5.379.3.80 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc
>::operator+=(initializer_list<_CharT> __l) [inline]`

Append an [initializer_list](#) of characters.

Parameters

l The [initializer_list](#) of characters to be appended.

Returns

Reference to this string.

Definition at line 953 of file `basic_string.h`.

5.379.3.81 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc
>::operator=(basic_string<_CharT, _Traits, _Alloc> && __str
) [inline]`

Move assign the value of *str* to this string.

Parameters

str Source string.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 573 of file `basic_string.h`.

5.379.3.82 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc
>::operator=(const basic_string<_CharT, _Traits, _Alloc> &
__str) [inline]`

Assign the value of *str* to this string.

Parameters

str Source string.

Definition at line 539 of file `basic_string.h`.

5.379.3.83 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator= (const _CharT * __s) [inline]`

Copy contents of *s* into this string.

Parameters

s Source null-terminated string.

Definition at line 547 of file basic_string.h.

5.379.3.84 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator= (initializer_list< _CharT > __l) [inline]`

Set value to string constructed from initializer list.

Parameters

l [std::initializer_list](#).

Definition at line 585 of file basic_string.h.

5.379.3.85 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::operator= (_CharT __c) [inline]`

Set value to string of length 1.

Parameters

c Source character.

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

Definition at line 558 of file basic_string.h.

5.379.3.86 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc
>::operator[] (size_type __pos) const [inline]`

Subscript access to the data contained in the string.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2089

Parameters

pos The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 816 of file basic_string.h.

```
5.379.3.87 template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::operator[] (
size_type __pos ) [inline]
```

Subscript access to the data contained in the string.

Parameters

pos The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).) Unshares the string.

Definition at line 833 of file basic_string.h.

```
5.379.3.88 template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::push_back (
_CharT __c ) [inline]
```

Append a single character.

Parameters

c Character to append.

Definition at line 1041 of file basic_string.h.

Referenced by `std::collate<_CharT>::do_transform()`, and `std::operator>>()`.

5.379.3.89 `template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc
>::rbegin () [inline]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 637 of file `basic_string.h`.

5.379.3.90 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc
>::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 646 of file `basic_string.h`.

5.379.3.91 `template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc
>::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 664 of file `basic_string.h`.

5.379.3.92 `template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend
() [inline]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 655 of file `basic_string.h`.

5.379.3.93 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace
(size_type __pos, size_type __n, const basic_string< _CharT,
_Traits, _Alloc > & __str) [inline]`

Replace characters with value from another string.

Parameters

pos Index of first character to replace.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2091

n Number of characters to be replaced.

str String to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos* is beyond the end of this string.

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*pos*,*pos*+*n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, [`out_of_range`](#) is thrown. If the length of the result exceeds `max_size()`, [`length_error`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1400 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

5.379.3.94 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace
(iterator __i1, iterator __i2, const basic_string< _CharT, _Traits,
_Alloc > & __str) [inline]`

Replace range of characters with string.

Parameters

i1 Iterator referencing start of range to replace.

i2 Iterator referencing end of range to replace.

str String value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*i1*,*i2*). In place, the value of *str* is inserted. If the length of result exceeds `max_size()`, [`length_error`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1506 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

5.379.3.95 `template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
 (iterator __i1, iterator __i2, const _CharT * __s, size_type __n
) [inline]`

Replace range of characters with C substring.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.
- n* Number of characters from *s* to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*i1*,*i2*). In place, the first *n* characters of *s* are inserted. If the length of result exceeds `max_size()`, [*length_error*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1524 of file `basic_string.h`.

5.379.3.96 `template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
 (iterator __i1, iterator __i2, const _CharT * __s) [inline]`

Replace range of characters with C string.

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- s* C string value to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2093

Removes the characters in the range [i1,i2). In place, the characters of *s* are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1545 of file `basic_string.h`.

```
5.379.3.97 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
( size_type __pos1, size_type __n1, const basic_string<_CharT,
_Traits, _Alloc> & __str, size_type __pos2, size_type __n2 )
[inline]
```

Replace characters with value from another string.

Parameters

pos1 Index of first character to replace.
n1 Number of characters to be replaced.
str String to insert.
pos2 Index of first character of *str* to use.
n2 Number of characters from *str* to use.

Returns

Reference to this string.

Exceptions

`std::out_of_range` If *pos1* > `size()` or *pos2* > `str.size()`.
`std::length_error` If new length exceeds `max_size()`.

Removes the characters in the range [*pos1*,*pos1* + *n*) from this string. In place, the value of *str* is inserted. If *pos* is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1422 of file `basic_string.h`.

Referenced by `std::basic_string<char>::replace()`.

```
5.379.3.98 template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
( size_type __pos, size_type __n1, size_type __n2, _CharT __c )
[inline]
```

Replace characters with multiple characters.

Parameters

pos Index of first character to replace.
n1 Number of characters to be replaced.
n2 Number of characters to insert.
c Character to insert.

Returns

Reference to this string.

Exceptions

[*std::out_of_range*](#) If *pos* > `size()`.
[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, *n2* copies of *c* are inserted. If *pos* is beyond end of string, [`out_of_range`](#) is thrown. If the length of result exceeds `max_size()`, [`length_error`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1488 of file `basic_string.h`.

5.379.3.99 `template<typename _CharT, typename _Traits, typename _Alloc>
 basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace
 (iterator __i1, iterator __i2, size_type __n, _CharT __c)
 [inline]`

Replace range of characters with multiple characters.

Parameters

i1 Iterator referencing start of range to replace.
i2 Iterator referencing end of range to replace.
n Number of characters to insert.
c Character to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [*i1*,*i2*). In place, *n* copies of *c* are inserted. If the length of result exceeds `max_size()`, [`length_error`](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1566 of file `basic_string.h`.

5.379 std::basic_string<_CharT, _Traits, _Alloc> Class Template Reference 2095

5.379.3.100 `template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator > basic_string& std::basic_string<
_CharT, _Traits, _Alloc>::replace (iterator __i1, iterator __i2,
_InputIterator __k1, _InputIterator __k2) [inline]`

Replace range of characters with range.

Parameters

i1 Iterator referencing start of range to replace.

i2 Iterator referencing end of range to replace.

k1 Iterator referencing start of range to insert.

k2 Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

[*std::length_error*](#) If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds `max_size()`, [*length_error*](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1589 of file `basic_string.h`.

5.379.3.101 `template<typename _CharT, typename _Traits, typename _Alloc
> basic_string<_CharT, _Traits, _Alloc> & std::basic_string<
_CharT, _Traits, _Alloc>::replace (size_type __pos, size_type
__n1, const _CharT * __s, size_type __n2)`

Replace characters with value of a C substring.

Parameters

pos Index of first character to replace.

n1 Number of characters to be replaced.

s C string to insert.

n2 Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos1* > *size()*.

std::length_error If new length exceeds *max_size()*.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, the first *n2* characters of *s* are inserted, or all of *s* if *n2* is too large. If *pos* is beyond end of string, *out_of_range* is thrown. If the length of result exceeds *max_size()*, *length_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 414 of file *basic_string.tcc*.

5.379.3.102 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::replace (size_type __pos, size_type __n1, const _CharT * __s
) [inline]`

Replace characters with value of a C string.

Parameters

pos Index of first character to replace.

n1 Number of characters to be replaced.

s C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range If *pos* > *size()*.

std::length_error If new length exceeds *max_size()*.

Removes the characters in the range [*pos*,*pos* + *n1*) from this string. In place, the characters of *s* are inserted. If *pos* is beyond end of string, *out_of_range* is thrown. If the length of result exceeds *max_size()*, *length_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1465 of file *basic_string.h*.

5.379.3.103 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc
>::replace (iterator __i1, iterator __i2, initializer_list< _CharT
> __l) [inline]`

Replace range of characters with *initializer_list*.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2097

Parameters

- i1* Iterator referencing start of range to replace.
- i2* Iterator referencing end of range to replace.
- l* The [initializer_list](#) of characters to insert.

Returns

Reference to this string.

Exceptions

[std::length_error](#) If new length exceeds `max_size()`.

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, [length_error](#) is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1657 of file `basic_string.h`.

Referenced by `std::basic_string< char >::replace()`.

5.379.3.104 `template<typename _CharT, typename _Traits, typename _Alloc
> void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
size_type __res_arg = 0)`

Attempt to preallocate enough memory for specified number of characters.

Parameters

res_arg Number of characters required.

Exceptions

[std::length_error](#) If *res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, [length_error](#) is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 502 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::capacity()`, `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

Referenced by `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::num_get<_CharT, _InIter>::do_get()`, and `std::operator>>()`.

5.379.3.105 `template<typename _CharT, typename _Traits, typename _Alloc>
> void std::basic_string<_CharT, _Traits, _Alloc>::resize (
size_type __n, _CharT __c)`

Resizes the string to the specified number of characters.

Parameters

- n* Number of characters the string should contain.
- c* Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *c*.

Definition at line 640 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::erase()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`.

5.379.3.106 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::resize (
size_type __n) [inline]`

Resizes the string to the specified number of characters.

Parameters

- n* Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 746 of file `basic_string.h`.

Referenced by `std::basic_string<char>::resize()`.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2099

5.379.3.107 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
const basic_string< _CharT, _Traits, _Alloc > & __str, size_type
__pos = npos) const [inline]`

Find last position of a string.

Parameters

str String to locate.

pos Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for value of *str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1853 of file basic_string.h.

Referenced by std::basic_string< char >::rfind().

5.379.3.108 `template<typename _CharT, typename _Traits , typename
_Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::rfind (_CharT __c,
size_type __pos = npos) const`

Find last position of a character.

Parameters

c Character to locate.

pos Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *pos*, searches backward for *c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 799 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

5.379.3.109 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (
const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a C string.

Parameters

s C string to locate.

pos Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the value of *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 1881 of file `basic_string.h`.

5.379.3.110 `template<typename _CharT, typename _Traits , typename
_Alloc > basic_string<_CharT, _Traits, _Alloc>::size_type
std::basic_string<_CharT, _Traits, _Alloc>::rfind (const
_CharT * __s, size_type __pos, size_type __n) const`

Find last position of a C substring.

Parameters

s C string to locate.

pos Index of character to search back from.

n Number of characters from *s* to search for.

Returns

Index of start of last occurrence.

Starting from *pos*, searches backward for the first *n* characters in *s* within this string. If found, returns the index where it begins. If not found, returns *npos*.

Definition at line 778 of file `basic_string.tcc`.

References `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

5.379 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference 2101

5.379.3.111 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit ()
[inline]`

A non-binding request to reduce [capacity\(\)](#) to [size\(\)](#).

Definition at line 752 of file `basic_string.h`.

5.379.3.112 `template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size ()
const [inline]`

Returns the number of characters in the string, not including any /// null-termination.

Definition at line 708 of file `basic_string.h`.

Referenced by `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::bitset< _S_match_flag_last >::bitset()`, `std::basic_string< _CharT, _Traits, _Alloc >::compare()`, `std::basic_string< char >::compare()`, `std::basic_string< _CharT, _Traits, _Alloc >::find()`, `std::basic_string< char >::find()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of()`, `std::basic_string< char >::find_first_not_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_first_of()`, `std::basic_string< char >::find_first_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of()`, `std::basic_string< char >::find_last_not_of()`, `std::basic_string< _CharT, _Traits, _Alloc >::find_last_of()`, `std::basic_string< char >::find_last_of()`, `std::basic_string< char >::insert()`, `std::operator+`, `std::basic_string< char >::replace()`, `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`, `std::basic_string< _CharT, _Traits, _Alloc >::resize()`, `std::basic_string< _CharT, _Traits, _Alloc >::rfind()`, `std::basic_string< char >::rfind()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::str()`, and `std::regex_traits< _Ch_type >::transform()`.

5.379.3.113 `template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr ()
size_type __pos = 0, size_type __n = npos) const [inline]`

Get a substring.

Parameters

pos Index of first character (default 0).

n Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

[*std::out_of_range*](#) If `pos > size()`.

Construct and return a new string using the *n* characters starting at *pos*. If the string is too short, use the remainder of the characters. If *pos* is beyond the end of the string, [*out_of_range*](#) is thrown.

Definition at line 2153 of file `basic_string.h`.

5.379.3.114 `template<typename _CharT, typename _Traits, typename _Alloc
> void std::basic_string< _CharT, _Traits, _Alloc >::swap (
basic_string< _CharT, _Traits, _Alloc > & __s)`

Swap contents with another string.

Parameters

s String to swap with.

Exchanges the contents of this string with that of *s* in constant time.

Definition at line 519 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`.

Referenced by `std::swap()`.

5.379.4 Member Data Documentation

5.379.4.1 `template<typename _CharT, typename _Traits, typename
_Alloc> const basic_string< _CharT, _Traits, _Alloc >::size_type
std::basic_string< _CharT, _Traits, _Alloc >::npos [static]`

Value returned by various member functions when they fail.

Definition at line 278 of file `basic_string.h`.

The documentation for this class was generated from the following files:

- [basic_string.h](#)
- [basic_string.tcc](#)

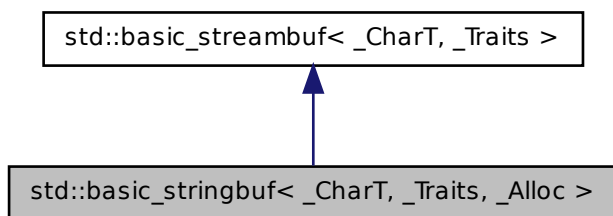
5.380 `std::basic_stringbuf< _CharT, _Traits, _Alloc >` Class Template Reference

The actual work of input and output (for `std::string`).

5.380 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.).

Inheritance diagram for `std::basic_stringbuf<_CharT, _Traits, _Alloc>`:



Public Types

- `typedef __string_type::size_type __size_type`
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`
- `typedef basic_string< char_type, _Traits, _Alloc > __string_type`
- `typedef _Alloc allocator_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`
- `typedef _Traits traits_type`

Public Member Functions

- `basic_stringbuf (ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `basic_stringbuf (const __string_type &__str, ios_base::openmode __mode=ios_base::in|ios_base::out)`
- `streamsize in_avail ()`
- `int_type sbumpc ()`
- `int_type sgetc ()`
- `streamsize sgetn (char_type *__s, streamsize __n)`
- `int_type snextc ()`

- [int_type sputback](#) ([char_type __c](#))
- [int_type sputc](#) ([char_type __c](#))
- [streamsize sputn](#) ([const char_type *__s](#), [streamsize __n](#))
- [void stoss](#) ()
- [void str](#) ([const __string_type &__s](#))
- [__string_type str](#) () [const](#)
- [int_type sungetc](#) ()

Protected Member Functions

- [void _M_stringbuf_init](#) ([ios_base::openmode __mode](#))
 - [void _M_sync](#) ([char_type *__base](#), [__size_type __i](#), [__size_type __o](#))
 - [void _M_update_egptr](#) ()
 - [void gbump](#) ([int __n](#))
 - [virtual void imbue](#) ([const locale &](#))
 - [virtual int_type overflow](#) ([int_type __c=traits_type::eof\(\)](#))
 - [virtual int_type overflow](#) ([int_type=traits_type::eof\(\)](#))
 - [virtual int_type pbackfail](#) ([int_type __c=traits_type::eof\(\)](#))
 - [virtual int_type pbackfail](#) ([int_type=traits_type::eof\(\)](#))
 - [void pbump](#) ([int __n](#))
 - [virtual pos_type seekoff](#) ([off_type](#), [ios_base::seekdir](#), [ios_base::openmode=ios_base::in|ios_base::out](#))
 - [virtual pos_type seekoff](#) ([off_type __off](#), [ios_base::seekdir __way](#), [ios_base::openmode __mode=ios_base::in|ios_base::out](#))
 - [virtual pos_type seekpos](#) ([pos_type __sp](#), [ios_base::openmode __mode=ios_base::in|ios_base::out](#))
 - [virtual pos_type seekpos](#) ([pos_type](#), [ios_base::openmode=ios_base::in|ios_base::out](#))
 - [virtual basic_streambuf< char_type, _Traits > * setbuf](#) ([char_type *](#), [streamsize](#))
 - [virtual __streambuf_type * setbuf](#) ([char_type *__s](#), [streamsize __n](#))
 - [void setg](#) ([char_type *__gbeg](#), [char_type *__gnext](#), [char_type *__gend](#))
 - [void setp](#) ([char_type *__pbeg](#), [char_type *__pend](#))
 - [virtual streamsize showmanyc](#) ()
 - [virtual int sync](#) ()
 - [virtual int_type uflow](#) ()
 - [virtual int_type underflow](#) ()
 - [virtual streamsize xsgetn](#) ([char_type *__s](#), [streamsize __n](#))
 - [virtual streamsize xsputn](#) ([const char_type *__s](#), [streamsize __n](#))
-
- [char_type * eback](#) () [const](#)
 - [char_type * gptr](#) () [const](#)
 - [char_type * egptr](#) () [const](#)
-
- [char_type * pbase](#) () [const](#)
 - [char_type * pptr](#) () [const](#)
 - [char_type * epptr](#) () [const](#)

Protected Attributes

- [ios_base::openmode](#) [_M_mode](#)
- [__string_type](#) [_M_string](#)

Friends

- `template<bool _IsMove, typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__-`
`type __copy_move_a2 (istreambuf_iterator< _CharT2 >, istreambuf_iterator<`
`_CharT2 >, _CharT2 *)`
 - [streamsize](#) [__copy_streambufs_eof](#) ([__streambuf_type](#) *, [__streambuf_type](#) *,
`bool &)`
 - `class basic_ios< char_type, traits_type >`
 - `class basic_istream< char_type, traits_type >`
 - `class basic_ostream< char_type, traits_type >`
 - `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_`
`iterator< _CharT2 > >::__type find (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & getline (basic_istream< _CharT2, _`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)`
 - `class istreambuf_iterator< char_type, traits_type >`
 - `template<typename _CharT2, typename _Traits2, typename _Alloc >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)`
 - `template<typename _CharT2, typename _Traits2 >`
`basic_istream< _CharT2, _Traits2 > & operator>> (basic_istream< _CharT2,`
`_Traits2 > &, _CharT2 *)`
 - `class ostreambuf_iterator< char_type, traits_type >`
-
- [char_type](#) * [_M_in_beg](#)
 - [char_type](#) * [_M_in_cur](#)
 - [char_type](#) * [_M_in_end](#)
 - [char_type](#) * [_M_out_beg](#)
 - [char_type](#) * [_M_out_cur](#)
 - [char_type](#) * [_M_out_end](#)
 - [locale](#) [_M_buf_locale](#)
 - [locale](#) [pubimbue](#) (const [locale](#) &__loc)
 - [locale](#) [getloc](#) () const
 - [__streambuf_type](#) * [pubsetbuf](#) ([char_type](#) * __s, [streamsize](#) __n)

- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `int pubsync` ()

5.380.1 Detailed Description

`template<typename _CharT, typename _Traits, typename _Alloc> class std::basic_stringbuf< _CharT, _Traits, _Alloc >`

The actual work of input and output (for `std::string`).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.). For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

Definition at line 58 of file `sstream`.

5.380.2 Member Typedef Documentation

5.380.2.1 `template<typename _CharT, typename _Traits, typename _Alloc> typedef basic_streambuf<char_type, traits_type> std::basic_stringbuf< _CharT, _Traits, _Alloc >::__streambuf_type`

This is a non-standard type.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 71 of file `sstream`.

5.380.2.2 `template<typename _CharT, typename _Traits, typename _Alloc> typedef _CharT std::basic_stringbuf< _CharT, _Traits, _Alloc >::__char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 62 of file `sstream`.

5.380 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

5.380.2.3 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::int_type std::basic_stringbuf< _CharT, _Traits,
_Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 67 of file sstream.

5.380.2.4 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::off_type std::basic_stringbuf< _CharT, _Traits,
_Alloc >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 69 of file sstream.

5.380.2.5 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef traits_type::pos_type std::basic_stringbuf< _CharT, _Traits,
_Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 68 of file sstream.

5.380.2.6 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _Traits std::basic_stringbuf< _CharT, _Traits, _Alloc
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_streambuf< _CharT, _Traits >](#).

Definition at line 63 of file sstream.

5.380.3 Constructor & Destructor Documentation

5.380.3.1 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf
(ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, explicit]`

Starts with an empty string buffer.

Parameters

mode Whether the buffer can read, or write, or both.

The default constructor initializes the parent class using its own default ctor.

Definition at line 92 of file sstream.

5.380.3.2 `template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf
(const __string_type & __str, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, explicit]`

Starts with an existing string buffer.

Parameters

str A string to copy as a starting buffer.

mode Whether the buffer can read, or write, or both.

This constructor initializes the parent class using its own default ctor.

Definition at line 105 of file sstream.

5.380.4 Member Function Documentation

5.380.4.1 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::eback () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence

5.380 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Referenced

- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 460 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.380.4.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::egptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 466 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.380.4.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::epptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 513 of file streambuf.

Referenced by std::basic_streambuf< _CharT, _Traits >::xsputn().

5.380.4.4 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::gbump (int __n)
[inline, protected, inherited]`

Moving the read position.

Parameters

n The delta by which to move.

This just advances the read position without returning any data.

Definition at line 476 of file streambuf.

5.380.4.5 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::getloc () const
[inline, inherited]`

Locale access.

Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 222 of file streambuf.

5.380.4.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::gptr () const
[inline, protected, inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- [eback\(\)](#) returns the beginning pointer for the input sequence
- [gptr\(\)](#) returns the next pointer for the input sequence
- [egptr\(\)](#) returns the end pointer for the input sequence

Definition at line 463 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::imbue()`, `std::basic_filebuf< _CharT, _Traits >::showmanyc()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`, and `std::basic_filebuf< _CharT, _Traits >::underflow()`.

5.380 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

5.380.4.7 `template<typename _CharT, typename _Traits> virtual void
std::basic_streambuf<_CharT, _Traits>::imbue (const locale &)
[inline, protected, virtual, inherited]`

Changes translations.

Parameters

loc A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 554 of file streambuf.

5.380.4.8 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::in_avail () [inline,
inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 262 of file streambuf.

5.380.4.9 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::overflow (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

c An additional character to consume.

Returns

eof() to indicate failure, something else (usually *c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *c* is not eof().

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented in [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#).

Definition at line 746 of file streambuf.

Referenced by [std::basic_streambuf<_CharT, _Traits>::xsputn\(\)](#).

5.380.4.10 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf<_CharT, _Traits>::pbackfail (int_type =
traits_type::eof()) [inline, protected, virtual,
inherited]`

Tries to back up the input sequence.

Parameters

c The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of [gptr\(\)](#), [eback\(\)](#), and [pptr\(\)](#) are the same as for [underflow\(\)](#).

Note

Base class version does nothing, returns eof().

5.380 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Referenced

Reimplemented in [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#).

Definition at line 702 of file streambuf.

5.380.4.11 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pbase () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 507 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`.

5.380.4.12 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::pbump (int __n)
[inline, protected, inherited]`

Moving the write position.

Parameters

- n* The delta by which to move.

This just advances the write position without returning any data.

Definition at line 523 of file streambuf.

Referenced by `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.380.4.13 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::pptr () const
[inline, protected, inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- [pbase\(\)](#) returns the beginning pointer for the output sequence
- [pptr\(\)](#) returns the next pointer for the output sequence
- [epptr\(\)](#) returns the end pointer for the output sequence

Definition at line 510 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::sync()`, and `std::basic_streambuf< _CharT, _Traits >::xsputn()`.

5.380.4.14 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale
& __loc) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 205 of file streambuf.

5.380.4.15 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type
__off, ios_base::seekdir __way, ios_base::openmode __mode =
ios_base::in | ios_base::out) [inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 239 of file streambuf.

5.380 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

5.380.4.16 `template<typename _CharT, typename _Traits> pos_type
std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type
__sp, ios_base::openmode __mode = ios_base::in | ios_base::out)
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 244 of file `streambuf`.

5.380.4.17 `template<typename _CharT, typename _Traits>
__streambuf_type* std::basic_streambuf< _CharT, _Traits
>::pubsetbuf (char_type * __s, streamsize __n) [inline,
inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 235 of file `streambuf`.

5.380.4.18 `template<typename _CharT, typename _Traits> int
std::basic_streambuf< _CharT, _Traits >::pubsync ()
[inline, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 249 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::sync()`.

5.380.4.19 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 294 of file `streambuf`.

Referenced by `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, and `std::istreambuf_iterator< _CharT, _Traits >::operator++()`.

5.380.4.20 `template<typename _CharT, typename _Traits> virtual
pos_type std::basic_streambuf< _CharT, _Traits >::seekoff
(off_type, ios_base::seekdir, ios_base::openmode =
ios_base::in | ios_base::out) [inline, protected,
virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 580 of file `streambuf`.

5.380.4.21 `template<typename _CharT, typename _Traits> virtual pos_type
std::basic_streambuf< _CharT, _Traits >::seekpos (pos_type,
ios_base::openmode = ios_base::in | ios_base::out) [inline,
protected, virtual, inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

5.380 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

Definition at line 592 of file streambuf.

5.380.4.22 `template<typename _CharT, typename _Traits, typename _Alloc>
virtual __streambuf_type* std::basic_stringbuf< _CharT, _Traits,
_Alloc >::setbuf(char_type * __s, streamsize __n) [inline,
protected, virtual]`

Manipulates the buffer.

Parameters

s Pointer to a buffer area.

n Size of *s*.

Returns

this

If no buffer has already been created, and both *s* and *n* are non-zero, then *s* is used as a buffer; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more.

Definition at line 196 of file sstream.

References `std::basic_string< _CharT, _Traits, _Alloc >::clear()`.

5.380.4.23 `template<typename _CharT, typename _Traits> virtual
basic_streambuf<char_type,_Traits>* std::basic_streambuf<
_CharT,_Traits >::setbuf(char_type *, streamsize) [inline,
protected, virtual, inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html> for more on this function.

Note

Base class version does nothing, returns *this*.

Definition at line 569 of file streambuf.

5.380.4.24 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setg (char_type *
__gbeg, char_type * __gnext, char_type * __gend) [inline,
protected, inherited]`

Setting the three read area pointers.

Parameters

gbeg A pointer.
gnext A pointer.
gend A pointer.

Postcondition

gbeg == `eback()`, *gnext* == `gptr()`, and *gend* == `egptr()`

Definition at line 487 of file streambuf.

5.380.4.25 `template<typename _CharT, typename _Traits> void
std::basic_streambuf< _CharT, _Traits >::setp (char_type
* __pbeg, char_type * __pend) [inline, protected,
inherited]`

Setting the three write area pointers.

Parameters

pbeg A pointer.
pend A pointer.

Postcondition

pbeg == `pbase()`, *pbeg* == `pptr()`, and *pend* == `epptr()`

Definition at line 533 of file streambuf.

5.380.4.26 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sgetc () [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

5.380 std::basic_stringbuf<_CharT, _Traits, _Alloc> Class Template Reference

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 316 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.380.4.27 `template<typename _CharT, typename _Traits> streamsize
std::basic_stringbuf<_CharT, _Traits>::sgetn(char_type * __s,
streamsize __n) [inline, inherited]`

Entry point for `xsgetn`.

Parameters

s A buffer area.

n A count.

Returns `xsgetn(s,n)`. The effect is to fill `s[0]` through `s[n-1]` with characters from the input sequence, if possible.

Definition at line 335 of file streambuf.

5.380.4.28 `template<typename _CharT, typename _Traits, typename _Alloc>
virtual streamsize std::basic_stringbuf<_CharT, _Traits, _Alloc>
::showmanyc() [inline, protected, virtual]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.
[27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic_streambuf<_CharT, _Traits>](#).

Definition at line 164 of file sstream.

5.380.4.29 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::snextc() [inline,
inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls [sbumpc\(\)](#), and if that function returns `traits::eof()`, so does this function. Otherwise, [sgetc\(\)](#).

Definition at line 276 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, `std::basic_istream<_CharT, _Traits>::getline()`, `std::basic_istream<_CharT, _Traits>::ignore()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.380.4.30 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputbackc(char_type
__c) [inline, inherited]`

Pushing characters back into the input stream.

Parameters

c The character to push back.

Returns

The previous character, if possible.

Similar to [sungetc\(\)](#), but *c* is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be *c*.

Definition at line 350 of file streambuf.

Referenced by `std::basic_istream<_CharT, _Traits>::putback()`.

5.380.4.31 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf<_CharT, _Traits>::sputc(char_type __c)
[inline, inherited]`

Entry point for all single-character output functions.

5.380 std::basic_stringbuf<_CharT, _Traits, _Alloc> Class Template Reference

Parameters

c A character to output.

Returns

c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *c* in that position, increments the position, and returns `traits::to_int_type(c)`. If a write position is not available, returns `overflow(c)`.

Definition at line 402 of file `streambuf`.

Referenced by `std::basic_istream<_CharT, _Traits>::get()`, and `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

5.380.4.32 `template<typename _CharT, typename _Traits> streamsize
std::basic_streambuf<_CharT, _Traits>::sputn (const char_type
* __s, streamsize __n) [inline, inherited]`

Entry point for all single-character output functions.

Parameters

s A buffer read area.

n A count.

One of two public output functions.

Returns `xspn(s,n)`. The effect is to write `s[0]` through `s[n-1]` to the output sequence, if possible.

Definition at line 428 of file `streambuf`.

5.380.4.33 `template<typename _CharT, typename _Traits> void
std::basic_streambuf<_CharT, _Traits>::stossc () [inline,
inherited]`

Tosses a character.

Advances the read pointer, ignoring the character that would have been read.

See <http://gcc.gnu.org/ml/libstdc++/2002-05/msg00168.html>

Definition at line 761 of file `streambuf`.

5.380.4.34 `template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (const
__string_type & __s) [inline]`

Setting a new buffer.

Parameters

s The string to use as a new sequence.

Deallocates any previous stored sequence, then copies *s* to use as a new one.

Definition at line 144 of file sstream.

References `std::basic_string< _CharT, _Traits, _Alloc >::assign()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

5.380.4.35 `template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (
) const [inline]`

Copying out the string buffer.

Returns

A copy of one of the underlying sequences.

If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1

Definition at line 120 of file sstream.

5.380.4.36 `template<typename _CharT, typename _Traits> int_type
std::basic_streambuf< _CharT, _Traits >::sungetc () [inline,
inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `ebackfail()`. The effect is to *unget* the last character *gotten*.

5.380 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

Definition at line 375 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

5.380.4.37 `template<typename _CharT, typename _Traits> virtual int
std::basic_streambuf< _CharT, _Traits >::sync (void)
[inline, protected, virtual, inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in [std::basic_filebuf< _CharT, _Traits >](#), [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#), [std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >](#), and [std::basic_filebuf< char_type, traits_type >](#).

Definition at line 605 of file streambuf.

5.380.4.38 `template<typename _CharT, typename _Traits> virtual int_type
std::basic_streambuf< _CharT, _Traits >::uflow () [inline,
protected, virtual, inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as [underflow\(\)](#), and in fact is required to call that function. It also returns the new character, like [underflow\(\)](#) does. However, this function also moves the read position forward by one.

Reimplemented in [__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >](#).

Definition at line 678 of file streambuf.

5.380.4.39 `template<class _CharT, class _Traits, class _Alloc >
 basic_stringbuf< _CharT, _Traits, _Alloc >::int_type
 std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow ()
 [protected, virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html>

Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 131 of file `sstream.tcc`.

References `std::basic_stringbuf< _CharT, _Traits, _Alloc >::M_mode`, `std::basic_streambuf< _CharT, _Traits >::egptr()`, `std::basic_streambuf< _CharT, _Traits >::gptr()`, and `std::ios_base::in`.

5.380.4.40 `template<typename _CharT, typename _Traits > streamsize
 std::basic_streambuf< _CharT, _Traits >::xsgetn (char_type *
 __s, streamsize __n) [protected, virtual, inherited]`

Multiple character extraction.

Parameters

s A buffer area.

n Maximum number of characters to assign.

Returns

The number of characters assigned.

5.380 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

Fills `s[0]` through `s[n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 45 of file `streambuf.tcc`.

References `std::min()`.

```
5.380.4.41  template<typename _CharT , typename _Traits > streamsize
              std::basic_streambuf< _CharT, _Traits >::xsputn ( const
              char_type * __s, streamsize __n ) [protected, virtual,
              inherited]
```

Multiple character insertion.

Parameters

`s` A buffer area.

`n` Maximum number of characters to write.

Returns

The number of characters written.

Writes `s[0]` through `s[n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 79 of file `streambuf.tcc`.

References `std::basic_streambuf<_CharT, _Traits>::epptr()`, `std::min()`, `std::basic_streambuf<_CharT, _Traits>::overflow()`, `std::basic_streambuf<_CharT, _Traits>::pbump()`, and `std::basic_streambuf<_CharT, _Traits>::pptr()`.

5.380.5 Member Data Documentation

5.380.5.1 `template<typename _CharT, typename _Traits> locale
std::basic_streambuf< _CharT, _Traits >::_M_buf_locale
[protected, inherited]`

Current locale setting.

Definition at line 188 of file streambuf.

Referenced by `std::basic_filebuf< _CharT, _Traits >::basic_filebuf()`.

5.380.5.2 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_beg
[protected, inherited]`

This is based on `_IO_FILE`, just reordered to be more consistent, and is intended to be the most minimal abstraction for an internal buffer.

- `get` == input == read
- `put` == output == write

Definition at line 180 of file streambuf.

5.380.5.3 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 181 of file streambuf.

5.380.5.4 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_in_end
[protected, inherited]`

Entry point for [imbue\(\)](#).

5.380 `std::basic_stringbuf<_CharT, _Traits, _Alloc>` Class Template Reference

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 182 of file `streambuf`.

5.380.5.5 `template<typename _CharT, typename _Traits, typename _Alloc>
ios_base::openmode std::basic_stringbuf<_CharT, _Traits, _Alloc
>::_M_mode [protected]`

Place to stash in || out || in | out settings for current stringbuf.

Definition at line 77 of file `sstream`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.

5.380.5.6 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_out_beg
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 183 of file `streambuf`.

5.380.5.7 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf<_CharT, _Traits>::_M_out_cur
[protected, inherited]`

Entry point for [imbue\(\)](#).

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 184 of file `streambuf`.

5.380.5.8 `template<typename _CharT, typename _Traits> char_type*
std::basic_streambuf< _CharT, _Traits >::_M_out_end
[protected, inherited]`

Entry point for `imbue()`.

Parameters

loc The new locale.

Returns

The previous locale.

Calls the derived `imbue(loc)`.

Definition at line 185 of file `streambuf`.

The documentation for this class was generated from the following files:

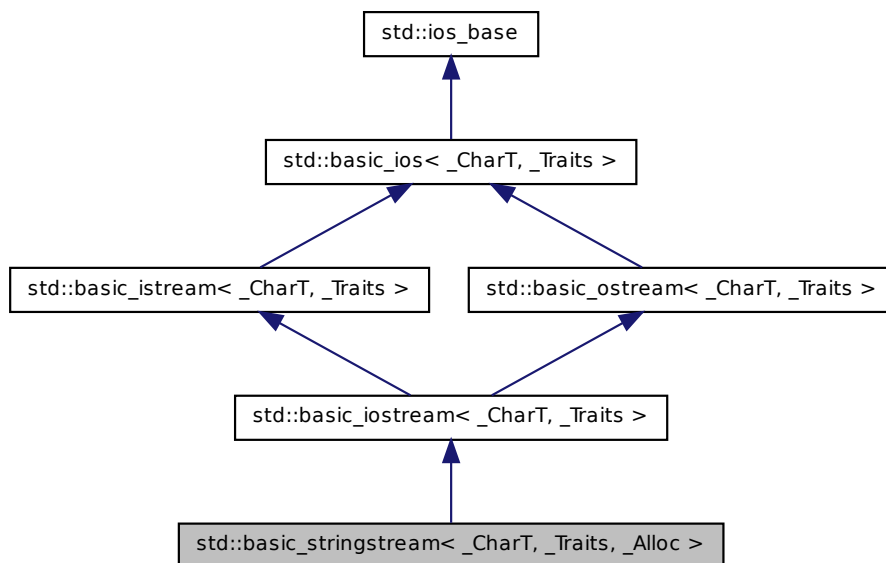
- [sstream](#)
- [sstream.tcc](#)

5.381 `std::basic_stringstream< _CharT, _Traits, _- Alloc >` Class Template Reference

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type [std::basic_string](#), using the inherited functions from [std::basic_iostream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as `sb`.

Inheritance diagram for `std::basic_stringstream< _CharT, _Traits, _Alloc >`:



Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`

- typedef [basic_stringbuf](#)< [_CharT](#), [_Traits](#), [_Alloc](#) > [__stringbuf_type](#)
- typedef [_Alloc](#) [allocator_type](#)
- typedef [_CharT](#) [char_type](#)
- typedef [_CharT](#) [char_type](#)
- enum [event](#) { [erase_event](#), [imbue_event](#), [copyfmt_event](#) }
- typedef void(* [event_callback](#))([event](#), [ios_base](#) &, int)
- typedef [_Ios_Fmtflags](#) [fmtflags](#)
- typedef [_Traits::int_type](#) [int_type](#)
- typedef [traits_type::int_type](#) [int_type](#)
- typedef int [io_state](#)
- typedef [_Ios_Iostate](#) [iostate](#)
- typedef [traits_type::off_type](#) [off_type](#)
- typedef [_Traits::off_type](#) [off_type](#)
- typedef int [open_mode](#)
- typedef [_Ios_Openmode](#) [openmode](#)
- typedef [traits_type::pos_type](#) [pos_type](#)
- typedef [_Traits::pos_type](#) [pos_type](#)
- typedef int [seek_dir](#)
- typedef [_Ios_Seekdir](#) [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [_Traits](#) [traits_type](#)

- typedef [num_put](#)< [_CharT](#), [ostreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_put_type](#)

Public Member Functions

- [basic_stringstream](#) ([ios_base::openmode](#) __m=[ios_base::out](#)|[ios_base::in](#))
- [basic_stringstream](#) (const [__string_type](#) &__str, [ios_base::openmode](#) __m=[ios_base::out](#)|[ios_base::in](#))
- [~basic_stringstream](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const

- [char_type fill](#) () const
- [char_type fill](#) (char_type __ch)
- [fmtflags flags](#) () const
- [fmtflags flags](#) (fmtflags __fmtfl)
- [__ostream_type & flush](#) ()
- [streamsize gcount](#) () const
- template<>
 [basic_istream](#)< char > & [getline](#) (char_type *__s, streamsize __n, char_type __delim)
- template<>
 [basic_istream](#)< wchar_t > & [getline](#) (char_type *__s, streamsize __n, char_type __delim)
- [locale getloc](#) () const
- bool [good](#) () const
- template<>
 [basic_istream](#)< char > & [ignore](#) (streamsize __n, int_type __delim)
- template<>
 [basic_istream](#)< char > & [ignore](#) (streamsize __n)
- template<>
 [basic_istream](#)< wchar_t > & [ignore](#) (streamsize __n)
- template<>
 [basic_istream](#)< wchar_t > & [ignore](#) (streamsize __n, int_type __delim)
- [locale imbue](#) (const [locale](#) &__loc)
- long & [iword](#) (int __ix)
- char [narrow](#) (char_type __c, char __dfault) const
- [streamsize precision](#) () const
- [streamsize precision](#) (streamsize __prec)
- void *& [pword](#) (int __ix)
- [__stringbuf_type * rdbuf](#) () const
- [basic_streambuf](#)< _CharT, _Traits > * [rdbuf](#) ([basic_streambuf](#)< _CharT, _Traits > * __sb)
- [iostate rdstate](#) () const
- void [register_callback](#) (event_callback __fn, int __index)
- [__ostream_type & seekp](#) (pos_type)
- [__ostream_type & seekp](#) (off_type, ios_base::seekdir)
- [fmtflags setf](#) (fmtflags __fmtfl)
- [fmtflags setf](#) (fmtflags __fmtfl, fmtflags __mask)
- void [setstate](#) (iostate __state)
- [__string_type str](#) () const
- void [str](#) (const __string_type &__s)
- [pos_type tellp](#) ()
- [basic_ostream](#)< _CharT, _Traits > * [tie](#) ([basic_ostream](#)< _CharT, _Traits > * __tistr)

- [basic_ostream](#)< [_CharT](#), [_Traits](#) > * [tie](#) () const
- void [unsetf](#) ([fmtflags](#) __mask)
- [char_type](#) [widen](#) (char __c) const
- [streamsize](#) [width](#) ([streamsize](#) __wide)
- [streamsize](#) [width](#) () const
- [__istream_type](#) & [operator](#)>> ([__istream_type](#) &(*__pf)([__istream_type](#) &))
- [__istream_type](#) & [operator](#)>> ([__ios_type](#) &(*__pf)([__ios_type](#) &))
- [__istream_type](#) & [operator](#)>> ([ios_base](#) &(*__pf)([ios_base](#) &))

Arithmetic Extractors

All the [operator](#)>> functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type [std::basic_istream::sentry](#) with the second argument ([noskipws](#)) set to false. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the [sentry](#) status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, [ios_base::badbit](#) will be turned on in the stream's error state without causing an [ios_base::failure](#) to be thrown. The original exception will then be rethrown.

- [__istream_type](#) & [operator](#)>> (bool &__n)
- [__istream_type](#) & [operator](#)>> (short &__n)
- [__istream_type](#) & [operator](#)>> (unsigned short &__n)
- [__istream_type](#) & [operator](#)>> (int &__n)
- [__istream_type](#) & [operator](#)>> (unsigned int &__n)
- [__istream_type](#) & [operator](#)>> (long &__n)
- [__istream_type](#) & [operator](#)>> (unsigned long &__n)
- [__istream_type](#) & [operator](#)>> (long long &__n)
- [__istream_type](#) & [operator](#)>> (unsigned long long &__n)
- [__istream_type](#) & [operator](#)>> (float &__f)
- [__istream_type](#) & [operator](#)>> (double &__f)
- [__istream_type](#) & [operator](#)>> (long double &__f)
- [__istream_type](#) & [operator](#)>> (void *&__p)
- [__istream_type](#) & [operator](#)>> ([__streambuf_type](#) *__sb)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type [std::basic_istream::sentry](#) with the second argument ([noskipws](#)) set to true. This has several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the [sentry](#) status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by [gcount\(\)](#).

If an exception is thrown during extraction, [ios_base::badbit](#) will be turned on in the stream's error state without causing an [ios_base::failure](#) to be thrown. The original exception will then be rethrown.

- [int_type](#) `get ()`
- [__istream_type](#) & `get (char_type &__c)`
- [__istream_type](#) & `get (char_type *__s, streamsize __n, char_type __delim)`
- [__istream_type](#) & `get (char_type *__s, streamsize __n)`
- [__istream_type](#) & `get (__streambuf_type &__sb, char_type __delim)`
- [__istream_type](#) & `get (__streambuf_type &__sb)`
- [__istream_type](#) & `getline (char_type *__s, streamsize __n, char_type __delim)`
- [__istream_type](#) & `getline (char_type *__s, streamsize __n)`
- [__istream_type](#) & `ignore ()`
- [__istream_type](#) & `ignore (streamsize __n)`
- [__istream_type](#) & `ignore (streamsize __n, int_type __delim)`
- [int_type](#) `peek ()`
- [__istream_type](#) & `read (char_type *__s, streamsize __n)`
- `streamsize` `readsome (char_type *__s, streamsize __n)`
- [__istream_type](#) & `putback (char_type __c)`
- [__istream_type](#) & `unget ()`
- `int` `sync ()`
- [pos_type](#) `tellg ()`
- [__istream_type](#) & `seekg (pos_type)`
- [__istream_type](#) & `seekg (off_type, ios_base::seekdir)`
- `operator void * () const`
- `bool operator! () const`
- [__ostream_type](#) & `operator<< (__ostream_type &(__pf)(__ostream_type &))`
- [__ostream_type](#) & `operator<< (__ios_type &(__pf)(__ios_type &))`
- [__ostream_type](#) & `operator<< (ios_base &(__pf)(ios_base &))`

Arithmetic Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type [std::basic_ostream::sentry](#). This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, [ios_base::badbit](#) will be turned on in the stream's error state without causing an [ios_base::failure](#) to be thrown. The original exception will then be rethrown.

- [__ostream_type](#) & `operator<< (long __n)`

- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`
- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`
- `__ostream_type & operator<< (const void *__p)`
- `__ostream_type & operator<< (__streambuf_type *__sb)`

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

Static Public Member Functions

- static bool `sync_with_stdio` (bool __sync=true)
- static int `xalloc` () throw ()

Static Public Attributes

- static const `fmtflags adjustfield`
- static const `openmode app`
- static const `openmode ate`
- static const `iostate badbit`
- static const `fmtflags basefield`
- static const `seekdir beg`
- static const `openmode binary`
- static const `fmtflags boolalpha`

- static const [seekdir](#) cur
- static const [fmtflags](#) dec
- static const [seekdir](#) end
- static const [iostate](#) eofbit
- static const [iostate](#) failbit
- static const [fmtflags](#) fixed
- static const [fmtflags](#) floatfield
- static const [iostate](#) goodbit
- static const [fmtflags](#) hex
- static const [openmode](#) in
- static const [fmtflags](#) internal
- static const [fmtflags](#) left
- static const [fmtflags](#) oct
- static const [openmode](#) out
- static const [fmtflags](#) right
- static const [fmtflags](#) scientific
- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- template<typename _ValueT >
 [__istream_type](#) & [_M_extract](#) (_ValueT &__v)
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- template<typename _ValueT >
 [__ostream_type](#) & [_M_insert](#) (_ValueT __v)
- void [init](#) ([basic_streambuf](#)< _CharT, _Traits > *__sb)

Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `streamsize _M_gcount`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

Friends

- class `sentry`
- class `sentry`

5.381.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc> class
std::basic_stringstream< _CharT, _Traits, _Alloc >
```

Controlling input and output for `std::string`.

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 476 of file `sstream`.

5.381.2 Member Typedef Documentation

5.381.2.1 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_istream<_CharT, _Traits
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file `istream`.

5.381.2.2 `template<typename _CharT, typename _Traits> typedef
ctype<_CharT> std::basic_ostream<_CharT, _Traits
>::__ctype_type [inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 71 of file `ostream`.

5.381.2.3 `template<typename _CharT, typename _Traits> typedef
num_get<_CharT, istreambuf_iterator<_CharT, _Traits>
> std::basic_istream<_CharT, _Traits>::__num_get_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file `istream`.

5.381.2.4 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented in [std::basic_ostream<_CharT, _Traits>](#), [std::basic_ostream<char, _Traits>](#), and [std::basic_ostream<char>](#).

Definition at line 84 of file `basic_ios.h`.

5.381.2.5 `template<typename _CharT, typename _Traits> typedef
num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>
> std::basic_ostream<_CharT, _Traits>::__num_put_type
[inherited]`

These are non-standard types.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Definition at line 70 of file ostream.

5.381.2.6 `template<typename _CharT, typename _Traits> typedef _CharT
std::basic_istream<_CharT, _Traits>::char_type [inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istreamstream<_CharT, _Traits, _Alloc>](#).

Definition at line 59 of file istream.

5.381.2.7 `template<typename _CharT, typename _Traits, typename _Alloc>
typedef _CharT std::basic_stringstream<_CharT, _Traits, _Alloc>
>::char_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits>](#).

Definition at line 480 of file sstream.

5.381.2.8 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)
[inherited]`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the [ios_base](#) object.

int The integer provided when the callback was registered.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2139

Event callbacks are user defined functions that get called during several [ios_base](#) and [basic_ios](#) functions, specifically [imbue\(\)](#), [copyfmt\(\)](#), and [~ios\(\)](#).

Definition at line 435 of file `ios_base.h`.

5.381.2.9 `typedef _Ios_Fmtflags std::ios_base::fmtflags` [inherited]

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 254 of file `ios_base.h`.

5.381.2.10 `template<typename _CharT, typename _Traits> typedef
_Traits::int_type std::basic_istream< _CharT, _Traits >::int_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istreamstream< _CharT, _Traits, _Alloc >](#).

Definition at line 60 of file istream.

5.381.2.11 `template<typename _CharT , typename _Traits , typename _Alloc
> typedef traits_type::int_type std::basic_stringstream< _CharT,
_Traits, _Alloc >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream< _CharT, _Traits >](#).

Definition at line 485 of file sstream.

5.381.2.12 `typedef _Ios_Iostate std::ios_base::iostate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file `ios_base.h`.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2141

5.381.2.13 `template<typename _CharT, typename _Traits, typename _Alloc> typedef traits_type::off_type std::basic_stringstream<_CharT, _Traits, _Alloc>::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits>](#).

Definition at line 487 of file `sstream`.

5.381.2.14 `template<typename _CharT, typename _Traits> typedef _Traits::off_type std::basic_istream<_CharT, _Traits>::off_type` **[inherited]**

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istream<_CharT, _Traits, _Alloc>](#).

Definition at line 62 of file `istream`.

5.381.2.15 `typedef _Ios_Openmode std::ios_base::openmode` **[inherited]**

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 360 of file `ios_base.h`.

5.381.2.16 `template<typename _CharT, typename _Traits, typename _Alloc
> typedef traits_type::pos_type std::basic_stringstream< _CharT,
_Traits, _Alloc >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_istream< _CharT, _Traits >](#).

Definition at line 486 of file sstream.

5.381.2.17 `template<typename _CharT, typename _Traits> typedef
_Traits::pos_type std::basic_istream< _CharT, _Traits >::pos_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Reimplemented in [std::basic_ifstream< _CharT, _Traits >](#), and [std::basic_istream< _CharT, _Traits, _Alloc >](#).

Definition at line 61 of file istream.

5.381.2.18 `typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`

This is an enumerated type.

_Ios_Seekdir is implementation-defined. Defined values of type *seekdir* are:

- *beg*
- *cur*, equivalent to *SEEK_CUR* in the C standard library.
- *end*, equivalent to *SEEK_END* in the C standard library.

Definition at line 392 of file ios_base.h.

5.381.2.19 `template<typename _CharT, typename _Traits> typedef
_Traits std::basic_istream< _CharT, _Traits >::traits_type
[inherited]`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2143

Reimplemented from [std::basic_ios<_CharT, _Traits>](#).

Reimplemented in [std::basic_ifstream<_CharT, _Traits>](#), and [std::basic_istream<_CharT, _Traits, _Alloc>](#).

Definition at line 63 of file `istream`.

5.381.2.20 `template<typename _CharT, typename _Traits, typename _Alloc
> typedef _Traits std::basic_stringstream<_CharT, _Traits, _Alloc
>::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependant on) the template parameters, which are specific to the implementation.

Reimplemented from [std::basic_iostream<_CharT, _Traits>](#).

Definition at line 481 of file `sstream`.

5.381.3 Member Enumeration Documentation

5.381.3.1 `enum std::ios_base::event [inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during [imbue\(\)](#). `copyfmt_event` is used during `copyfmt()`.

Definition at line 418 of file `ios_base.h`.

5.381.4 Constructor & Destructor Documentation

5.381.4.1 `template<typename _CharT, typename _Traits, typename
_Alloc> std::basic_stringstream<_CharT, _Traits, _Alloc
>::basic_stringstream (ios_base::openmode __m =
ios_base::out | ios_base::in) [inline, explicit]`

Default constructor starts with an empty string buffer.

Parameters

mode Whether the buffer can read, or write, or both.

Initializes `sb` using `mode`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 510 of file sstream.

```
5.381.4.2  template<typename _CharT , typename _Traits , typename
             _Alloc > std::basic_stringstream< _CharT, _Traits, _Alloc
             >::basic_stringstream ( const __string_type & __str,
             ios_base::openmode __m = ios_base::out | ios_base::in )
             [inline, explicit]
```

Starts with an existing string buffer.

Parameters

str A string to copy as a starting buffer.

mode Whether the buffer can read, or write, or both.

Initializes *sb* using *str* and *mode*, and passes *&sb* to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 526 of file sstream.

```
5.381.4.3  template<typename _CharT , typename _Traits , typename
             _Alloc > std::basic_stringstream< _CharT, _Traits, _Alloc
             >::~~basic_stringstream ( ) [inline]
```

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 537 of file sstream.

5.381.5 Member Function Documentation

```
5.381.5.1  const locale& std::ios_base::_M_getloc ( ) const [inline,
             inherited]
```

Locale access.

Returns

A reference to the current locale.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2145

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 705 of file `ios_base.h`.

Referenced by `std::money_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::time_put<_CharT, _OutIter>::do_put()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::time_put<_CharT, _OutIter>::put()`.

5.381.5.2 `template<typename _CharT, typename _Traits> void
std::basic_ostream<_CharT, _Traits>::_M_write (const char_type
* __s, streamsize __n) [inline, inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 287 of file `ostream`.

5.381.5.3 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits>::bad () const [inline, inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

5.381.5.4 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::clear (iostate __state = goodbit)
[inherited]`

[Re]sets the error state.

Parameters

state The new state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values. Most users will not need to pass an argument.

Definition at line 40 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdbuf()`.

5.381.5.5 `template<typename _CharT, typename _Traits> basic_ios<
_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
const basic_ios<_CharT, _Traits> & __rhs) [inherited]`

Copies fields of `__rhs` into this.

Parameters

__rhs The source values for the copies.

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy [exceptions\(\)](#).

Definition at line 62 of file `basic_ios.tcc`.

References `std::basic_ios<_CharT, _Traits>::exceptions()`, `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::getloc()`, `std::ios_base::precision()`, `std::basic_ios<_CharT, _Traits>::tie()`, and `std::ios_base::width()`.

5.381.5.6 `template<typename _CharT, typename _Traits> bool std::basic_ios<
_CharT, _Traits>::eof () const [inline, inherited]`

Fast error checking.

5.381 `std::basic_stringstream< _CharT, _Traits, _Alloc >` Class Template Reference

2147

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.381.5.7 `template<typename _CharT, typename _Traits> iostate
std::basic_ios< _CharT, _Traits >::exceptions () const [inline,
inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of [exceptions\(iostate\)](#) for the meaning of the return value.

Definition at line 212 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`.

5.381.5.8 `template<typename _CharT, typename _Traits> void std::basic_ios<
_CharT, _Traits >::exceptions (iostate __except) [inline,
inherited]`

Throwing exceptions on errors.

Parameters

except The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type [std::ios_base::failure](#) is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
```

```

#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}

```

Definition at line 247 of file `basic_ios.h`.

5.381.5.9 `template<typename _CharT, typename _Traits> bool std::basic_ios<_CharT, _Traits>::fail() const [inline, inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 191 of file `basic_ios.h`.

Referenced by `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::regex_traits< _Ch_type >::value()`.

5.381.5.10 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::fill() const [inline, inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2149

Definition at line 360 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.381.5.11 `template<typename _CharT, typename _Traits> char_type
std::basic_ios<_CharT, _Traits>::fill (char_type __ch)
[inline, inherited]`

Sets a new *empty* character.

Parameters

ch The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 380 of file `basic_ios.h`.

5.381.5.12 `fmtflags std::ios_base::flags () const [inline, inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::basic_ostream<_CharT, _Traits>::operator<<()`, `std::operator<<()`, and `std::operator>>()`.

5.381.5.13 `fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline,
inherited]`

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.381.5.14 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

Referenced by `std::flush()`.

5.381.5.15 `template<typename _CharT, typename _Traits> streamsize std::basic_istream<_CharT, _Traits>::gcount () const [inline, inherited]`

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 249 of file istream.

5.381.5.16 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::get (void) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 236 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.381.5.17 `template<typename _CharT, typename _Traits> basic_istream< _CharT, _Traits> & std::basic_istream< _CharT, _Traits>::get (char_type & __c) [inherited]`

Simple extraction.

Parameters

c The character in which to store data.

Returns

`*this`

Tries to extract a character and store it in *c*. If none are available, sets failbit and returns `traits::eof()`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.381.5.18 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits>::get (char_type * __s, streamsize __n) [inline, inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

Returns

*this

Returns `get(s,n,widen('\n'))`.

Definition at line 333 of file `istream`.

5.381.5.19 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (__streambuf_type & __sb, char_type __delim) [inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

delim A "stop" character.

Returns

*this

Characters are extracted and inserted into *sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, `std::basic_streambuf<_CharT, _Traits>::snextc()`, and `std::basic_streambuf<_CharT, _Traits>::sputc()`.

5.381.5.20 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::get (__streambuf_type &
__sb) [inline, inherited]`

Extraction into another streambuf.

Parameters

sb A streambuf in which to store data.

Returns

*this

Returns `get(sb, widen('\n'))`.

Definition at line 366 of file istream.

5.381.5.21 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get
(char_type * __s, streamsize __n, char_type __delim)
[inherited]`

Simple multiple-character extraction.

Parameters

s Pointer to an array.

n Maximum number of characters to store in *s*.

delim A "stop" character.

Returns

*this

Characters are extracted and stored into *s* until one of the following happens:

- *n*−1 characters are stored
- the input sequence reaches EOF
- the next character equals *delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.381.5.22 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n, char_type __delim)`
[**inherited**]

String extraction.

Parameters

s A character array in which to store the data.

n Maximum number of characters to extract.

delim A "stop" character.

Returns

*this

Extracts and stores characters into *s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals *delim*, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. *n*-1 characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2155

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.381.5.23 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream<_CharT, _Traits>::getline (char_type * __s, streamsize __n) [inline, inherited]`

String extraction.

Parameters

s A character array in which to store the data.

n Maximum number of characters to extract.

Returns

`*this`

Returns `getline(s,n,widen('\n'))`.

Definition at line 406 of file `istream`.

Referenced by `std::basic_istream<char>::getline()`.

5.381.5.24 `locale std::ios_base::getloc () const [inline, inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, `std::basic_ios<_CharT, _Traits>::imbue()`, `std::operator>>()`, and `std::ws()`.

5.381.5.25 `template<typename _CharT, typename _Traits> bool
std::basic_ios< _CharT, _Traits >::good () const [inline,
inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 170 of file `basic_ios.h`.

5.381.5.26 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(streamsize __n) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 493 of file `istream.tcc`.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, `std::basic_streambuf< _CharT, _Traits >::sgetc()`, and `std::basic_streambuf< _CharT, _Traits >::snextc()`.

5.381.5.27 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore
(void) [inherited]`

Discarding characters.

Parameters

n Number of characters to discard.

delim A "stop" character.

Returns

`*this`

Extracts characters and throws them away until one of the following happens:

- if `n != std::numeric_limits<int>::max()`, `n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `delim` (in this case, the character is extracted); note that this condition will never occur if `delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 460 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.381.5.28 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (streamsize __n, int_type __delim) [inherited]`

Simple extraction.

Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 555 of file istream.tcc.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_streambuf<_CharT, _Traits>::sbumpc()`, `std::basic_ios<_CharT, _Traits>::setstate()`, `std::basic_streambuf<_CharT, _Traits>::sgetc()`, and `std::basic_streambuf<_CharT, _Traits>::snextc()`.

5.381.5.29 `template<typename _CharT, typename _Traits> locale std::basic_ios<_CharT, _Traits>::imbue (const locale & __loc) [inherited]`

Moves to a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Reimplemented from `std::ios_base`.

Definition at line 113 of file `basic_ios.tcc`.

References `std::ios_base::getloc()`, and `std::basic_ios<_CharT, _Traits>::rdbuf()`.

Referenced by `std::operator<<()`.

5.381.5.30 `template<typename _CharT, typename _Traits> void
std::basic_ios<_CharT, _Traits>::init (basic_streambuf<
_CharT, _Traits> * __sb) [protected, inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 125 of file `basic_ios.tcc`.

References `std::ios_base::badbit`, and `std::ios_base::goodbit`.

5.381.5.31 `long& std::ios_base::iword (int __ix) [inline, inherited]`

Access to integer array.

Parameters

__ix Index into the array.

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2159

Definition at line 740 of file `ios_base.h`.

5.381.5.32 `template<typename _CharT, typename _Traits> char
std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char
__dfault) const [inline, inherited]`

Squeezes characters.

Parameters

c The character to narrow.

dfault The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 420 of file `basic_ios.h`.

5.381.5.33 `template<typename _CharT, typename _Traits> std::basic_ios<
_CharT, _Traits>::operator void * () const [inline,
inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 111 of file `basic_ios.h`.

5.381.5.34 `template<typename _CharT, typename _Traits> bool
std::basic_ios<_CharT, _Traits>::operator! () const
[inline, inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 115 of file `basic_ios.h`.

5.381.5.35 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (float __f)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 213 of file ostream.

5.381.5.36 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (bool __n)
[inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 173 of file ostream.

5.381.5.37 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
short __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 180 of file ostream.

5.381.5.38 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 108 of file ostream.

5.381.5.39 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 117 of file ostream.

5.381.5.40 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the `iomanip` header.

Definition at line 127 of file ostream.

5.381.5.41 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 169 of file ostream.

5.381.5.42 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (short __n) [inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.381.5.43 `template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (int __n) [inherited]`

Basic arithmetic inserters.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2163

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::oct`.

5.381.5.44 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 200 of file `ostream`.

5.381.5.45 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (const void * __p) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 225 of file ostream.

5.381.5.46 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (double __f
) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 209 of file ostream.

5.381.5.47 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (long double
__f) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the [num_get](#) facet) to perform numeric formatting.

Definition at line 221 of file ostream.

5.381.5.48 `template<typename _CharT , typename _Traits > basic_ostream<
_CharT, _Traits > & std::basic_ostream< _CharT, _Traits
>::operator<< (__streambuf_type * __sb) [inherited]`

Extracting from another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from *sb* and inserted into **this* until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from *sb*, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.381.5.49 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 165 of file ostream.

5.381.5.50 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int
__n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 191 of file ostream.

5.381.5.51 `template<typename _CharT, typename _Traits> __ostream_type&
std::basic_ostream< _CharT, _Traits >::operator<< (unsigned
long long __n) [inline, inherited]`

Basic arithmetic inserters.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 204 of file ostream.

5.381.5.52 `template<typename _CharT , typename _Traits > basic_istream<
_CharT, _Traits > & std::basic_istream< _CharT, _Traits
>::operator>> (short & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.381.5.53 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (bool & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file istream.

5.381.5.54 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 203 of file istream.

5.381.5.55 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (ios_base
&(*)(ios_base &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

5.381.5.56 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (double &
__f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 207 of file `istream`.

5.381.5.57 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (long double
& __f) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*`this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 211 of file `istream`.

5.381.5.58 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (unsigned
short & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 174 of file istream.

5.381.5.59 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (long & __n
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 185 of file istream.

5.381.5.60 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream<_CharT, _Traits>::operator>> (void *& __p
) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 215 of file `istream`.

5.381.5.61 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 194 of file `istream`.

5.381.5.62 `template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::failbit`, `std::num_get< _CharT, _InIter >::get()`, `std::ios_base::goodbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.381 std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference **2171**

5.381.5.63 `template<typename _CharT, typename _Traits > basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]`

Extracting into another streambuf.

Parameters

sb A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If *sb* is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the *sb* streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.381.5.64 `template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 181 of file istream.

5.381.5.65 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (
__istream_type &(*)(__istream_type &) __pf) [inline,
inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 120 of file `istream`.

5.381.5.66 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (__ios_type
&(*)(__ios_type &) __pf) [inline, inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `ioanip` header.

Definition at line 124 of file `istream`.

5.381.5.67 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
long & __n) [inline, inherited]`

Basic arithmetic extractors.

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 189 of file `istream`.

5.381.5.68 `template<typename _CharT, typename _Traits> __istream_type&
std::basic_istream< _CharT, _Traits >::operator>> (unsigned
long long & __n) [inline, inherited]`

Basic arithmetic extractors.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2173

Parameters

A variable of builtin type.

Returns

**this* if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 198 of file `istream`.

5.381.5.69 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek(void) [inherited]`

Looking ahead in the stream.

Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.381.5.70 `streamsize std::ios_base::precision() const [inline, inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, and `std::operator<<()`.

5.381.5.71 `streamsize std::ios_base::precision (streamsize __prec)`
`[inline, inherited]`

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of `precision()`.

Definition at line 629 of file `ios_base.h`.

5.381.5.72 `template<typename _CharT, typename _Traits > basic_ostream<`
`_CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (`
`char_type __c) [inherited]`

Simple insertion.

Parameters

c The character to insert.

Returns

`*this`

Tries to insert *c*.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

Referenced by `std::endl()`, and `std::ends()`.

5.381.5.73 `template<typename _CharT, typename _Traits > basic_istream<`
`_CharT, _Traits > & std::basic_istream< _CharT, _Traits`
`>::putback (char_type __c) [inherited]`

Unextracting a single character.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2175

Parameters

c The character to push back into the input stream.

Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

Note

Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::eof()`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, `std::basic_ios<_CharT, _Traits>::setstate()`, and `std::basic_streambuf<_CharT, _Traits>::sputbackc()`.

Referenced by `std::operator>>()`.

5.381.5.74 `void*& std::ios_base::pword (int __ix) [inline, inherited]`

Access to void pointer array.

Parameters

__ix Index into the array.

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file `ios_base.h`.

5.381.5.75 `template<typename _CharT, typename _Traits, typename _Alloc
> __stringbuf_type* std::basic_stringstream< _CharT, _Traits,
_Alloc >::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current [basic_stringbuf](#) buffer.

This hides both signatures of [std::basic_ios::rdbuf\(\)](#).

Reimplemented from [std::basic_ios< _CharT, _Traits >](#).

Definition at line 548 of file sstream.

5.381.5.76 `template<typename _CharT, typename _Traits> basic_streambuf<
_CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
basic_streambuf< _CharT, _Traits > * __sb) [inherited]`

Changing the underlying buffer.

Parameters

sb The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument [rdbuf\(\)](#), which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```

Definition at line 52 of file basic_ios.tcc.

References [std::basic_ios< _CharT, _Traits >::clear\(\)](#).

5.381 std::basic_stringstream<_CharT, _Traits, _Alloc> Class Template Reference 2177

5.381.5.77 `template<typename _CharT, typename _Traits> istate
std::basic_ios<_CharT, _Traits>::rdstate() const [inline,
inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See [std::ios_base::iostate](#) for the possible bit values. Most users will call one of the interpreting wrappers, e.g., [good\(\)](#).

Definition at line 127 of file `basic_ios.h`.

5.381.5.78 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::read(
char_type * __s, streamsize __n) [inherited]`

Extraction without delimiters.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

*this

If the stream state is [good\(\)](#), extracts characters and stores them into *s* until one of the following happens:

- *n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file `istream.tcc`.

References `std::basic_istream<_CharT, _Traits>::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.381.5.79 `template<typename _CharT, typename _Traits> streamsize
std::basic_istream< _CharT, _Traits >::readsome (char_type *
__s, streamsize __n) [inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

- s* A character array.
- n* Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called *A* here:

- if *A* == -1, sets eofbit and extracts no characters
- if *A* == 0, extracts no characters
- if *A* > 0, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, `std::min()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

5.381.5.80 `void std::ios_base::register_callback (event_callback __fn, int
__index) [inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

- __fn* The function to add.
- __index* The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

5.381.5.81 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg
(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current read position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 870 of file istream.tcc.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.381.5.82 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg
(pos_type __pos) [inherited]`

Changing the current read position.

Parameters

pos A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 837 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::in`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.381.5.83 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(off_type __off, ios_base::seekdir __dir) [inherited]`

Changing the current write position.

Parameters

off A file offset object.

dir The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.381.5.84 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp(pos_type __pos) [inherited]`

Changing the current write position.

Parameters

pos A file position object.

Returns

*this

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference 2181

If `fail()` is not true, calls `rdbuf()` -> `pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::ios_base::out`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.381.5.85 `fmtflags std::ios_base::setf(fmtflags __fmtfl) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.381.5.86 `fmtflags std::ios_base::setf(fmtflags __fmtfl, fmtflags __mask) [inline, inherited]`

Setting new format flags.

Parameters

fmtfl Additional flags to set.

mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.381.5.87 `template<typename _CharT, typename _Traits> void
std::basic_ios< _CharT, _Traits >::setstate (iostate __state)
[inline, inherited]`

Sets additional flags in the error state.

Parameters

state The additional state flag(s) to set.

See [std::ios_base::iostate](#) for the possible bit values.

Definition at line 147 of file `basic_ios.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::operator>>()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::unget()`, and `std::ws()`.

5.381.5.88 `template<typename _CharT , typename _Traits , typename _Alloc
> __string_type std::basic_stringstream< _CharT, _Traits, _Alloc
>::str () const [inline]`

Copying out the string buffer.

Returns

`rdbuf() -> str()`

Definition at line 556 of file `sstream`.

5.381.5.89 `template<typename _CharT , typename _Traits , typename _Alloc
> void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (
const __string_type & __s) [inline]`

Setting a new buffer.

Parameters

s The string to use as a new sequence.

5.381 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

2183

Calls `rdbuf()` \rightarrow `str(s)`.

Definition at line 566 of file `sstream`.

5.381.5.90 `template<typename _CharT, typename _Traits> int
std::basic_istream<_CharT, _Traits>::sync (void)
[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()` \rightarrow `pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 777 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::goodbit`, `std::basic_streambuf<_CharT, _Traits>::pubsync()`, `std::basic_ios<_CharT, _Traits>::rdbuf()`, and `std::basic_ios<_CharT, _Traits>::setstate()`.

5.381.5.91 `static bool std::ios_base::sync_with_stdio (bool __sync = true)
[static, inherited]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.381.5.92 `template<typename _CharT, typename _Traits> basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg(void) [inherited]`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf()->pubseekoff(0, cur, in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 813 of file `istream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::in`, and `std::basic_ios<_CharT, _Traits>::rddbuf()`.

5.381.5.93 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>::pos_type std::basic_ostream<_CharT, _Traits>::tellp() [inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rddbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

References `std::ios_base::badbit`, `std::ios_base::cur`, `std::basic_ios<_CharT, _Traits>::fail()`, `std::ios_base::out`, and `std::basic_ios<_CharT, _Traits>::rddbuf()`.

5.381.5.94 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits>* __tiestr) [inline, inherited]`

Ties this stream to an output stream.

Parameters

tiestr The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see [tie\(\)](#) for more.

Definition at line 297 of file basic_ios.h.

5.381.5.95 `template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits
>::tie() const [inline, inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, [std::cin](#) is tied to [std::cout](#).

Definition at line 285 of file basic_ios.h.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`.

5.381.5.96 `template<typename _CharT, typename _Traits> basic_istream<
_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget
(void) [inherited]`

Unextracting the previous character.

Returns

**this*

If [rdbuf\(\)](#) is not null, calls [rdbuf\(\)->sungetc\(c\)](#).

If [rdbuf\(\)](#) is null or if [sungetc\(\)](#) fails, sets badbit in the error state.

Note

Since no characters are extracted, the next call to [gcount\(\)](#) will return 0, as required by DR 60.

Definition at line 744 of file istream.tcc.

References `std::basic_istream< _CharT, _Traits >::_M_gcount`, `std::ios_base::badbit`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::ios_base::goodbit`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_streambuf< _CharT, _Traits >::sungetc()`.

5.381.5.97 `void std::ios_base::unsetf (fmtflags __mask) [inline, inherited]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file ios_base.h.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.381.5.98 `template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char __c) const [inline, inherited]`

Widens characters.

Parameters

c The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localizati>

Definition at line 439 of file basic_ios.h.

Referenced by `std::endl()`, `std::getline()`, and `std::operator>>()`.

5.381 std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference **2187**

5.381.5.99 `streamsize std::ios_base::width (streamsize __wide) [inline, inherited]`

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of `width()`.

Definition at line 652 of file `ios_base.h`.

5.381.5.100 `streamsize std::ios_base::width () const [inline, inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::copyfmt()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::operator>>()`.

5.381.5.101 `template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inherited]`

Character string insertion.

Parameters

s The array to insert.

n Maximum number of characters to insert.

Returns

`*this`

Characters are copied from *s* and inserted into the stream until one of the following happens:

- n characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

5.381.5.102 `static int std::ios_base::xalloc () throw () [static, inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.381.6 Member Data Documentation

5.381.6.1 `template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected, inherited]`

The number of characters extracted in the previous unformatted function; see [gcount\(\)](#).

Definition at line 79 of file `istream`.

Referenced by `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.381.6.2 `const fmtflags std::ios_base::adjustfield [static, inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

5.381 `std::basic_stringstream< _CharT, _Traits, _Alloc >` Class Template Reference 2189

Definition at line 309 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`.

5.381.6.3 `const openmode std::ios_base::app` **[static, inherited]**

Seek to end before each write.

Definition at line 363 of file `ios_base.h`.

5.381.6.4 `const openmode std::ios_base::ate` **[static, inherited]**

Open and seek to end immediately after opening.

Definition at line 366 of file `ios_base.h`.

Referenced by `std::basic_filebuf< _CharT, _Traits >::open()`.

5.381.6.5 `const iostate std::ios_base::badbit` **[static, inherited]**

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 333 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::flush()`, `std::basic_istream< _CharT, _Traits >::get()`, `std::basic_istream< _CharT, _Traits >::getline()`, `std::basic_istream< _CharT, _Traits >::ignore()`, `std::basic_ios< _CharT, _Traits >::init()`, `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::operator<<()`, `std::operator>>()`, `std::basic_istream< _CharT, _Traits >::operator>>()`, `std::basic_istream< _CharT, _Traits >::peek()`, `std::basic_ostream< _CharT, _Traits >::put()`, `std::basic_istream< _CharT, _Traits >::putback()`, `std::basic_istream< _CharT, _Traits >::read()`, `std::basic_istream< _CharT, _Traits >::readsome()`, `std::basic_istream< _CharT, _Traits >::seekg()`, `std::basic_ostream< _CharT, _Traits >::seekp()`, `std::basic_istream< _CharT, _Traits >::sync()`, `std::basic_istream< _CharT, _Traits >::tellg()`, `std::basic_ostream< _CharT, _Traits >::tellp()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

5.381.6.6 `const fmtflags std::ios_base::basefield` **[static, inherited]**

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 312 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, and `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.381.6.7 const seekdir std::ios_base::beg [static, inherited]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file ios_base.h.

5.381.6.8 const openmode std::ios_base::binary [static, inherited]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see /// <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.381.6.9 const fmtflags std::ios_base::boolalpha [static, inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.381.6.10 const seekdir std::ios_base::cur [static, inherited]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.381.6.11 const fmtflags std::ios_base::dec [static, inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file ios_base.h.

5.381.6.12 const seekdir std::ios_base::end [static, inherited]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.381.6.13 const iostate std::ios_base::eofbit [static, inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.381.6.14 const iostate std::ios_base::failbit [static, inherited]

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.381.6.15 const fmtflags std::ios_base::fixed [static, inherited]

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.381.6.16 const fmtflags std::ios_base::floatfield [static, inherited]

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 315 of file ios_base.h.

5.381.6.17 const iostate std::ios_base::goodbit [static, inherited]

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.381.6.18 const fmtflags std::ios_base::hex [static, inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.381.6.19 const openmode std::ios_base::in [static, inherited]

Open for input. Default for ifstream and fstream.

Definition at line 374 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.381.6.20 const fmtflags std::ios_base::internal [static, inherited]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 271 of file ios_base.h.

5.381.6.21 const fmtflags std::ios_base::left [static, inherited]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.381.6.22 const fmtflags std::ios_base::oct [static, inherited]

Converts integer input or generates integer output in octal base.

Definition at line 278 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

5.381.6.23 const openmode std::ios_base::out [static, inherited]

Open for output. Default for ofstream and fstream.

Definition at line 377 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.381.6.24 const fmtflags std::ios_base::right [static, inherited]

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file ios_base.h.

5.381.6.25 const fmtflags std::ios_base::scientific [static, inherited]

Generates floating-point output in scientific notation.

Definition at line 285 of file ios_base.h.

5.381.6.26 const fmtflags std::ios_base::showbase [static, inherited]

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file ios_base.h.

5.381.6.27 const fmtflags std::ios_base::showpoint [static, inherited]

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file ios_base.h.

5.381.6.28 const fmtflags std::ios_base::showpos [static, inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file ios_base.h.

5.381.6.29 const fmtflags std::ios_base::skipws [static, inherited]

Skips leading white space before certain input operations.

Definition at line 299 of file ios_base.h.

5.381.6.30 const openmode std::ios_base::trunc [static, inherited]

Open for input. Default for ofstream.

Definition at line 380 of file ios_base.h.

5.381.6.31 const fmtflags std::ios_base::unitbuf [static, inherited]

Flushes output after each output operation.

Definition at line 302 of file ios_base.h.

5.381.6.32 const fmtflags std::ios_base::uppercase [static, inherited]

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

The documentation for this class was generated from the following file:

- [sstream](#)

5.382 std::bernoulli_distribution Class Reference

A Bernoulli random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef bool [result_type](#)

Public Member Functions

- [bernoulli_distribution](#) (double __p=0.5)
- **bernoulli_distribution** (const [param_type](#) &__p)
- [result_type](#) max () const
- [result_type](#) min () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator**() (_UniformRandomNumberGenerator &__urng, const
[param_type](#) &__p)
- double [p](#) () const
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- void [reset](#) ()

5.382.1 Detailed Description

A Bernoulli random number distribution. Generates a sequence of true and false values with likelihood p that true will come up and $(1 - p)$ that false will appear.

Definition at line 3243 of file random.h.

5.382.2 Member Typedef Documentation

5.382.2.1 typedef bool std::bernoulli_distribution::result_type

The type of the range of the distribution.

Definition at line 3247 of file random.h.

5.382.3 Constructor & Destructor Documentation

5.382.3.1 `std::bernoulli_distribution::bernoulli_distribution (double __p = 0.5) [inline, explicit]`

Constructs a Bernoulli distribution with likelihood `p`.

Parameters

`__p` [IN] The likelihood of a true result being returned. Must be in the interval `[0, 1]`.

Definition at line 3280 of file `random.h`.

5.382.4 Member Function Documentation

5.382.4.1 `result_type std::bernoulli_distribution::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3330 of file `random.h`.

5.382.4.2 `result_type std::bernoulli_distribution::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3323 of file `random.h`.

5.382.4.3 `template<typename _UniformRandomNumberGenerator> result_type std::bernoulli_distribution::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3338 of file `random.h`.

References `operator()()`, and `param()`.

Referenced by `operator()()`.

5.382.4.4 `double std::bernoulli_distribution::p () const [inline]`

Returns the `p` parameter of the distribution.

Definition at line 3301 of file `random.h`.

5.382.4.5 `void std::bernoulli_distribution::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 3316 of file `random.h`.

5.382.4.6 `param_type std::bernoulli_distribution::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 3308 of file `random.h`.

Referenced by `operator()()`, `std::operator==()`, and `std::operator>>()`.

5.382.4.7 `void std::bernoulli_distribution::reset () [inline]`

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3295 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

5.383 `std::bernoulli_distribution::param_type` Struct Reference

Public Types

- typedef [bernoulli_distribution](#) `distribution_type`

Public Member Functions

- `param_type` (double `__p`=0.5)
- double `p` () const

Friends

- `bool operator==(const param_type &__p1, const param_type &__p2)`

5.383.1 Detailed Description

Parameter type.

Definition at line 3249 of file `random.h`.

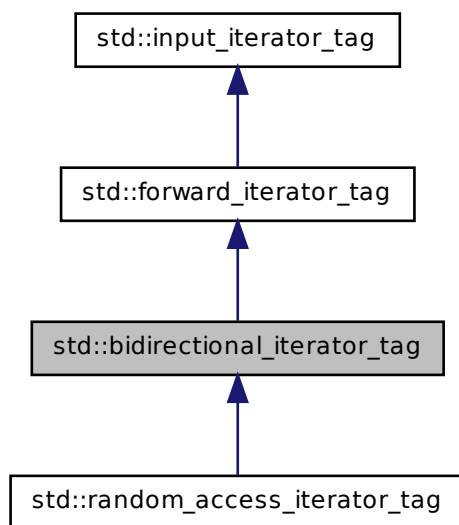
The documentation for this struct was generated from the following file:

- [random.h](#)

5.384 `std::bidirectional_iterator_tag` Struct Reference

Bidirectional iterators support a superset of forward iterator /// operations.

Inheritance diagram for `std::bidirectional_iterator_tag`:



5.385 std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference

5.384.1 Detailed Description

Bidirectional iterators support a superset of forward iterator /// operations.

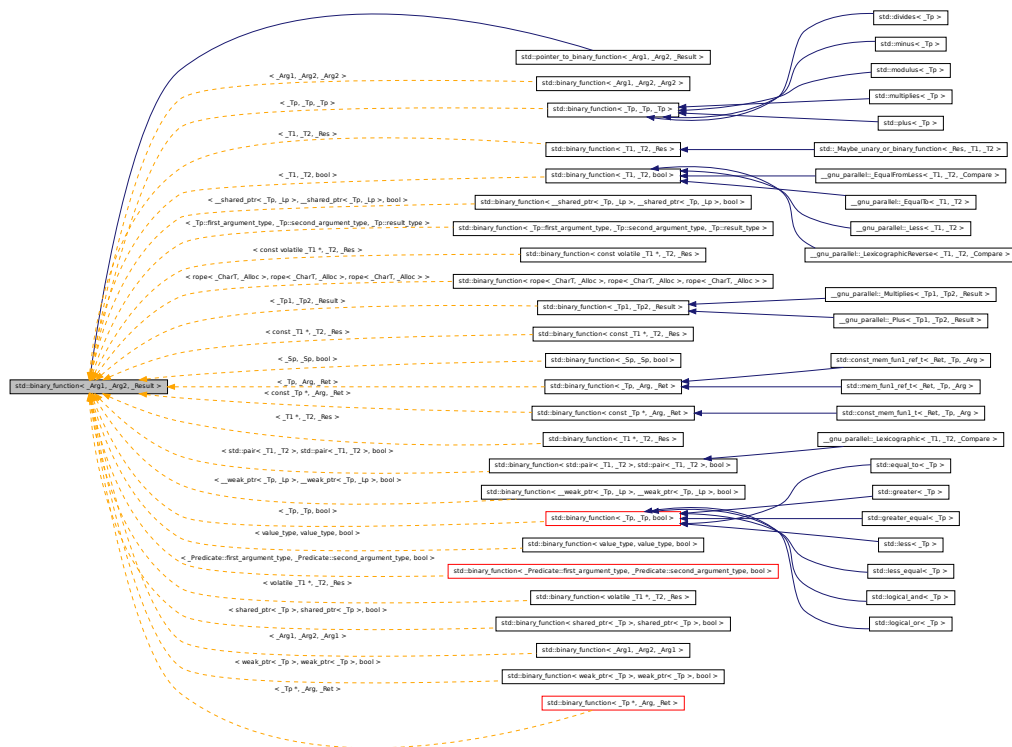
Definition at line 98 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.385 std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference

Inheritance diagram for std::binary_function< _Arg1, _Arg2, _Result >:



Public Types

- typedef `_Arg1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Arg2` [second_argument_type](#)

5.385.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result> struct
std::binary_function< _Arg1, _Arg2, _Result >
```

This is one of the [functor base classes](#).

Definition at line 112 of file `stl_function.h`.

5.385.2 Member Typedef Documentation

5.385.2.1 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`
`>::first_argument_type`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.385.2.2 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`
`>::result_type`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.385.2.3 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`
`>::second_argument_type`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.386 std::binary_negate< _Predicate > Class Template Reference

One of the [negation functors](#).

Inheritance diagram for std::binary_negate< _Predicate >:



Public Types

- typedef _Predicate::first_argument_type [first_argument_type](#)
- typedef bool [result_type](#)
- typedef _Predicate::second_argument_type [second_argument_type](#)

Public Member Functions

- **binary_negate** (const _Predicate &__x)
- bool **operator()** (const typename _Predicate::first_argument_type &__x, const typename _Predicate::second_argument_type &__y) const

Protected Attributes

- _Predicate **_M_pred**

5.386.1 Detailed Description

```
template<typename _Predicate> class std::binary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 369 of file stl_function.h.

5.386.2 Member Typedef Documentation

5.386.2.1 typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::first_argument_type **[inherited]**

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.386.2.2 `typedef bool std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::result_type [inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.386.2.3 `typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

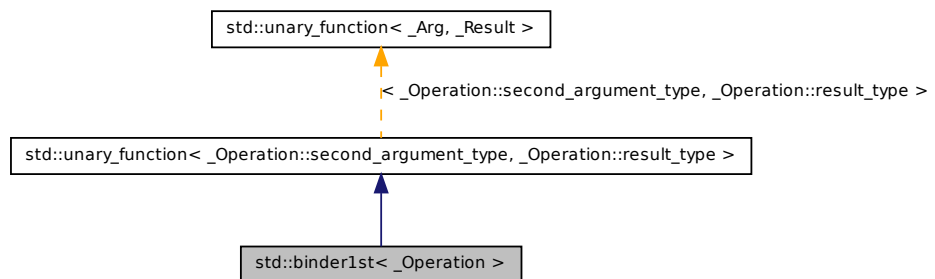
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.387 std::binder1st< _Operation > Class Template Reference

One of the [binder functors](#).

Inheritance diagram for std::binder1st< _Operation >:



Public Types

- typedef _Operation::second_argument_type [argument_type](#)
- typedef _Operation::result_type [result_type](#)

Public Member Functions

- **binder1st** (const _Operation &__x, const typename _Operation::first_argument_type &__y)
- _Operation::result_type **operator()** (typename _Operation::second_argument_type &__x) const
- _Operation::result_type **operator()** (const typename _Operation::second_argument_type &__x) const

Protected Attributes

- _Operation **op**
- _Operation::first_argument_type **value**

5.387.1 Detailed Description

template<typename _Operation> class std::binder1st< _Operation >

One of the [binder functors](#).

Definition at line 98 of file binders.h.

5.387.2 Member Typedef Documentation

5.387.2.1 typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type, _Operation::result_type >::argument_type **[inherited]**

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

5.387.2.2 typedef _Operation::result_type std::unary_function< _Operation::second_argument_type, _Operation::result_type >::result_type **[inherited]**

`result_type` is the return type

Definition at line 105 of file stl_function.h.

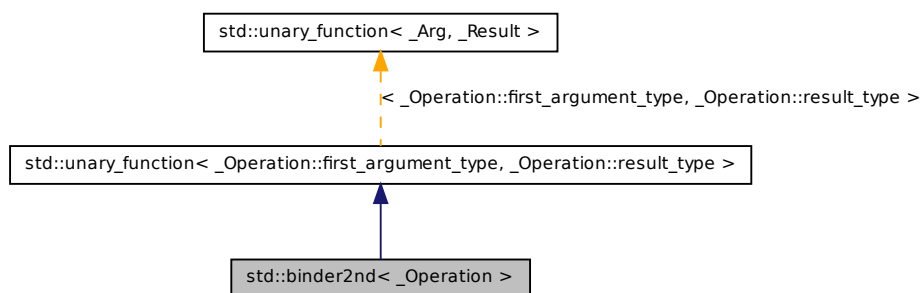
The documentation for this class was generated from the following file:

- [binders.h](#)

5.388 std::binder2nd< _Operation > Class Template Reference

One of the [binder functors](#).

Inheritance diagram for std::binder2nd< _Operation >:



Public Types

- typedef `_Operation::first_argument_type` [argument_type](#)
- typedef `_Operation::result_type` [result_type](#)

Public Member Functions

- **binder2nd** (`const _Operation &__x`, `const typename _Operation::second_argument_type &__y`)
- `_Operation::result_type` **operator()** (`typename _Operation::first_argument_type &__x`) `const`
- `_Operation::result_type` **operator()** (`const typename _Operation::first_argument_type &__x`) `const`

Protected Attributes

- `_Operation op`
- `_Operation::second_argument_type value`

5.388.1 Detailed Description

`template<typename _Operation> class std::binder2nd< _Operation >`

One of the [binder functors](#).

Definition at line 133 of file `binders.h`.

5.388.2 Member Typedef Documentation

5.388.2.1 `typedef _Operation::first_argument_type std::unary_function< _Operation::first_argument_type , _Operation::result_type >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.388.2.2 `typedef _Operation::result_type std::unary_function< _Operation::first_argument_type , _Operation::result_type >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

5.389 `std::binomial_distribution< _IntType >` Class Template Reference

A discrete binomial random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- **binomial_distribution** (`_IntType __t=_IntType(1), double __p=0.5`)
- **binomial_distribution** (const [param_type](#) &__p)
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng, const`
`param_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`)
- double **p** () const
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) **param** () const
- void **reset** ()
- `_IntType` **t** () const

Friends

- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >
[std::basic_ostream](#)< `_CharT`, `_Traits` > & **operator<<** ([std::basic_ostream](#)< `_`
`CharT`, `_Traits` > &, const [std::binomial_distribution](#)< `_IntType1` > &)
- template<typename `_IntType1` >
bool **operator==** (const [std::binomial_distribution](#)< `_IntType1` > &__d1, const
[std::binomial_distribution](#)< `_IntType1` > &__d2)
- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >
[std::basic_istream](#)< `_CharT`, `_Traits` > & **operator>>** ([std::basic_istream](#)< `_`
`CharT`, `_Traits` > &, [std::binomial_distribution](#)< `_IntType1` > &)

5.389.1 Detailed Description

```
template<typename _IntType = int> class std::binomial_distribution< _IntType
>
```

A discrete binomial random number distribution. The formula for the binomial probability density function is $p(i|t, p) = \binom{n}{i} p^i (1 - p)^{t-i}$ where t and p are the parameters of the distribution.

Definition at line 3420 of file random.h.

5.389.2 Member Typedef Documentation

5.389.2.1 `template<typename _IntType = int> typedef _IntType
std::binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3427 of file `random.h`.

5.389.3 Member Function Documentation

5.389.3.1 `template<typename _IntType = int> result_type
std::binomial_distribution< _IntType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3530 of file `random.h`.

Referenced by `std::binomial_distribution< _IntType >::operator()`.

5.389.3.2 `template<typename _IntType = int> result_type
std::binomial_distribution< _IntType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3523 of file `random.h`.

5.389.3.3 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::binomial_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3538 of file `random.h`.

References `std::binomial_distribution< _IntType >::operator()`, and `std::binomial_distribution< _IntType >::param()`.

Referenced by `std::binomial_distribution< _IntType >::operator()`.

5.389.3.4 `template<typename _IntType > template<typename
_UniformRandomNumberGenerator > binomial_distribution<
_IntType >::result_type std::binomial_distribution< _IntType
>::operator() (_UniformRandomNumberGenerator & __urng,
const param_type & __param)`

A rejection algorithm when $t * p \geq 8$ and a simple waiting time method - the second in the referenced book - otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

Definition at line 1427 of file `random.tcc`.

References `std::abs()`, `std::log()`, and `std::binomial_distribution< _IntType >::max()`.

5.389.3.5 `template<typename _IntType = int> double
std::binomial_distribution< _IntType >::p () const [inline]`

Returns the distribution `p` parameter.

Definition at line 3501 of file `random.h`.

5.389.3.6 `template<typename _IntType = int> void std::binomial_
distribution< _IntType >::param (const param_type & __param)
[inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 3516 of file `random.h`.

5.389.3.7 `template<typename _IntType = int> param_type
std::binomial_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 3508 of file `random.h`.

Referenced by `std::binomial_distribution< _IntType >::operator()()`.

5.389.3.8 `template<typename _IntType = int> void
std::binomial_distribution< _IntType >::reset() [inline]`

Resets the distribution state.

Definition at line 3487 of file random.h.

References std::normal_distribution< _RealType >::reset().

5.389.3.9 `template<typename _IntType = int> _IntType
std::binomial_distribution< _IntType >::t() const [inline]`

Returns the distribution t parameter.

Definition at line 3494 of file random.h.

5.389.4 Friends And Related Function Documentation

5.389.4.1 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_ostream< _CharT ,
_Traits>& operator<<(std::basic_ostream< _CharT , _Traits > & ,
const std::binomial_distribution< _IntType1 > &) [friend]`

Inserts a binomial_distribution random number distribution __x into the output stream __os.

Parameters

__os An output stream.

__x A binomial_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

5.389.4.2 `template<typename _IntType = int> template<typename _IntType1
> bool operator==(const std::binomial_distribution< _IntType1 >
& __d1, const std::binomial_distribution< _IntType1 > & __d2)
[friend]`

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3553 of file random.h.

5.389.4.3 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> (std::basic_istream<_CharT, _Traits > & ,
std::binomial_distribution<_IntType1 > &) [friend]`

Extracts a `binomial_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `binomial_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.390 `std::binomial_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `binomial_distribution<_IntType>` `distribution_type`

Public Member Functions

- `param_type` (`_IntType __t=_IntType(1)`, `double __p=0.5`)
- `double p () const`
- `_IntType t () const`

Friends

- class `binomial_distribution<_IntType>`
- `bool operator==` (`const param_type &__p1`, `const param_type &__p2`)

5.390.1 Detailed Description

template<typename _IntType = int> struct std::binomial_distribution< _IntType >::param_type

Parameter type.

Definition at line 3429 of file random.h.

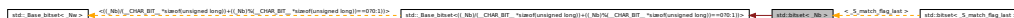
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.391 std::bitset< _Nb > Class Template Reference

The bitset class represents a *fixed-size* sequence of bits.

Inheritance diagram for std::bitset< _Nb >:



Classes

- class [reference](#)

Public Member Functions

- [bitset](#) ()
- [bitset](#) (unsigned long long __val)
- template<class _CharT, class _Traits, class _Alloc >
[bitset](#) (const [std::basic_string](#)< _CharT, _Traits, _Alloc > &__s, size_t __-
position, size_t __n)
- template<class _CharT, class _Traits, class _Alloc >
bitset (const [std::basic_string](#)< _CharT, _Traits, _Alloc > &__s, size_t __-
position, size_t __n, _CharT __zero, _CharT __one=_CharT('1'))
- template<class _CharT, class _Traits, class _Alloc >
[bitset](#) (const [std::basic_string](#)< _CharT, _Traits, _Alloc > &__s, size_t __-
position=0)
- [bitset](#) (const char *__str)
- size_t [Find_first](#) () const

- `size_t Find_next (size_t __prev) const`
- `template<class _CharT, class _Traits >`
`void M_copy_from_ptr (const _CharT *, size_t, size_t, size_t, _CharT, _-`
`CharT)`
- `template<class _CharT, class _Traits, class _Alloc >`
`void M_copy_from_string (const std::basic_string< _CharT, _Traits, _Alloc`
`> &__s, size_t __pos, size_t __n, _CharT __zero, _CharT __one)`
- `template<class _CharT, class _Traits, class _Alloc >`
`void M_copy_from_string (const std::basic_string< _CharT, _Traits, _Alloc`
`> &__s, size_t __pos, size_t __n)`
- `template<class _CharT, class _Traits, class _Alloc >`
`void M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &, _-`
`CharT, _CharT) const`
- `template<class _CharT, class _Traits, class _Alloc >`
`void M_copy_to_string (std::basic_string< _CharT, _Traits, _Alloc > &__s)`
`const`
- `bool all () const`
- `bool any () const`
- `size_t count () const`
- `bitset< _Nb > & flip ()`
- `bitset< _Nb > & flip (size_t __position)`
- `bool none () const`
- `bitset< _Nb > operator~ () const`
- `bitset< _Nb > & reset ()`
- `bitset< _Nb > & reset (size_t __position)`
- `bitset< _Nb > & set ()`
- `bitset< _Nb > & set (size_t __position, bool __val=true)`
- `size_t size () const`
- `bool test (size_t __position) const`
- `template<class _CharT, class _Traits >`
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string ()`
`const`
- `template<class _CharT >`
`std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`
`> > to_string () const`
- `template<class _CharT >`
`std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT`
`> > to_string (_CharT __zero, _CharT __one=_CharT('1')) const`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_string< _CharT, _Traits, _Alloc > to_string () const`
- `template<class _CharT, class _Traits >`
`std::basic_string< _CharT, _Traits, std::allocator< _CharT > > to_string (_-`
`CharT __zero, _CharT __one=_CharT('1')) const`

- `template<class _CharT, class _Traits, class _Alloc>`
`std::basic_string<_CharT, _Traits, _Alloc>` **to_string** (`_CharT __zero, _CharT __one=_CharT('1')`) `const`
- `std::basic_string<char, std::char_traits<char>, std::allocator<char>>` **to_string** (`char __zero, char __one='1'`) `const`
- `std::basic_string<char, std::char_traits<char>, std::allocator<char>>` **to_string** () `const`
- unsigned long long **to_ullong** () `const`
- unsigned long **to_ulong** () `const`

- `bitset<_Nb>` & **operator&=** (`const bitset<_Nb> &__rhs`)
- `bitset<_Nb>` & **operator|=** (`const bitset<_Nb> &__rhs`)
- `bitset<_Nb>` & **operator^=** (`const bitset<_Nb> &__rhs`)

- `bitset<_Nb>` & **operator<<=** (`size_t __position`)
- `bitset<_Nb>` & **operator>>=** (`size_t __position`)

- `bitset<_Nb>` & **_Unchecked_set** (`size_t __pos`)
- `bitset<_Nb>` & **_Unchecked_set** (`size_t __pos, int __val`)
- `bitset<_Nb>` & **_Unchecked_reset** (`size_t __pos`)
- `bitset<_Nb>` & **_Unchecked_flip** (`size_t __pos`)
- `bool _Unchecked_test` (`size_t __pos`) `const`

- **reference operator[]** (`size_t __position`)
- `bool operator[]` (`size_t __position`) `const`

- `bool operator==` (`const bitset<_Nb> &__rhs`) `const`
- `bool operator!=` (`const bitset<_Nb> &__rhs`) `const`

- `bitset<_Nb>` **operator<<** (`size_t __position`) `const`
- `bitset<_Nb>` **operator>>** (`size_t __position`) `const`

Private Member Functions

- `size_t _M_are_all_aux` () `const`
- `void _M_do_and` (`const _Base_bitset<_Nw> &__x`)
- `size_t _M_do_count` () `const`
- `size_t _M_do_find_first` (`size_t __not_found`) `const`
- `size_t _M_do_find_next` (`size_t __prev, size_t __not_found`) `const`
- `void _M_do_flip` ()
- `void _M_do_left_shift` (`size_t __shift`)
- `void _M_do_or` (`const _Base_bitset<_Nw> &__x`)
- `void _M_do_reset` ()

- void **_M_do_right_shift** (size_t __shift)
- void **_M_do_set** ()
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const [_Base_bitset](#)< _Nw > &__x)
- const _WordT * **_M_getdata** () const
- _WordT & **_M_getword** (size_t __pos)
- _WordT **_M_getword** (size_t __pos) const
- _WordT & **_M_hiword** ()
- _WordT **_M_hiword** () const
- bool **_M_is_any** () const
- bool **_M_is_equal** (const [_Base_bitset](#)< _Nw > &__x) const

Static Private Member Functions

- static _WordT **_S_maskbit** (size_t __pos)
- static size_t **_S_whichbit** (size_t __pos)
- static size_t **_S_whichbyte** (size_t __pos)
- static size_t **_S_whichword** (size_t __pos)

Private Attributes

- _WordT [_M_w](#) [_Nw]

Friends

- class **hash**
- class **reference**

5.391.1 Detailed Description

template<size_t _Nb> class std::bitset< _Nb >

The bitset class represents a *fixed-size* sequence of bits. (Note that bitset does *not* meet the formal requirements of a [container](#). Mainly, it lacks iterators.)

The template argument, *Nb*, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let B be the number of bits in a word, then (Nb+(B-1))/B words will be used for storage. B - NbB bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `bitset` as *a simple array of bits*, be aware that your mental picture is reversed: a `bitset` behaves the same way as bits in integers do, with the bit at index 0 in the *least significant / right-hand* position, and the bit at index `Nb-1` in the *most significant / left-hand* position. Thus, unlike other containers, a `bitset`'s index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long          a = 'a';
    bitset<10>    b(a);

    cout << "b('a') is " << b << endl;

    ostringstream s;
    s << b;
    string str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch33s02.html> for a description of extensions.

Most of the actual code isn't contained in `bitset<>` itself, but in the base class `_Base_bitset`. The base class works with whole words, not with individual bits. This allows us to specialize `_Base_bitset` for the important special case where the `bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `_Base_bitset` is a regular array, and is indexed as such. This is carefully encapsulated.

Definition at line 708 of file `bitset`.

5.391.2 Constructor & Destructor Documentation

5.391.2.1 template<size_t _Nb> std::bitset< _Nb >::bitset () [inline]

All bits set to zero.

Definition at line 802 of file `bitset`.

5.391.2.2 `template<size_t _Nb> std::bitset< _Nb >::bitset (unsigned long long __val) [inline]`

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 807 of file `bitset`.

5.391.2.3 `template<size_t _Nb> template<class _CharT , class _Traits , class _Alloc > std::bitset< _Nb >::bitset (const std::basic_string< _CharT, _Traits, _Alloc > & __s, size_t __position = 0) [inline, explicit]`

Use a subset of a string.

Parameters

s A string of *0* and *1* characters.

position Index of the first character in *s* to use; defaults to zero.

Exceptions

std::out_of_range If *pos* is bigger the size of *s*.

std::invalid_argument If a character appears in the string which is neither *0* nor *1*.

Definition at line 825 of file `bitset`.

5.391.2.4 `template<size_t _Nb> template<class _CharT , class _Traits , class _Alloc > std::bitset< _Nb >::bitset (const std::basic_string< _CharT, _Traits, _Alloc > & __s, size_t __position, size_t __n) [inline]`

Use a subset of a string.

Parameters

s A string of *0* and *1* characters.

position Index of the first character in *s* to use.

n The number of characters to copy.

Exceptions

std::out_of_range If *pos* is bigger the size of *s*.

std::invalid_argument If a character appears in the string which is neither *0* nor *1*.

Definition at line 847 of file `bitset`.

5.391.2.5 `template<size_t _Nb> std::bitset< _Nb >::bitset (const char * __str
) [inline, explicit]`

Construct from a string.

Parameters

str A string of 0 and 1 characters.

Exceptions

std::invalid_argument If a character appears in the string which is neither 0 nor 1.

Definition at line 879 of file `bitset`.

5.391.3 Member Function Documentation

5.391.3.1 `template<size_t _Nb> bool std::bitset< _Nb >::all () const
[inline]`

Tests whether all the bits are on.

Returns

True if all the bits are set.

Definition at line 1266 of file `bitset`.

5.391.3.2 `template<size_t _Nb> bool std::bitset< _Nb >::any () const
[inline]`

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

Definition at line 1274 of file `bitset`.

5.391.3.3 `template<size_t _Nb> size_t std::bitset< _Nb >::count () const
[inline]`

Returns the number of bits which are set.

Definition at line 1226 of file `bitset`.

5.391.3.4 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::flip (`
`size_t __position) [inline]`

Toggles a given bit to its opposite value.

Parameters

position The index of the bit.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of the set.

Definition at line 1066 of file `bitset`.

5.391.3.5 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::flip ()`
`[inline]`

Toggles every bit to its opposite value.

Definition at line 1053 of file `bitset`.

5.391.3.6 `template<size_t _Nb> bool std::bitset<_Nb>::none () const`
`[inline]`

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

Definition at line 1282 of file `bitset`.

5.391.3.7 `template<size_t _Nb> bool std::bitset<_Nb>::operator!= (const`
`bitset<_Nb> & __rhs) const [inline]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1241 of file `bitset`.

5.391.3.8 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>`
`>::operator&= (const bitset<_Nb> & __rhs) [inline]`

Operations on bitsets.

Parameters

rhs A same-sized bitset.

These should be self-explanatory.

Definition at line 900 of file bitset.

5.391.3.9 `template<size_t _Nb> bitset<_Nb> std::bitset< _Nb
>::operator<<(size_t __position) const [inline]`

Self-explanatory.

Definition at line 1288 of file bitset.

5.391.3.10 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::operator<<=(size_t __position) [inline]`

Operations on bitsets.

Parameters

position The number of places to shift.

These should be self-explanatory.

Definition at line 929 of file bitset.

5.391.3.11 `template<size_t _Nb> bool std::bitset< _Nb >::operator==(const
bitset< _Nb > & __rhs) const [inline]`

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1237 of file bitset.

5.391.3.12 `template<size_t _Nb> bitset<_Nb> std::bitset< _Nb
>::operator>>(size_t __position) const [inline]`

Self-explanatory.

Definition at line 1292 of file bitset.

5.391.3.13 `template<size_t _Nb> bitset<_Nb>& std::bitset< _Nb
>::operator>>=(size_t __position) [inline]`

Operations on bitsets.

Parameters

position The number of places to shift.

These should be self-explanatory.

Definition at line 942 of file `bitset`.

5.391.3.14 `template<size_t _Nb> bool std::bitset<_Nb>::operator[] (size_t
__position) const [inline]`

Array-indexing support.

Parameters

position Index into the `bitset`.

Returns

A `bool` for a *const* `bitset`. For non-const `bitsets`, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1098 of file `bitset`.

5.391.3.15 `template<size_t _Nb> reference std::bitset<_Nb>::operator[] (
size_t __position) [inline]`

Array-indexing support.

Parameters

position Index into the `bitset`.

Returns

A `bool` for a *const* `bitset`. For non-const `bitsets`, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1094 of file `bitset`.

5.391.3.16 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator^=(const bitset<_Nb> & __rhs) [inline]`

Operations on bitsets.

Parameters

rhs A same-sized bitset.

These should be self-explanatory.

Definition at line 914 of file `bitset`.

5.391.3.17 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::operator|=(const bitset<_Nb> & __rhs) [inline]`

Operations on bitsets.

Parameters

rhs A same-sized bitset.

These should be self-explanatory.

Definition at line 907 of file `bitset`.

5.391.3.18 `template<size_t _Nb> bitset<_Nb> std::bitset<_Nb>::operator~() const [inline]`

See the no-argument [flip\(\)](#).

Definition at line 1075 of file `bitset`.

5.391.3.19 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::reset () [inline]`

Sets every bit to false.

Definition at line 1028 of file `bitset`.

5.391.3.20 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::reset (size_t __position) [inline]`

Sets a given bit to false.

Parameters

position The index of the bit.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of the set.

Same as writing `set (pos, false)`.

Definition at line 1042 of file `bitset`.

5.391.3.21 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::set () [inline]`

Sets every bit to true.

Definition at line 1003 of file `bitset`.

5.391.3.22 `template<size_t _Nb> bitset<_Nb>& std::bitset<_Nb>::set (size_t __position, bool __val = true) [inline]`

Sets a given bit to a particular value.

Parameters

position The index of the bit.

val Either true or false, defaults to true.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of the set.

Definition at line 1017 of file `bitset`.

5.391.3.23 `template<size_t _Nb> size_t std::bitset< _Nb >::size () const`
`[inline]`

Returns the total number of bits.

Definition at line 1231 of file `bitset`.

5.391.3.24 `template<size_t _Nb> bool std::bitset< _Nb >::test (size_t`
`__position) const [inline]`

Tests the value of a bit.

Parameters

position The index of a bit.

Returns

The value at *pos*.

Exceptions

[*std::out_of_range*](#) If *pos* is bigger the size of the set.

Definition at line 1252 of file `bitset`.

5.391.3.25 `template<size_t _Nb> template<class _CharT , class _Traits , class`
`_Alloc > std::basic_string< _CharT, _Traits, _Alloc> std::bitset<`
`_Nb >::to_string () const [inline]`

Returns a character interpretation of the `bitset`.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1128 of file `bitset`.

5.391.3.26 `template<size_t _Nb> unsigned long std::bitset< _Nb >::to_ulong (`
`) const [inline]`

Returns a numerical interpretation of the `bitset`.

Returns

The integral equivalent of the bits.

Exceptions

[*std::overflow_error*](#) If there are too many bits to be represented in an unsigned long.

Definition at line 1109 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)

5.392 std::bitset< _Nb >::reference Class Reference**Public Member Functions**

- **reference** ([bitset](#) &__b, size_t __pos)
- [reference](#) & **flip** ()
- **operator bool** () const
- [reference](#) & **operator=** (const [reference](#) &__j)
- [reference](#) & **operator=** (bool __x)
- bool **operator~** () const

Friends

- class **bitset**

5.392.1 Detailed Description

template<size_t _Nb> class std::bitset< _Nb >::reference

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)

Definition at line 739 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)

5.393 `std::cauchy_distribution<_RealType>` Class Template Reference

A [cauchy_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- **`cauchy_distribution`** (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- **`cauchy_distribution`** (`const param_type &__p`)
- `_RealType a` () const
- `_RealType b` () const
- [result_type](#) `max` () const
- [result_type](#) `min` () const
- `template<typename _UniformRandomNumberGenerator>`
[result_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator>`
[result_type](#) **`operator()`** (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (`const param_type &__param`)
- [param_type](#) `param` () const
- void [reset](#) ()

5.393.1 Detailed Description

```
template<typename _RealType = double> class std::cauchy_distribution< _RealType >
```

A [cauchy_distribution](#) random number distribution. The formula for the normal probability mass function is $p(x|a, b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2707 of file random.h.

5.393.2 Member Typedef Documentation

5.393.2.1 `template<typename _RealType = double> typedef _RealType std::cauchy_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2714 of file random.h.

5.393.3 Member Function Documentation

5.393.3.1 `template<typename _RealType = double> result_type std::cauchy_distribution< _RealType >::max() const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2798 of file random.h.

5.393.3.2 `template<typename _RealType = double> result_type std::cauchy_distribution< _RealType >::min() const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2791 of file random.h.

5.393.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::cauchy_distribution< _RealType >::operator()(_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2806 of file random.h.

References `std::cauchy_distribution< _RealType >::operator()()`, and `std::cauchy_distribution< _RealType >::param()`.

5.394 `std::cauchy_distribution<_RealType>::param_type` Struct Reference 2227

Referenced by `std::cauchy_distribution<_RealType>::operator()`.

5.393.3.4 `template<typename _RealType = double> void
std::cauchy_distribution<_RealType>::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 2784 of file `random.h`.

5.393.3.5 `template<typename _RealType = double> param_type
std::cauchy_distribution<_RealType>::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2776 of file `random.h`.

Referenced by `std::cauchy_distribution<_RealType>::operator()`,
`std::operator==()`, and `std::operator>>()`.

5.393.3.6 `template<typename _RealType = double> void
std::cauchy_distribution<_RealType>::reset () [inline]`

Resets the distribution state.

Definition at line 2758 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.394 `std::cauchy_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `cauchy_distribution<_RealType>` `distribution_type`

Public Member Functions

- **param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.394.1 Detailed Description

template<typename _RealType = double> struct std::cauchy_distribution< _RealType >::param_type

Parameter type.

Definition at line 2716 of file random.h.

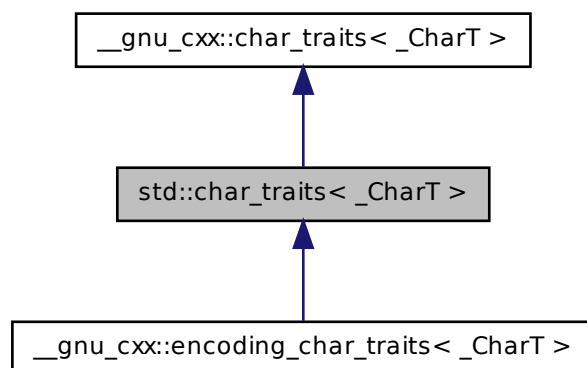
The documentation for this struct was generated from the following file:

- [random.h](#)

5.395 std::char_traits< _CharT > Struct Template Reference

Basis for explicit traits specializations.

Inheritance diagram for std::char_traits< _CharT >:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types< _CharT >::int_type` **int_type**
- typedef `_Char_types< _CharT >::off_type` **off_type**
- typedef `_Char_types< _CharT >::pos_type` **pos_type**
- typedef `_Char_types< _CharT >::state_type` **state_type**

Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static int_type **eof** ()
- static bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static bool **eq_int_type** (`const int_type &__c1`, `const int_type &__c2`)
- static `const char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)

- static std::size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.395.1 Detailed Description

template<class _CharT> struct std::char_traits< _CharT >

Basis for explicit traits specializations.

Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around [__gnu_cxx::char_traits](#), it is possible to achieve a more appropriate definition by specializing [__gnu_cxx::char_traits](#).

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out [include/ext/pod_char_traits.h](#).

Definition at line 224 of file char_traits.h.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.396 std::char_traits< __gnu_cxx::character< V, I, S > > Struct Template Reference

char_traits<__gnu_cxx::character> specialization.

Public Types

- typedef [__gnu_cxx::character](#)< V, I, S > **char_type**
- typedef char_type::int_type **int_type**
- typedef [streamoff](#) **off_type**
- typedef [fpos](#)< state_type > **pos_type**
- typedef char_type::state_type **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **eof** ()
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__i)
- static int_type **to_int_type** (const char_type &__c)

5.396.1 Detailed Description

template<typename V, typename I, typename S> struct std::char_traits< __gnu_cxx::character< V,I,S > >

char_traits<__gnu_cxx::character> specialization.

Definition at line 88 of file pod_char_traits.h.

The documentation for this struct was generated from the following file:

- [pod_char_traits.h](#)

5.397 std::char_traits< char > Struct Template Reference

21.1.3.1 [char_traits](#) specializations

Public Types

- typedef char **char_type**
- typedef int **int_type**
- typedef [streamoff](#) **off_type**
- typedef [streampos](#) **pos_type**
- typedef mbstate_t **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **eof** ()
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.397.1 Detailed Description

template<> struct std::char_traits< char >

21.1.3.1 [char_traits](#) specializations

Definition at line 230 of file char_traits.h.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.398 std::char_traits< wchar_t > Struct Template Reference

21.1.3.2 [char_traits](#) specializations

Public Types

- typedef wchar_t **char_type**
- typedef wint_t **int_type**
- typedef [streamoff](#) **off_type**
- typedef [wstreampos](#) **pos_type**
- typedef mbstate_t **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **eof** ()
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type *__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.398.1 Detailed Description

`template<> struct std::char_traits< wchar_t >`

21.1.3.2 [char_traits](#) specializations

Definition at line 301 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

5.399 `std::chi_squared_distribution< _RealType >` Class Template Reference

A [chi_squared_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- **chi_squared_distribution** (_RealType __n=_RealType(1))
- **chi_squared_distribution** (const [param_type](#) &__p)
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- void [reset](#) ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_ostream](#)< _CharT, _Traits > & **operator<<** ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::chi_squared_distribution](#)< _RealType1 > &)
- template<typename _RealType1 >
bool **operator==** (const [std::chi_squared_distribution](#)< _RealType1 > &__d1, const [std::chi_squared_distribution](#)< _RealType1 > &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator>>** ([std::basic_istream](#)< _CharT, _Traits > &, [std::chi_squared_distribution](#)< _RealType1 > &)

5.399.1 Detailed Description

template<typename _RealType = double> class [std::chi_squared_distribution](#)< _RealType >

A [chi_squared_distribution](#) random number distribution. The formula for the normal probability mass function is $p(x|n) = \frac{x^{(n/2)-1} e^{-x/2}}{\Gamma(n/2) 2^{n/2}}$

Definition at line 2542 of file random.h.

5.399.2 Member Typedef Documentation

5.399.2.1 **template<typename _RealType = double> typedef [_RealType](#) [std::chi_squared_distribution](#)< _RealType >::result_type**

The type of the range of the distribution.

5.399 std::chi_squared_distribution< _RealType > Class Template Reference

Definition at line 2549 of file random.h.

5.399.3 Member Function Documentation

5.399.3.1 `template<typename _RealType = double> result_type
std::chi_squared_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2622 of file random.h.

5.399.3.2 `template<typename _RealType = double> result_type
std::chi_squared_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2615 of file random.h.

5.399.3.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::chi_squared_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2630 of file random.h.

5.399.3.4 `template<typename _RealType = double> param_type
std::chi_squared_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2600 of file random.h.

5.399.3.5 `template<typename _RealType = double> void
std::chi_squared_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 2608 of file random.h.

5.399.3.6 `template<typename _RealType = double> void
std::chi_squared_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2586 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

5.399.4 Friends And Related Function Documentation

5.399.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::chi_squared_distribution< _RealType1 > &) [friend]`

Inserts a `chi_squared_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

__os An output stream.

__x A `chi_squared_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.399.4.2 `template<typename _RealType = double> template<typename _
RealType1 > bool operator==(const std::chi_squared_distribution<
_RealType1 > & __d1, const std::chi_squared_distribution<
_RealType1 > & __d2) [friend]`

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2650 of file random.h.

5.399.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> &, std::chi_squared_distribution<_RealType1> &) [friend]`

Extracts a `chi_squared_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `chi_squared_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

5.400 `std::chi_squared_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [chi_squared_distribution<_RealType>](#) `distribution_type`

Public Member Functions

- `param_type` (`_RealType __n=_RealType(1)`)
- `_RealType n` () const

Friends

- bool `operator==` (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.400.1 Detailed Description

```
template<typename _RealType = double> struct std::chi_squared_distribution<
_RealType >::param_type
```

Parameter type.

Definition at line 2551 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.401 std::chrono::duration< _Rep, _Period > Struct Template Reference

duration

Public Types

- typedef _Period **period**
- typedef _Rep **rep**

Public Member Functions

- template<typename _Rep2 , typename = typename enable_if<is_convertible<_Rep2, rep>::value && (treat_as_floating_point<rep>::value || !treat_as_floating_point<_Rep2>::value)>::type>
duration (const _Rep2 &__rep)
- **duration** (const [duration](#) &)
- template<typename _Rep2 , typename _Period2 , typename = typename enable_if<treat_as_floating_point<rep>::value || (ratio_divide<_Period2, period>::type::den == 1 && !treat_as_floating_point<_Rep2>::value)>::type>
duration (const [duration](#)< _Rep2, _Period2 > &__d)
- rep **count** () const
- template<typename _Rep2 = rep>
[enable_if](#)<!treat_as_floating_point< _Rep2 >::value, [duration](#) & >::type **operator** %= (const rep &__rhs)
- template<typename _Rep2 = rep>
[enable_if](#)<!treat_as_floating_point< _Rep2 >::value, [duration](#) & >::type **operator** %= (const [duration](#) &__d)
- [duration](#) & **operator** *= (const rep &__rhs)
- [duration](#) **operator** + () const

- `duration` & `operator++` ()
- `duration` `operator++` (int)
- `duration` & `operator+=` (const `duration` & __d)
- `duration` `operator-` () const
- `duration` & `operator--` ()
- `duration` `operator--` (int)
- `duration` & `operator-=` (const `duration` & __d)
- `duration` & `operator/=` (const rep & __rhs)
- `duration` & `operator=` (const `duration` &)
- `static_assert` (!__is_duration< _Rep >::value, "rep cannot be a `duration`")
- `static_assert` (_Period::num > 0, "period must be positive")
- `static_assert` (__is_ratio< _Period >::value, "period must be a specialization of `ratio`")

Static Public Member Functions

- static const `duration` `max` ()
- static const `duration` `min` ()
- static const `duration` `zero` ()

5.401.1 Detailed Description

`template<typename _Rep, typename _Period> struct std::chrono::duration< _Rep, _Period >`

`duration`

Definition at line 201 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

5.402 `std::chrono::duration_values< _Rep >` Struct Template Reference

`duration_values`

Static Public Member Functions

- static const `_Rep` **max** ()
- static const `_Rep` **min** ()
- static const `_Rep` **zero** ()

5.402.1 Detailed Description

`template<typename _Rep> struct std::chrono::duration_values< _Rep >`

[duration_values](#)

Definition at line 174 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

5.403 `std::chrono::system_clock` Struct Reference

[system_clock](#)

Public Types

- typedef [chrono::seconds](#) **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**
- typedef [chrono::time_point](#)< [system_clock](#), [duration](#) > **time_point**

Static Public Member Functions

- static [time_point](#) **from_time_t** (std::time_t __t)
- static [time_point](#) **now** () throw ()
- static std::time_t **to_time_t** (const [time_point](#) &__t)

Static Public Attributes

- static const bool **is_monotonic**

5.404 `std::chrono::time_point< _Clock, _Duration >` Struct Template Reference 2241

5.403.1 Detailed Description

[system_clock](#)

Definition at line 628 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

5.404 `std::chrono::time_point< _Clock, _Duration >` Struct Template Reference

[time_point](#)

Public Types

- typedef `_Clock` **clock**
- typedef `_Duration` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**

Public Member Functions

- **time_point** (const `duration` &__dur)
- template<typename `_Duration2` >
 time_point (const [time_point](#)< `clock`, `_Duration2` > &__t)
- [time_point](#) & **operator+=** (const `duration` &__dur)
- [time_point](#) & **operator-=** (const `duration` &__dur)
- `duration` **time_since_epoch** () const

Static Public Member Functions

- static const [time_point](#) **max** ()
- static const [time_point](#) **min** ()

5.404.1 Detailed Description

```
template<typename _Clock, typename _Duration> struct std::chrono::time_
point< _Clock, _Duration >
```

[time_point](#)

Definition at line 492 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

5.405 `std::chrono::treat_as_floating_point< _Rep >` Struct Template Reference

[treat_as_floating_point](#)

Inherits `is_floating_point< _Rep >`.

5.405.1 Detailed Description

```
template<typename _Rep> struct std::chrono::treat_as_floating_point< _Rep >
```

[treat_as_floating_point](#)

Definition at line 168 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

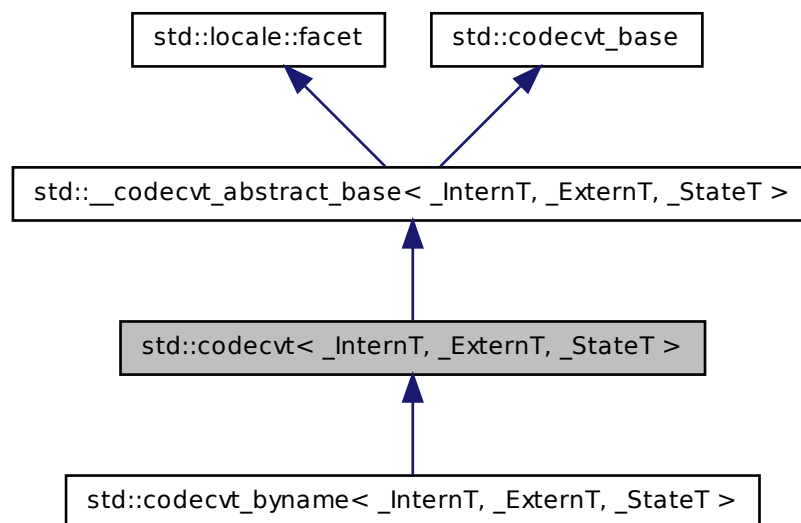
5.406 `std::codecvt< _InternT, _ExternT, _StateT >` Class Template Reference

Primary class template `codecvt`.

NB: Generic, mostly useless implementation.

5.406 `std::codecvt<_InternT, _ExternT, _StateT >` Class Template Reference

Inheritance diagram for `std::codecvt<_InternT, _ExternT, _StateT >`:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`__c_locale __cloc, size_t __refs=0`)
- **bool always_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type *__from, const extern_type * __from_end, const extern_type *&__from_next, intern_type *__to, intern_type * __to_end, intern_type *&__to_next`) const

- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static **locale::id** id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class locale::_Impl

5.406.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT> class std::codecvt< _InternT, _ExternT, _StateT >

Primary class template codecvt.

NB: Generic, mostly useless implementation.

Definition at line 275 of file codecvt.h.

5.406.2 Member Function Documentation

5.406.2.1 **template<typename _InternT , typename _ExternT , typename _StateT > virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const** **[protected, virtual]**

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

[out](#) for more information.

Implements [std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >](#).

5.406.2.2 **template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next) const** **[inline, inherited]**

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

Referenced by `std::basic_filebuf<_CharT, _Traits >::underflow()`.

5.406.2.3 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

5.406 std::codecvt<_InternT, _ExternT, _StateT > Class Template Reference 2247

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 115 of file codecvt.h.

```
5.406.2.4 template<typename _InternT, typename _ExternT, typename
        _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
        _StateT >::unshift ( state_type & __state, extern_type * __to,
        extern_type * __to_end, extern_type *& __to_next ) const
        [inline, inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to_end* End of output buffer.
- to_next* Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.407 `std::codecvt< _InternT, _ExternT, encoding_state >` Class Template Reference

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Inheritance diagram for `std::codecvt< _InternT, _ExternT, encoding_state >`:



Public Types

- typedef `state_type::descriptor_type` **descriptor_type**

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (state_type &__enc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static `locale::id` **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()

- virtual result `do_out` (`state_type` &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result `do_out` (`state_type` &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result `do_unshift` (`state_type` &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result `do_unshift` (`state_type` &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale` &__cloc) throw ()
- static void `_S_create_c_locale` (`__c_locale` &__cloc, const char *__s, `__c_locale` __old=0)
- static void `_S_destroy_c_locale` (`__c_locale` &__cloc)
- static `__c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char * `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale` __cloc, const char *__s)

Friends

- class `locale::_Impl`

5.407.1 Detailed Description

`template<typename _InternT, typename _ExternT> class std::codecvt< _InternT, _ExternT, encoding_state >`

`codecvt<InternT, _ExternT, encoding_state>` specialization.

Definition at line 227 of file `codecvt_specializations.h`.

5.407.2 Member Function Documentation

5.407.2.1 virtual result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const [protected, pure virtual, inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

5.407.2.2 result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next) const [inline, inherited]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

state Persistent conversion state data.

from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 195 of file codecvt.h.

5.407.2.3 `result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 115 of file codecvt.h.

5.407.2.4 result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const
[inline, inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

state Persistent conversion state data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 154 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt_specializations.h](#)

5.408 std::codecvt< char, char, mbstate_t > Class Template Reference

class [codecvt<char, char, mbstate_t>](#) specialization.

Inheritance diagram for std::codecvt< char, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type * __from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type * __from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class `locale::_Impl`

5.408.1 Detailed Description

`template<> class std::codecvt< char, char, mbstate_t >`

class `codecvt<char, char, mbstate_t>` specialization.

Definition at line 337 of file `codecvt.h`.

5.408.2 Member Function Documentation

5.408.2.1 virtual result `std::__codecvt_abstract_base< char , char , mbstate_t >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const` [`protected`, `pure virtual`, `inherited`]

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

out for more information.

5.408.2.2 result `std::__codecvt_abstract_base< char , char , mbstate_t >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next) const` [`inline`, `inherited`]

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted

character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

5.408.2.3 `result std::__codecvt_abstract_base< char , char , mbstate_t >::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

5.408.2.4 `result std::__codecvt_abstract_base< char , char , mbstate_t
 >::unshift (state_type & __state, extern_type * __to, extern_type
 * __to_end, extern_type *& __to_next) const [inline,
 inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

5.409 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference 2259

Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to_end* End of output buffer.
- to_next* Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.409 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference

class `codecvt<wchar_t, char, mbstate_t>` specialization.

Inheritance diagram for `std::codecvt< wchar_t, char, mbstate_t >`:



Public Types

- typedef char **extern_type**
- typedef wchar_t **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `mbstate_t` **state_type**

Public Member Functions

- `codecvt` (`size_t __refs=0`)
- `codecvt` (`__c_locale __cloc, size_t __refs=0`)
- bool **always_noconv** () const throw ()

- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class locale::_Impl

5.409.1 Detailed Description

template<> class std::codecvt< wchar_t, char, mbstate_t >

class [codecvt<wchar_t, char, mbstate_t>](#) specialization.

Definition at line 395 of file codecvt.h.

5.409.2 Member Function Documentation

5.409.2.1 virtual result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [protected, pure virtual, inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

```

5.409.2.2 result std::__codecvt_abstract_base< wchar_t , char , mbstate_t
>::in ( state_type & __state, const extern_type * __from, const
extern_type * __from_end, const extern_type *& __from_next,
intern_type * __to, intern_type * __to_end, intern_type *&
__to_next ) const [inline, inherited]

```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 195 of file `codecvt.h`.

5.409.2.3 result std::__codecvt_abstract_base< wchar_t , char , mbstate_t
 >::out (state_type & __state, const intern_type * __from, const
 intern_type * __from_end, const intern_type *& __from_next,
 extern_type * __to, extern_type * __to_end, extern_type *&
 __to_next) const [inline, inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

state Persistent conversion state data.

from Start of input.

from_end End of input.

from_next Returns start of unconverted data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 115 of file codecvt.h.

```
5.409.2.4 result std::__codecvt_abstract_base< wchar_t , char , mbstate_t
>::unshift ( state_type & __state, extern_type * __to, extern_type
* __to_end, extern_type *& __to_next ) const [inline,
inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

- state* Persistent conversion state data.
- to* Start of output buffer.
- to_end* End of output buffer.
- to_next* Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

The documentation for this class was generated from the following file:

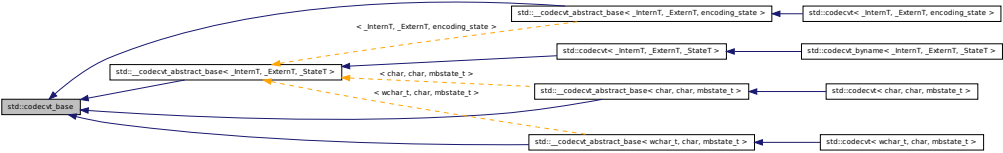
- [codecvt.h](#)

5.410 std::codecvt_base Class Reference

Empty base class for `codecvt` facet [22.2.1.5].

5.411 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference2265

Inheritance diagram for std::codecvt_base:



Public Types

- enum result { ok, partial, error, noconv }

5.410.1 Detailed Description

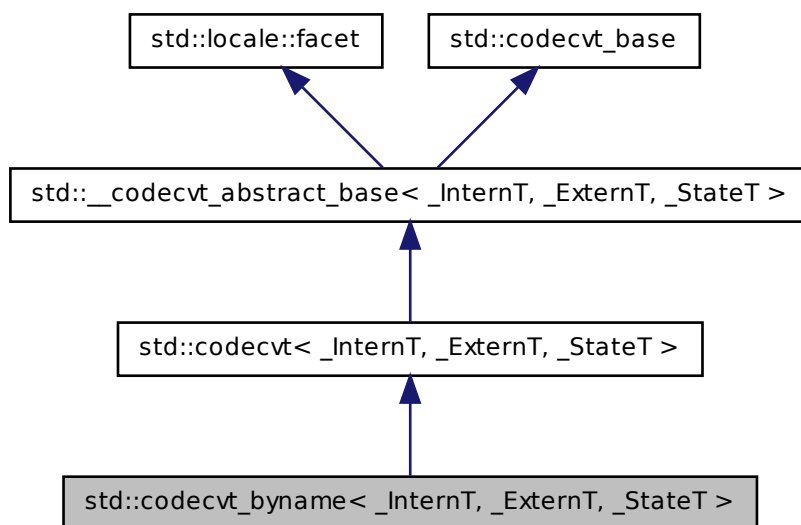
Empty base class for codecvt facet [22.2.1.5].
Definition at line 45 of file codecvt.h.
The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.411 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference

class [codecvt_byname](#) [22.2.1.6].

Inheritance diagram for `std::codecvt_byname< _InternT, _ExternT, _StateT >`:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- **codecvt_byname** (`const char *__s, size_t __refs=0`)
- **bool always_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next`) const
- **int length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const

- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static **locale::id** id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class `locale::_Impl`

5.411.1 Detailed Description

`template<typename _InternT, typename _ExternT, typename _StateT> class std::codecvt_byname< _InternT, _ExternT, _StateT >`

class `codecvt_byname` [22.2.1.6].

Definition at line 455 of file `codecvt.h`.

5.411.2 Member Function Documentation

5.411.2.1 `template<typename _InternT , typename _ExternT , typename _StateT > virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [protected, virtual, inherited]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

[out](#) for more information.

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`.

5.411.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next) const [inline, inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

5.411 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference 2269

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 195 of file codecvt.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::underflow().

```
5.411.2.3 template<typename _InternT, typename _ExternT, typename  
_StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,  
_StateT >::out ( state_type & __state, const intern_type * __from,  
const intern_type * __from_end, const intern_type *& __from_next,  
extern_type * __to, extern_type * __to_end, extern_type *&  
__to_next ) const [inline, inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling [codecvt::do_out](#).

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.
from Start of input.
from_end End of input.
from_next Returns start of unconverted data.
to Start of output buffer.
to_end End of output buffer.
to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 115 of file `codecvt.h`.

```
5.411.2.4 template<typename _InternT, typename _ExternT, typename
    _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT,
    _StateT >::unshift ( state_type & __state, extern_type * __to,
    extern_type * __to_end, extern_type *& __to_next ) const
    [inline, inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

state Persistent conversion state data.

to Start of output buffer.

to_end End of output buffer.

to_next Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 154 of file `codecvt.h`.

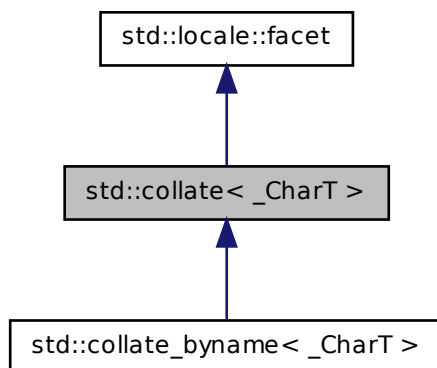
The documentation for this class was generated from the following file:

- [codecvt.h](#)

5.412 `std::collate<_CharT>` Class Template Reference

Facet for localized string comparison.

Inheritance diagram for `std::collate<_CharT>`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef [basic_string<_CharT>](#) [string_type](#)

Public Member Functions

- [collate](#) (size_t __refs=0)
- [collate](#) (__c_locale __cloc, size_t __refs=0)
- template<>
int **_M_compare** (const char *, const char *) const throw()
- template<>
int **_M_compare** (const wchar_t *, const wchar_t *) const throw()
- int **_M_compare** (const _CharT *, const _CharT *) const throw ()
- template<>
size_t **_M_transform** (char *, const char *, size_t) const throw()
- template<>
size_t **_M_transform** (wchar_t *, const wchar_t *, size_t) const throw()
- size_t **_M_transform** (_CharT *, const _CharT *, size_t) const throw ()
- int [compare](#) (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const

- long [hash](#) (const _CharT *__lo, const _CharT *__hi) const
- [string_type transform](#) (const _CharT *__lo, const _CharT *__hi) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~collate](#) ()
- virtual int [do_compare](#) (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- virtual long [do_hash](#) (const _CharT *__lo, const _CharT *__hi) const
- virtual [string_type do_transform](#) (const _CharT *__lo, const _CharT *__hi) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale [_M_c_locale_collate](#)

Friends

- class [locale::_Impl](#)

5.412.1 Detailed Description

template<typename _CharT> class std::collate<_CharT>

Facet for localized string comparison. This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 616 of file locale_classes.h.

5.412.2 Member Typedef Documentation

5.412.2.1 `template<typename _CharT> typedef _CharT std::collate<_CharT>::char_type`

Public typedefs.

Reimplemented in [std::collate_byname<_CharT>](#).

Definition at line 622 of file locale_classes.h.

5.412.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate<_CharT>::string_type`

Public typedefs.

Reimplemented in [std::collate_byname<_CharT>](#).

Definition at line 623 of file locale_classes.h.

5.412.3 Constructor & Destructor Documentation

5.412.3.1 `template<typename _CharT> std::collate<_CharT>::collate (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 643 of file locale_classes.h.

5.412.3.2 `template<typename _CharT> std::collate<_CharT>::collate (__c_locale __cloc, size_t __refs = 0) [inline, explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

cloc The C locale.
refs Passed to the base facet class.

Definition at line 657 of file locale_classes.h.

5.412.3.3 `template<typename _CharT> virtual std::collate<_CharT>::~collate() [inline, protected, virtual]`

Destructor.

Definition at line 720 of file locale_classes.h.

5.412.4 Member Function Documentation

5.412.4.1 `template<typename _CharT> int std::collate<_CharT>::compare (const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [inline]`

Compare two strings.

This function compares two strings and returns the result by calling [collate::do_compare\(\)](#).

Parameters

lo1 Start of string 1.
hi1 End of string 1.
lo2 Start of string 2.
hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 674 of file locale_classes.h.

5.412.4.2 `template<typename _CharT> int std::collate<_CharT>::do_compare (const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [protected, virtual]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

See also

[compare\(\)](#).

Parameters

lo1 Start of string 1.

hi1 End of string 1.

lo2 Start of string 2.

hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 134 of file locale_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

5.412.4.3 `template<typename _CharT> long std::collate<_CharT>
>::do_hash (const _CharT * __lo, const _CharT * __hi) const
[protected, virtual]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

lo Start of string.

hi End of string.

Returns

Hash value.

Definition at line 229 of file locale_classes.tcc.

5.412.4.4 `template<typename _CharT> collate<_CharT>::string_type
std::collate<_CharT>::do_transform (const _CharT * __lo, const
_CharT * __hi) const [protected, virtual]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

Parameters

- lo1* Start of string 1.
- hi1* End of string 1.
- lo2* Start of string 2.
- hi2* End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 173 of file locale_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::length()`, and `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`.

5.412.4.5 `template<typename _CharT> long std::collate< _CharT >::hash (const _CharT * __lo, const _CharT * __hi) const [inline]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do_hash\(\)](#).

Parameters

- lo* Start of string.
- hi* End of string.

Returns

Hash value.

Definition at line 707 of file locale_classes.h.

5.412.4.6 `template<typename _CharT> string_type std::collate< _CharT >::transform (const _CharT * __lo, const _CharT * __hi) const [inline]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning [collate::do_transform\(\)](#).

Parameters

lo Start of string.

hi End of string.

Returns

Transformed string_type.

Definition at line 693 of file locale_classes.h.

Referenced by std::regex_traits< _Ch_type >::transform().

5.412.5 Member Data Documentation

5.412.5.1 `template<typename _CharT> locale::id std::collate< _CharT >::id` `[static]`

Numpunct facet id.

Definition at line 633 of file locale_classes.h.

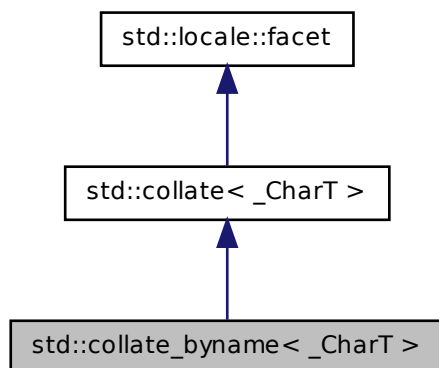
The documentation for this class was generated from the following files:

- [locale_classes.h](#)
- [locale_classes.tcc](#)

5.413 `std::collate_byname< _CharT >` Class Template Reference

class [collate_byname](#) [22.2.4.2].

Inheritance diagram for std::collate_byname< _CharT >:



Public Types

- typedef `_CharT` [char_type](#)
- typedef [basic_string](#)< `_CharT` > [string_type](#)

Public Member Functions

- **collate_byname** (const char *__s, size_t __refs=0)
- **int _M_compare** (const _CharT *, const _CharT *) const throw ()
- template<>
 int _M_compare (const char *, const char *) const throw()
- template<>
 int _M_compare (const wchar_t *, const wchar_t *) const throw()
- **size_t _M_transform** (_CharT *, const _CharT *, size_t) const throw ()
- template<>
 size_t _M_transform (char *, const char *, size_t) const throw()
- template<>
 size_t _M_transform (wchar_t *, const wchar_t *, size_t) const throw()
- **int compare** (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- **long hash** (const _CharT *__lo, const _CharT *__hi) const
- **string_type transform** (const _CharT *__lo, const _CharT *__hi) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual int [do_compare](#) (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- virtual long [do_hash](#) (const _CharT *__lo, const _CharT *__hi) const
- virtual [string_type do_transform](#) (const _CharT *__lo, const _CharT *__hi) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale [_M_c_locale_collate](#)

Friends

- class [locale::Impl](#)

5.413.1 Detailed Description

`template<typename _CharT> class std::collate_byname< _CharT >`

class [collate_byname](#) [22.2.4.2].

Definition at line 792 of file `locale_classes.h`.

5.413.2 Member Typedef Documentation

5.413.2.1 `template<typename _CharT > typedef _CharT std::collate_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::collate< _CharT >](#).

Definition at line 797 of file locale_classes.h.

5.413.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::collate_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::collate< _CharT >](#).

Definition at line 798 of file locale_classes.h.

5.413.3 Member Function Documentation

5.413.3.1 `template<typename _CharT> int std::collate< _CharT >::compare (const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [inline, inherited]`

Compare two strings.

This function compares two strings and returns the result by calling [collate::do_compare\(\)](#).

Parameters

lo1 Start of string 1.

hi1 End of string 1.

lo2 Start of string 2.

hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 674 of file locale_classes.h.

5.413.3.2 `template<typename _CharT> int std::collate< _CharT
>::do_compare (const _CharT * __lo1, const _CharT *
__hi1, const _CharT * __lo2, const _CharT * __hi2) const
[protected, virtual, inherited]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

See also

[compare\(\)](#).

Parameters

lo1 Start of string 1.

hi1 End of string 1.

lo2 Start of string 2.

hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 134 of file locale_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::length()`.

5.413.3.3 `template<typename _CharT> long std::collate< _CharT
>::do_hash (const _CharT * __lo, const _CharT * __hi) const
[protected, virtual, inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

lo Start of string.

hi End of string.

Returns

Hash value.

Definition at line 229 of file locale_classes.tcc.

5.413.3.4 `template<typename _CharT> collate< _CharT >::string_type
std::collate< _CharT >::do_transform (const _CharT * __lo, const
_CharT * __hi) const [protected, virtual, inherited]`

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

Parameters

lo1 Start of string 1.

hi1 End of string 1.

lo2 Start of string 2.

hi2 End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 173 of file locale_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::append()`, `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::length()`, and `std::basic_string< _CharT, _Traits, _Alloc >::push_back()`.

5.413.3.5 `template<typename _CharT> long std::collate< _CharT >::hash (
const _CharT * __lo, const _CharT * __hi) const [inline,
inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning [collate::do_hash\(\)](#).

Parameters

lo Start of string.

hi End of string.

Returns

Hash value.

Definition at line 707 of file locale_classes.h.

5.413.3.6 `template<typename _CharT> string_type std::collate< _CharT
>::transform (const _CharT * __lo, const _CharT * __hi) const
[inline, inherited]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning [collate::do_transform\(\)](#).

Parameters

lo Start of string.

hi End of string.

Returns

Transformed `string_type`.

Definition at line 693 of file `locale_classes.h`.

Referenced by `std::regex_traits< _Ch_type >::transform()`.

5.413.4 Member Data Documentation

5.413.4.1 `template<typename _CharT> locale::id std::collate< _CharT >::id
[static, inherited]`

Numpunct facet id.

Definition at line 633 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

5.414 `std::complex< _Tp >` Struct Template Reference

Public Types

- `typedef _Tp value_type`

Public Member Functions

- `complex` (const _Tp &__r=_Tp(), const _Tp &__i=_Tp())
- `template<typename _Up >`
`complex` (const `complex`< _Up > &__z)
- `const complex & __rep () const`
- `_Tp imag () const`
- `void imag (_Tp __val)`
- `complex< _Tp > & operator*= (const _Tp &)`
- `template<typename _Up >`
`complex< _Tp > & operator*= (const complex< _Up > &)`
- `complex< _Tp > & operator+= (const _Tp &__t)`
- `template<typename _Up >`
`complex< _Tp > & operator+= (const complex< _Up > &)`
- `complex< _Tp > & operator-= (const _Tp &__t)`
- `template<typename _Up >`
`complex< _Tp > & operator-= (const complex< _Up > &)`
- `template<typename _Up >`
`complex< _Tp > & operator/= (const complex< _Up > &)`
- `complex< _Tp > & operator/= (const _Tp &)`
- `template<typename _Up >`
`complex< _Tp > & operator= (const complex< _Up > &)`
- `complex< _Tp > & operator= (const _Tp &)`
- `void real (_Tp __val)`
- `_Tp real () const`

5.414.1 Detailed Description

`template<typename _Tp> struct std::complex< _Tp >`

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

Parameters

Tp Type of real and imaginary values.

Definition at line 122 of file `complex`.

5.414.2 Member Typedef Documentation

5.414.2.1 `template<typename _Tp> typedef _Tp std::complex< _Tp >::value_type`

Value typedef.

Definition at line 125 of file `complex`.

5.414.3 Constructor & Destructor Documentation

5.414.3.1 `template<typename _Tp> std::complex< _Tp >::complex (const _Tp & __r = _Tp (), const _Tp & __i = _Tp ()) [inline]`

Default constructor. First parameter is x, second parameter is y. /// Unspecified parameters default to 0.

Definition at line 129 of file `complex`.

5.414.3.2 `template<typename _Tp> template<typename _Up > std::complex< _Tp >::complex (const complex< _Up > & __z) [inline]`

Copy constructor.

Definition at line 136 of file `complex`.

5.414.4 Member Function Documentation

5.414.4.1 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator+=(const _Tp & __t) [inline]`

Add *t* to this complex number.

Definition at line 179 of file `complex`.

5.414.4.2 `template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator-= (const _Tp & __t) [inline]`

Subtract *t* from this complex number.

Definition at line 188 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

5.415 std::condition_variable Class Reference

[condition_variable](#)

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **condition_variable** (const [condition_variable](#) &)
- native_handle_type **native_handle** ()
- void **notify_all** ()
- void **notify_one** ()
- [condition_variable](#) & **operator=** (const [condition_variable](#) &)
- template<typename _Predicate >
void **wait** ([unique_lock](#)< [mutex](#) > &__lock, _Predicate __p)
- void **wait** ([unique_lock](#)< [mutex](#) > &__lock)
- template<typename _Rep, typename _Period, typename _Predicate >
bool **wait_for** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::duration](#)< _Rep, _Period > &__rtime, _Predicate __p)
- template<typename _Rep, typename _Period >
[cv_status](#) **wait_for** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<typename _Duration >
[cv_status](#) **wait_until** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::time_point](#)< __clock_t, _Duration > &__atime)
- template<typename _Clock, typename _Duration >
[cv_status](#) **wait_until** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- template<typename _Clock, typename _Duration, typename _Predicate >
bool **wait_until** ([unique_lock](#)< [mutex](#) > &__lock, const [chrono::time_point](#)< _Clock, _Duration > &__atime, _Predicate __p)

5.415.1 Detailed Description

[condition_variable](#)

Definition at line 57 of file condition_variable.

The documentation for this class was generated from the following file:

- [condition_variable](#)

5.416 std::condition_variable_any Class Reference

[condition_variable_any](#)

Public Types

- typedef condition_variable::native_handle_type **native_handle_type**

Public Member Functions

- **condition_variable_any** (const [condition_variable_any](#) &)
- native_handle_type **native_handle** ()
- void **notify_all** ()
- void **notify_one** ()
- [condition_variable_any](#) & **operator=** (const [condition_variable_any](#) &)
- template<typename _Lock , typename _Predicate >
void **wait** (_Lock &__lock, _Predicate __p)
- template<typename _Lock >
void **wait** (_Lock &__lock)
- template<typename _Lock , typename _Rep , typename _Period , typename _Predicate >
bool **wait_for** (_Lock &__lock, const [chrono::duration](#)< _Rep, _Period > &__rtime, _Predicate __p)
- template<typename _Lock , typename _Rep , typename _Period >
[cv_status](#) **wait_for** (_Lock &__lock, const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<typename _Lock , typename _Clock , typename _Duration >
[cv_status](#) **wait_until** (_Lock &__lock, const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- template<typename _Lock , typename _Clock , typename _Duration , typename _Predicate >
bool **wait_until** (_Lock &__lock, const [chrono::time_point](#)< _Clock, _Duration > &__atime, _Predicate __p)

5.416.1 Detailed Description

[condition_variable_any](#)

Definition at line 166 of file condition_variable.

The documentation for this class was generated from the following file:

- [condition_variable](#)

5.417 `std::conditional< _Cond, _Iftrue, _Iffalse >` Struct Template Reference

`conditional`

Public Types

- `typedef _Iftrue type`

5.417.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse> struct  
std::conditional< _Cond, _Iftrue, _Iffalse >
```

`conditional`

Definition at line 397 of file `type_traits`.

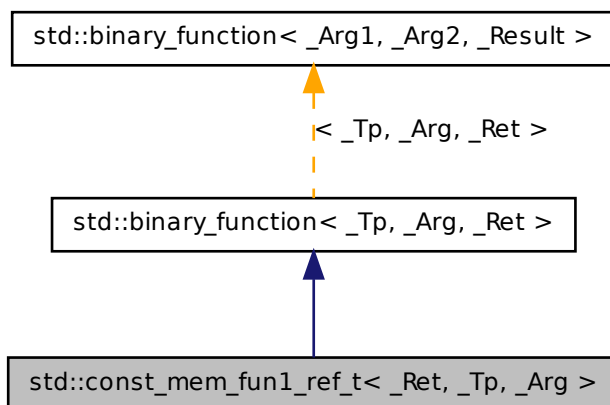
The documentation for this struct was generated from the following file:

- [type_traits](#)

5.418 `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Ret` [result_type](#)
- typedef `_Arg` [second_argument_type](#)

Public Member Functions

- `const_mem_fun1_ref_t` (`_Ret(_Tp::*__pf)(_Arg) const`)
- `_Ret operator()` (`const _Tp &__r, _Arg __x`) `const`

5.418.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`

One of the [adaptors for member /// pointers](#).

Definition at line 650 of file `stl_function.h`.

5.419 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

5.418.2 Member Typedef Documentation

5.418.2.1 `typedef _Tp std::binary_function< _Tp , _Arg , _Ret
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.418.2.2 `typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.418.2.3 `typedef _Arg std::binary_function< _Tp , _Arg , _Ret
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

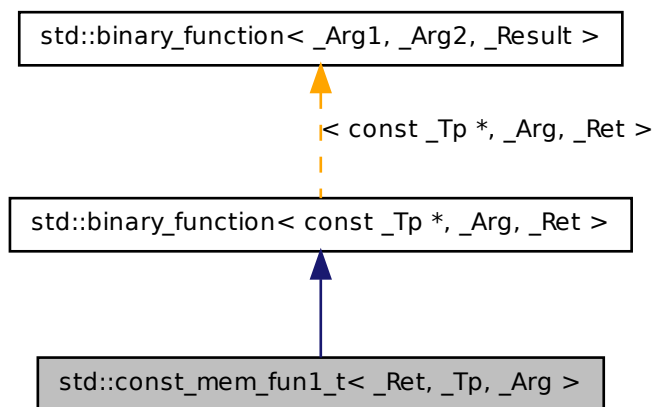
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.419 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`:



Public Types

- typedef const _Tp * [first_argument_type](#)
- typedef _Ret [result_type](#)
- typedef _Arg [second_argument_type](#)

Public Member Functions

- **const_mem_fun1_t** (_Ret(_Tp::*__pf)(_Arg) const)
- _Ret **operator()** (const _Tp *__p, _Arg __x) const

5.419.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg> class std::const_mem_fun1_t<_Ret, _Tp, _Arg>

One of the [adaptors for member /// pointers](#).

Definition at line 614 of file `stl_function.h`.

5.419.2 Member Typedef Documentation

5.419.2.1 `typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.419.2.2 `typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.419.2.3 `typedef _Arg std::binary_function< const _Tp *, _Arg, _Ret >::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

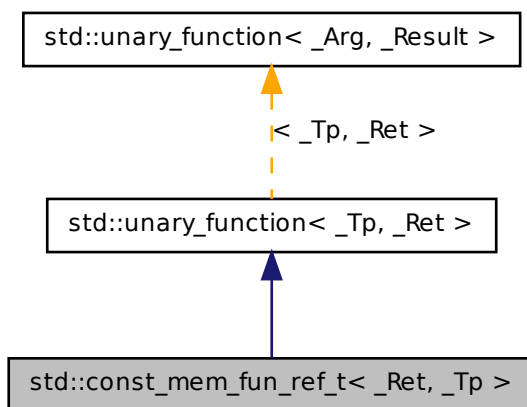
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.420 `std::const_mem_fun_ref_t< _Ret, _Tp >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun_ref_t< _Ret, _Tp >`:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `const_mem_fun_ref_t` (`_Ret`(`_Tp::*__pf`)() const)
- `_Ret operator()` (const `_Tp` &`__r`) const

5.420.1 Detailed Description

`template<typename _Ret, typename _Tp> class std::const_mem_fun_ref_t< _Ret, _Tp >`

One of the [adaptors for member /// pointers](#).

Definition at line 578 of file `stl_function.h`.

5.420.2 Member Typedef Documentation

5.420.2.1 `typedef _Tp std::unary_function<_Tp, _Ret>::argument_type` [`inherited`]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.420.2.2 `typedef _Ret std::unary_function<_Tp, _Ret>::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

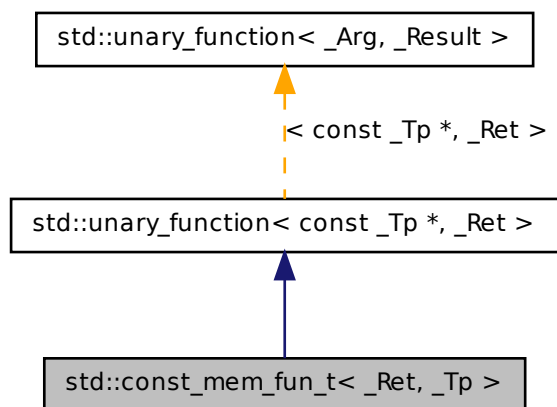
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.421 `std::const_mem_fun_t<_Ret, _Tp>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::const_mem_fun_t< _Ret, _Tp >`:



Public Types

- typedef `const _Tp *` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `const_mem_fun_t` (`_Ret(_Tp::*__pf)()` const)
- `_Ret operator()` (`const _Tp *__p`) const

5.421.1 Detailed Description

```
template<typename _Ret, typename _Tp> class std::const_mem_fun_t< _Ret,
_Tp >
```

One of the [adaptors for member /// pointers](#).

Definition at line 542 of file `stl_function.h`.

5.421.2 Member Typedef Documentation

5.421.2.1 `typedef const _Tp * std::unary_function< const _Tp *, _Ret >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.421.2.2 `typedef _Ret std::unary_function< const _Tp *, _Ret >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

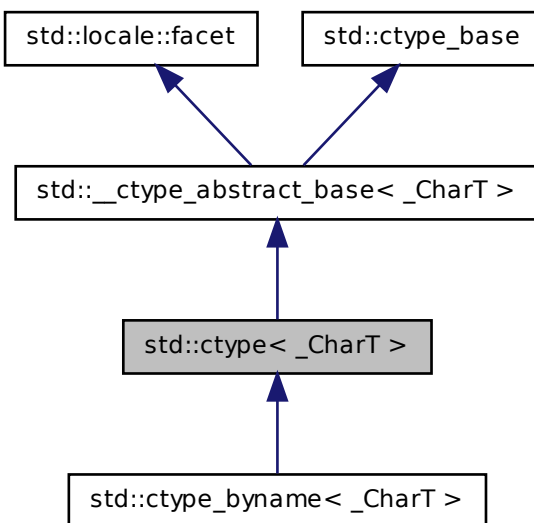
- [stl_function.h](#)

5.422 `std::ctype<_CharT>` Class Template Reference

Primary class template `ctype` facet.

This template class defines classification and conversion functions for character sets. It wraps `cctype` functionality. `Ctype` gets used by streams for many I/O operations.

Inheritance diagram for `std::ctype<_CharT>`:



Public Types

- typedef const int * **__to_type**
- typedef `_CharT` **char_type**
- typedef `__ctype_abstract_base<_CharT>::mask` **mask**

Public Member Functions

- **ctype** (size_t __refs=0)
- bool **is** (mask __m, **char_type** __c) const
- const **char_type** * **is** (const **char_type** *__lo, const **char_type** *__hi, mask *__-vec) const
- const **char_type** * **narrow** (const **char_type** *__lo, const **char_type** *__hi, **char** __default, **char** *__to) const
- **char** **narrow** (**char_type** __c, **char** __default) const
- const **char_type** * **scan_is** (mask __m, const **char_type** *__lo, const **char_type** *__hi) const

- const [char_type](#) * [scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [tolower](#) ([char_type](#) __c) const
- const [char_type](#) * [tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [toupper](#) ([char_type](#) __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, [char_type](#) * __to) const
- [char_type](#) [widen](#) (char __c) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool [do_is](#) (mask __m, [char_type](#) __c) const
- virtual const [char_type](#) * [do_is](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __dfault, char * __dest) const
- virtual char [do_narrow](#) ([char_type](#), char __dfault) const
- virtual const [char_type](#) * [do_scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual const [char_type](#) * [do_scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#) __c) const
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#) __c) const

- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, [char_type](#) *__dest) const
- virtual [char_type do_widen](#) (char __c) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.422.1 Detailed Description

template<typename _CharT> class std::ctype< _CharT >

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations. This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in [__ctype_abstract_base](#), to allow for implementation flexibility. See [ctype<wchar_t>](#) for an example. The functions are documented in [__ctype_abstract_base](#).

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 604 of file locale_facets.h.

5.422.2 Member Typedef Documentation

5.422.2.1 **template<typename _CharT> typedef _CharT std::ctype< _CharT >::char_type**

Typedef for the template parameter.

Reimplemented from [std::__ctype_abstract_base< _CharT >](#).

Definition at line 608 of file locale_facets.h.

5.422.3 Member Function Documentation

5.422.3.1 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const [protected, virtual]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

[do_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do_is\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.2 `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is (mask __m, char_type __c) const [protected, virtual]`

Test char_type classification.

This function finds a mask M for c and compares it to mask m.

[do_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do_is\(\)](#) must always return the same result for the same input.

Parameters

c The char_type to find the mask of.

m The mask to compare against.

Returns

(M & m) != 0.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.422.3.3 `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow (char_type, char __dfault) const [protected, virtual]`

Narrow char_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead.

[do_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char_type to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.422.3.4 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, virtual]`

Narrow char_type array to char.

This virtual function converts each char_type in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *dfault* is used instead.

[do_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

dfault Char to use if conversion fails.

to Pointer to the destination array.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.5 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual]`

Find char_type matching mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *hi*.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.6 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual]`

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.
lo Pointer to start of range.
hi Pointer to end of range.

Returns

Pointer to a non-matching `char_type` if found, else *hi*.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.7 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower (char_type * __lo, const char_type * __hi) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.
hi Pointer to end of range.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.8 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower (char_type) const [protected, virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The lowercase char_type if convertible, else *c*.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.9 `template<typename _CharT> virtual const char_type* std::ctype<
_CharT >::do_toupper(char_type * __lo, const char_type * __hi)
const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.10 `template<typename _CharT> virtual char_type std::ctype<
_CharT >::do_toupper(char_type) const [protected,
virtual]`

Convert to uppercase.

This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The char_type to convert.

Returns

The uppercase char_type if convertible, else *c*.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.11 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen (char) const [protected, virtual]`

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.12 `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [protected, virtual]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.422.3.13 `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is(mask __m, char_type __c) const [inline, inherited]`

Test char_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

c The char_type to compare the mask of.

m The mask to compare against.

Returns

(*M* & *m*) != 0.

Definition at line 161 of file locale_facets.h.

Referenced by `std::regex_traits< _Ch_type >::isctype()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

5.422.3.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is(const char_type * __lo, const char_type * __hi, mask * __vec) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char_type in the range [*lo*,*hi*) and successively writes it to *vec*. *vec* must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Definition at line 178 of file locale_facets.h.

5.422.3.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow (char_type __c, char __dfault) const
[inline, inherited]`

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

Returns

The converted `char`.

Definition at line 323 of file `locale_facets.h`.

Referenced by `std::time_put<_CharT, _OutIter>::put()`.

5.422.3.16 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::narrow (const char_type
* __lo, const char_type * __hi, char __dfault, char * __to) const
[inline, inherited]`

Narrow array to `char` array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(lo, hi, dfault, to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 345 of file locale_facets.h.

```
5.422.3.17 template<typename _CharT> const char_type*  
std::__ctype_abstract_base< _CharT >::scan_is ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]
```

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning [ctype<char_type>::do_scan_is\(\)](#).

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to matching char_type if found, else *hi*.

Definition at line 194 of file locale_facets.h.

```
5.422.3.18 template<typename _CharT> const char_type*  
std::__ctype_abstract_base< _CharT >::scan_not ( mask __m,  
const char_type * __lo, const char_type * __hi ) const [inline,  
inherited]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char_type>::do_scan_not\(\)](#).

Parameters

m The mask to compare against.

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale_facets.h.

5.422.3.19 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::tolower (char_type *
__lo, const char_type * __hi) const [inline, inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 268 of file `locale_facets.h`.

5.422.3.20 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower (char_type __c) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper(c)`.

Parameters

c The `char_type` to convert.

Returns

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

5.422.3.21 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::toupper (char_type *
__lo, const char_type * __hi) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 239 of file `locale_facets.h`.

5.422.3.22 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper (char_type __c) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

5.422.3.23 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen (char __c) const [inline, inherited]`

Widen char to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type.

Definition at line 285 of file locale_facets.h.

Referenced by std::money_get<_CharT, _InIter>::do_get(), std::time_put<_CharT, _OutIter>::do_put(), std::money_put<_CharT, _OutIter>::do_put(), std::regex_traits<_Ch_type>::istype(), and std::operator<<().

5.422.3.24 `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen (const char * __lo, const char * __hi, char_type * __to) const [inline, inherited]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 304 of file locale_facets.h.

5.422.4 Member Data Documentation

5.422.4.1 `template<typename _CharT> locale::id std::ctype<_CharT>::id [static]`

The facet id for ctype<char_type>

Definition at line 612 of file locale_facets.h.

The documentation for this class was generated from the following file:

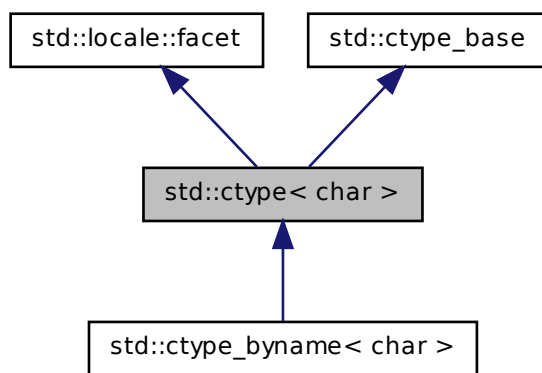
- [locale_facets.h](#)

5.423 std::ctype< char > Class Template Reference

The `ctype<char>` specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Inheritance diagram for std::ctype< char >:



Public Types

- typedef const int * **__to_type**
- typedef char `char_type`
- typedef unsigned short **mask**

Public Member Functions

- `ctype` (const mask *__table=0, bool __del=false, size_t __refs=0)
- `ctype` (__c_locale __cloc, const mask *__table=0, bool __del=false, size_t __refs=0)
- const char * **is** (const char *__lo, const char *__hi, mask *__vec) const
- bool **is** (mask __m, char __c) const
- char **narrow** (`char_type` __c, char __default) const

- const [char_type](#) * [narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __default, char * __to) const
- const char * [scan_is](#) (mask __m, const char * __lo, const char * __hi) const
- const char * [scan_not](#) (mask __m, const char * __lo, const char * __hi) const
- const mask * [table](#) () const throw ()
- [char_type](#) tolower ([char_type](#) __c) const
- const [char_type](#) * [tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) toupper ([char_type](#) __c) const
- [char_type](#) widen (char __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, [char_type](#) * __to) const

Static Public Member Functions

- static const mask * [classic_table](#) () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t [table_size](#)
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual [~ctype](#) ()
- virtual char [do_narrow](#) ([char_type](#) __c, char) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char, char * __dest) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#)) const
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#)) const

- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual [char_type](#) [do_widen](#) (char __c) const
- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, [char_type](#) *__dest) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale [_M_c_locale_ctype](#)
- bool [_M_del](#)
- char [_M_narrow](#) [1+static_cast< unsigned char >(-1)]
- char [_M_narrow_ok](#)
- const mask * [_M_table](#)
- __to_type [_M_tolower](#)
- __to_type [_M_toupper](#)
- char [_M_widen](#) [1+static_cast< unsigned char >(-1)]
- char [_M_widen_ok](#)

Friends

- class [locale::_Impl](#)

5.423.1 Detailed Description

`template<> class std::ctype< char >`

The [ctype<char>](#) specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Definition at line 673 of file `locale_facets.h`.

5.423.2 Member Typedef Documentation

5.423.2.1 `typedef char std::ctype< char >::char_type`

Typedef for the template parameter char.

Definition at line 678 of file locale_facets.h.

5.423.3 Constructor & Destructor Documentation

5.423.3.1 `std::ctype< char >::ctype (const mask * __table = 0, bool __del = false, size_t __refs = 0) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

table If non-zero, table is used as the per-char mask. Else [classic_table\(\)](#) is used.

del If true, passes ownership of table to this facet.

refs Passed to the base facet class.

5.423.3.2 `std::ctype< char >::ctype (__c_locale __cloc, const mask * __table = 0, bool __del = false, size_t __refs = 0) [explicit]`

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

cloc Handle to C locale data.

table If non-zero, table is used as the per-char mask.

del If true, passes ownership of table to this facet.

refs Passed to the base facet class.

5.423.3.3 `virtual std::ctype< char >::~~ctype () [protected, virtual]`

Destructor.

This function deletes [table\(\)](#) if *del* was true in the constructor.

5.423.4 Member Function Documentation

5.423.4.1 `static const mask* std::ctype< char >::classic_table () throw ()`
`[static]`

Returns a pointer to the C locale mask table.

5.423.4.2 `virtual char std::ctype< char >::do_narrow (char_type __c, char`
`)const [inline, protected, virtual]`

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

Definition at line 1123 of file `locale_facets.h`.

5.423.4.3 `virtual const char_type* std::ctype< char >::do_narrow (const`
`char_type * __lo, const char_type * __hi, char, char * __dest)`
`const [inline, protected, virtual]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.
hi Pointer to end of range.
dfault Char to use if conversion fails.
to Pointer to the destination array.

Returns

hi.

Definition at line 1149 of file locale_facets.h.

5.423.4.4 `virtual const char_type* std::ctype< char >::do_tolower (char_type
 * __lo, const char_type * __hi) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.
hi Pointer to end of range.

Returns

hi.

5.423.4.5 `virtual char_type std::ctype< char >::do_tolower (char_type)
 const [protected, virtual]`

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The lowercase char if convertible, else *c*.

5.423.4.6 `virtual const char_type* std::ctype< char >::do_toupper (char_type
* __lo, const char_type* __hi) const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.423.4.7 `virtual char_type std::ctype< char >::do_toupper (char_type)
const [protected, virtual]`

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The uppercase char if convertible, else *c*.

5.423.4.8 **virtual** char_type std::ctype< char >::do_widen (char __c) const [inline, protected, virtual]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be returned unchanged.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted character.

Definition at line 1074 of file `locale_facets.h`.

5.423.4.9 **virtual const** char* std::ctype< char >::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [inline, protected, virtual]

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be copied unchanged.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 1097 of file `locale_facets.h`.

5.423.4.10 `const char * std::ctype< char >::is (const char * __lo, const char * __hi, mask * __vec) const [inline]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

Definition at line 46 of file ctype_inline.h.

5.423.4.11 `bool std::ctype< char >::is (mask __m, char __c) const [inline]`

Test char classification.

This function compares the mask table[c] to *m*.

Parameters

c The char to compare the mask of.

m The mask to compare against.

Returns

True if *m* & table[c] is true, false otherwise.

Definition at line 41 of file ctype_inline.h.

5.423.4.12 `char std::ctype< char >::narrow (char_type __c, char __dfault) const [inline]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The char to convert.
- dfault* Char to return if conversion fails.

Returns

The converted character.

Definition at line 922 of file `locale_facets.h`.

5.423.4.13 `const char_type* std::ctype<char>::narrow (const char_type *
__lo, const char_type * __hi, char __dfault, char * __to) const
[inline]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 955 of file `locale_facets.h`.

5.423.4.14 `const char * std::ctype< char >::scan_is (mask __m, const char * __lo, const char * __hi) const [inline]`

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char if found, else *hi*.

Definition at line 55 of file ctype_inline.h.

5.423.4.15 `const char * std::ctype< char >::scan_not (mask __m, const char * __lo, const char * __hi) const [inline]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [lo,hi) for which is(m,char) is false.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching char if found, else *hi*.

Definition at line 65 of file ctype_inline.h.

5.423.4.16 `const mask* std::ctype< char >::table () const throw () [inline]`

Returns a pointer to the mask table provided to the constructor, or /// the default from [classic_table\(\)](#) if none was provided.

Definition at line 973 of file locale_facets.h.

5.423.4.17 `char_type std::ctype< char >::tolower (char_type __c) const` **[inline]**

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`tolower()` acts as if it returns `ctype<char>::do_tolower(c)`. `do_tolower()` must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The lowercase char if convertible, else *c*.

Definition at line 827 of file locale_facets.h.

5.423.4.18 `const char_type* std::ctype< char >::tolower (char_type * __lo, const char_type * __hi) const` **[inline]**

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(lo, hi)`. `do_tolower()` must always return the same result for the same input.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

hi.

Definition at line 844 of file locale_facets.h.

5.423.4.19 `const char_type* std::ctype< char >::toupper (char_type * __lo, const char_type * __hi) const` **[inline]**

Convert array to uppercase.

This function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

[toupper\(\)](#) acts as if it returns `ctype<char>::do_toupper(lo, hi)`. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

hi.

Definition at line 811 of file locale_facets.h.

5.423.4.20 char_type std::ctype< char >::toupper (char_type __c) const [inline]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[toupper\(\)](#) acts as if it returns `ctype<char>::do_toupper(c)`. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The uppercase char if convertible, else *c*.

Definition at line 794 of file locale_facets.h.

5.423.4.21 char_type std::ctype< char >::widen (char __c) const [inline]

Widen char.

This function converts the char to char_type using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted character.

Definition at line 864 of file locale_facets.h.

5.423.4.22 `const char* std::ctype< char >::widen (const char * __lo, const char * __hi, char_type * __to) const [inline]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 891 of file locale_facets.h.

5.423.5 Member Data Documentation

5.423.5.1 `locale::id std::ctype< char >::id [static]`

The facet id for `ctype<char>`

Definition at line 695 of file locale_facets.h.

5.423.5.2 `const size_t std::ctype< char >::table_size [static]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 697 of file locale_facets.h.

The documentation for this class was generated from the following files:

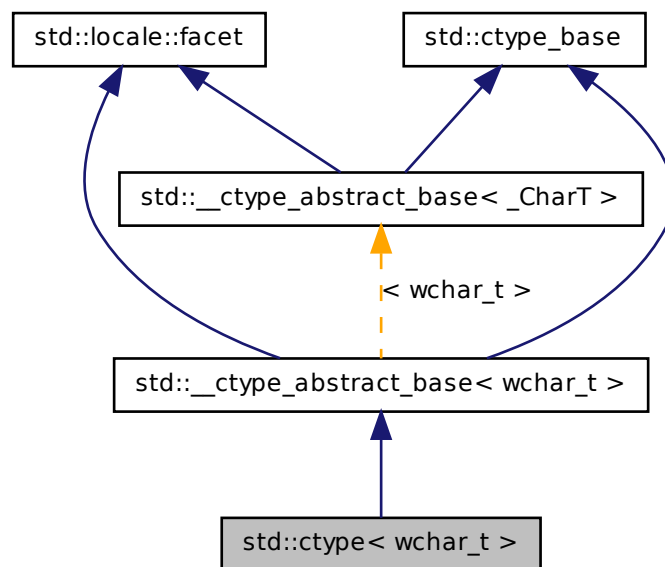
- [locale_facets.h](#)
- [ctype_inline.h](#)

5.424 std::ctype< wchar_t > Class Template Reference

The [ctype<wchar_t>](#) specialization.

This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.

Inheritance diagram for std::ctype< wchar_t >:



Public Types

- typedef const int * **__to_type**
- typedef wctype_t **__wmask_type**
- typedef wchar_t [char_type](#)
- typedef unsigned short **mask**

Public Member Functions

- [ctype](#) (size_t __refs=0)
- [ctype](#) (__c_locale __cloc, size_t __refs=0)
- const [char_type](#) * **is** (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const
- bool **is** (mask __m, [char_type](#) __c) const
- char **narrow** ([char_type](#) __c, char __dfault) const
- const [char_type](#) * **narrow** (const [char_type](#) * __lo, const [char_type](#) * __hi, char __dfault, char * __to) const
- const [char_type](#) * **scan_is** (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * **scan_not** (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) **tolower** ([char_type](#) __c) const
- const [char_type](#) * **tolower** ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) **toupper** ([char_type](#) __c) const
- const [char_type](#) * **toupper** ([char_type](#) * __lo, const [char_type](#) * __hi) const
- const char * **widen** (const char * __lo, const char * __hi, [char_type](#) * __to) const
- [char_type](#) **widen** (char __c) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual `~ctype()`
- `__wmask_type _M_convert_to_wmask (const mask __m) const` throw ()
- `void _M_initialize_ctype ()` throw ()
- virtual const `char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const`
- virtual bool `do_is (mask __m, char_type __c) const` =0
- virtual const `char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const` =0
- virtual bool `do_is (mask __m, char_type __c) const`
- virtual char `do_narrow (char_type, char __dfault) const`
- virtual char `do_narrow (char_type, char __dfault) const` =0
- virtual const `char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__dest) const` =0
- virtual const `char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__dest) const`
- virtual const `char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const` =0
- virtual const `char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const`
- virtual const `char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const`
- virtual const `char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const` =0
- virtual const `char_type * do_tolower (char_type *__lo, const char_type *__hi) const`
- virtual `char_type do_tolower (char_type) const` =0
- virtual const `char_type * do_tolower (char_type *__lo, const char_type *__hi) const` =0
- virtual `char_type do_tolower (char_type) const`
- virtual `char_type do_toupper (char_type) const`
- virtual const `char_type * do_toupper (char_type *__lo, const char_type *__hi) const`
- virtual `char_type do_toupper (char_type) const` =0
- virtual const `char_type * do_toupper (char_type *__lo, const char_type *__hi) const` =0
- virtual const char * `do_widen (const char *__lo, const char *__hi, char_type *__dest) const` =0
- virtual `char_type do_widen (char) const`
- virtual const char * `do_widen (const char *__lo, const char *__hi, char_type *__dest) const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Protected Attributes

- mask `_M_bit` [16]
- `__c_locale _M_c_locale_ctype`
- char `_M_narrow` [128]
- bool `_M_narrow_ok`
- `wint_t _M_widen` [1+static_cast< unsigned char >(-1)]
- `__wmask_type _M_wmask` [16]

Friends

- class `locale::_Impl`

5.424.1 Detailed Description

`template<> class std::ctype< wchar_t >`

The [ctype<wchar_t>](#) specialization.

This class defines classification and conversion functions for the `wchar_t` type. It gets used by `wchar_t` streams for many I/O operations. The `wchar_t` specialization provides a number of optimizations as well. [ctype<wchar_t>](#) inherits its public methods from [__ctype_abstract_base<wchar_t>](#).

Definition at line 1174 of file `locale_facets.h`.

5.424.2 Member Typedef Documentation

5.424.2.1 `typedef wchar_t std::ctype< wchar_t >::char_type`

Typedef for the template parameter `wchar_t`.

Reimplemented from [std::__ctype_abstract_base< wchar_t >](#).

Definition at line 1179 of file `locale_facets.h`.

5.424.3 Constructor & Destructor Documentation

5.424.3.1 std::ctype< wchar_t >::ctype (size_t __refs = 0) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

5.424.3.2 std::ctype< wchar_t >::ctype (__c_locale __cloc, size_t __refs = 0) [explicit]

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

cloc Handle to C locale data.

refs Passed to the base facet class.

5.424.3.3 virtual std::ctype< wchar_t >::~~ctype () [protected, virtual]

Destructor.

5.424.4 Member Function Documentation

5.424.4.1 virtual bool std::ctype< wchar_t >::do_is (mask __m, char_type __c) const [protected, virtual]

Test wchar_t classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

[do_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do_is\(\)](#) must always return the same result for the same input.

Parameters

c The wchar_t to find the mask of.

m The mask to compare against.

Returns

$(M \& m) \neq 0$.

5.424.4.2 `virtual bool std::__ctype_abstract_base< wchar_t >::do_is (mask
__m, char_type __c) const [protected, pure virtual,
inherited]`

Test char_type classification.

This function finds a mask M for c and compares it to mask m .

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

c The char_type to find the mask of.

m The mask to compare against.

Returns

$(M \& m) \neq 0$.

5.424.4.3 `virtual const char_type* std::ctype< wchar_t >::do_is (const
char_type * __lo, const char_type * __hi, mask * __vec) const
[protected, virtual]`

Return a mask array.

This function finds the mask for each `wchar_t` in the range $[lo, hi)$ and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi .

5.424.4.4 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_is(const char_type * __lo, const char_type * __hi, mask * __vec) const` `[protected, pure virtual, inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

vec Pointer to an array of mask storage.

Returns

hi.

5.424.4.5 `virtual char std::__ctype_abstract_base< wchar_t >::do_narrow(char_type, char __dfault) const` `[protected, pure virtual, inherited]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The `char_type` to convert.

dfault Char to return if conversion fails.

Returns

The converted `char`.

5.424.4.6 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, pure virtual, inherited]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *dfault* is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

5.424.4.7 `virtual char std::ctype< wchar_t >::do_narrow (char_type, char __dfault) const [protected, virtual]`

Narrow `wchar_t` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<wchar_t>` facet, *c* will be cast to `char` and returned.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `wchar_t` to convert.
- dfault* Char to return if conversion fails.

Returns

The converted `char`.

5.424.4.8 `virtual const char_type* std::ctype< wchar_t >::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, virtual]`

Narrow wchar_t array to char array.

This virtual function converts each wchar_t in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any wchar_t in the input that cannot be converted, *dfault* is used instead. For an undervived `ctype<wchar_t>` facet, the argument will be copied, casting each element to char.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

5.424.4.9 `virtual const char_type* std::ctype< wchar_t >::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual]`

Find wchar_t matching mask.

This function searches for and returns the first wchar_t c in [lo,hi) for which `is(m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to a matching wchar_t if found, else *hi*.

5.424.4.10 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [protected, pure virtual, inherited]`

Find char_type matching mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *hi*.

5.424.4.11 `virtual const char_type* std::ctype< wchar_t >::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual]`

Find wchar_t not matching mask.

This function searches for and returns a pointer to the first wchar_t c of [lo,hi) for which is(m,c) is false.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to a non-matching wchar_t if found, else *hi*.

5.424.4.12 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_scan_not(mask __m, const char_type * __lo, const char_type * __hi) const [protected, pure virtual, inherited]`

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *hi*.

5.424.4.13 `virtual char_type std::ctype< wchar_t >::do_tolower(char_type) const [protected, virtual]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

c The wchar_t to convert.

Returns

The lowercase wchar_t if convertible, else *c*.

5.424.4.14 `virtual char_type std::__ctype_abstract_base< wchar_t >::do_tolower(char_type) const [protected, pure virtual, inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The lowercase `char_type` if convertible, else *c*.

5.424.4.15 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_tolower (char_type * __lo, const char_type * __hi) const [protected, pure virtual, inherited]`

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.424.4.16 `virtual const char_type* std::ctype< wchar_t >::do_tolower (char_type * __lo, const char_type * __hi) const [protected, virtual]`

Convert array to lowercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

[`do_tolower\(\)`](#) is a hook for a derived facet to change the behavior of lowercasing. [`do_tolower\(\)`](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.424.4.17 `virtual char_type std::__ctype_abstract_base< wchar_t
>::do_toupper (char_type) const [protected, pure
virtual, inherited]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

5.424.4.18 `virtual char_type std::ctype< wchar_t >::do_toupper (char_type
) const [protected, virtual]`

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

c The `wchar_t` to convert.

Returns

The uppercase `wchar_t` if convertible, else *c*.

5.424.4.19 `virtual const char_type* std::__ctype_abstract_base< wchar_t >::do_toupper (char_type * __lo, const char_type * __hi) const [protected, pure virtual, inherited]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.424.4.20 `virtual const char_type* std::ctype< wchar_t >::do_toupper (char_type * __lo, const char_type * __hi) const [protected, virtual]`

Convert array to uppercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[`do_toupper\(\)`](#) is a hook for a derived facet to change the behavior of uppercasing. [`do_toupper\(\)`](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.424.4.21 `virtual char_type std::ctype< wchar_t >::do_widen (char) const [protected, virtual]`

Widen `char` to `wchar_t`.

This virtual function converts the char to wchar_t using the simplest reasonable transformation. For an underived [ctype<wchar_t>](#) facet, the argument will be cast to wchar_t.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted wchar_t.

Implements [std::__ctype_abstract_base< wchar_t >](#).

5.424.4.22 `virtual const char* std::ctype< wchar_t >::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [protected, virtual]`

Widen char array to wchar_t array.

This function converts each char in the input to wchar_t using the simplest reasonable transformation. For an underived [ctype<wchar_t>](#) facet, the argument will be copied, casting each element to wchar_t.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

5.424.4.23 `virtual const char* std::__ctype_abstract_base< wchar_t >::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [protected, pure virtual, inherited]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

lo Pointer to start range.
hi Pointer to end of range.
to Pointer to the destination array.

Returns

hi.

5.424.4.24 `const char_type* std::__ctype_abstract_base< wchar_t >::is (const char_type * __lo, const char_type * __hi, mask * __vec) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

lo Pointer to start of range.
hi Pointer to end of range.
vec Pointer to an array of mask storage.

Returns

hi.

Definition at line 178 of file locale_facets.h.

5.424.4.25 `bool std::__ctype_abstract_base< wchar_t >::is (mask __m, char_type __c) const [inline, inherited]`

Test char_type classification.

This function finds a mask M for c and compares it to mask m. It does so by returning the value of ctype<char_type>::do_is().

Parameters

- c* The char_type to compare the mask of.
m The mask to compare against.

Returns

(M & m) != 0.

Definition at line 161 of file locale_facets.h.

5.424.4.26 `const char_type* std::__ctype_abstract_base< wchar_t >::narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __to) const [inline, inherited]`

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

- lo* Pointer to start of range.
hi Pointer to end of range.
dfault Char to use if conversion fails.
to Pointer to the destination array.

Returns

hi.

Definition at line 345 of file locale_facets.h.

5.424.4.27 `char std::__ctype_abstract_base< wchar_t >::narrow (char_type __c, char __dfault) const [inline, inherited]`

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. It does so by returning ctype<char_type>::do_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

c The char_type to convert.
dfault Char to return if conversion fails.

Returns

The converted char.

Definition at line 323 of file locale_facets.h.

5.424.4.28 `const char_type* std::__ctype_abstract_base< wchar_t >::scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [inline, inherited]`

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

m The mask to compare against.
lo Pointer to start of range.
hi Pointer to end of range.

Returns

Pointer to matching char_type if found, else *hi*.

Definition at line 194 of file locale_facets.h.

5.424.4.29 `const char_type* std::__ctype_abstract_base< wchar_t >::scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [inline, inherited]`

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

m The mask to compare against.
lo Pointer to first char in range.
hi Pointer to end of range.

Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale_facets.h.

5.424.4.30 `const char_type* std::__ctype_abstract_base< wchar_t >::tolower (char_type * __lo, const char_type * __hi) const [inline, inherited]`

Convert array to lowercase.

This function converts each char_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>::do_toupper(lo, hi).

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 268 of file locale_facets.h.

5.424.4.31 `char_type std::__ctype_abstract_base< wchar_t >::tolower (char_type __c) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

c The char_type to convert.

Returns

The lowercase char_type if convertible, else *c*.

Definition at line 253 of file locale_facets.h.

5.424.4.32 `const char_type* std::__ctype_abstract_base< wchar_t >::toupper (char_type * __lo, const char_type * __hi) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 239 of file `locale_facets.h`.

5.424.4.33 `char_type std::__ctype_abstract_base< wchar_t >::toupper (char_type __c) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

5.424.4.34 `char_type std::__ctype_abstract_base< wchar_t >::widen (char __c) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type.

Definition at line 285 of file locale_facets.h.

5.424.4.35 `const char* std::__ctype_abstract_base< wchar_t >::widen (const char * __lo, const char * __hi, char_type * __to) const [inline, inherited]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecv for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 304 of file locale_facets.h.

5.424.5 Member Data Documentation

5.424.5.1 `locale::id std::ctype< wchar_t >::id [static]`

The facet id for [ctype<wchar_t>](#)

Definition at line 1197 of file locale_facets.h.

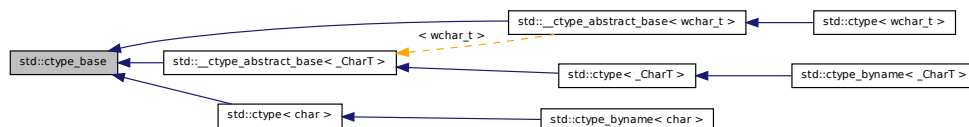
The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.425 std::ctype_base Struct Reference

Base class for ctype.

Inheritance diagram for std::ctype_base:



Public Types

- typedef const int * **__to_type**
- typedef unsigned short **mask**

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

5.425.1 Detailed Description

Base class for ctype.

Definition at line 40 of file ctype_base.h.

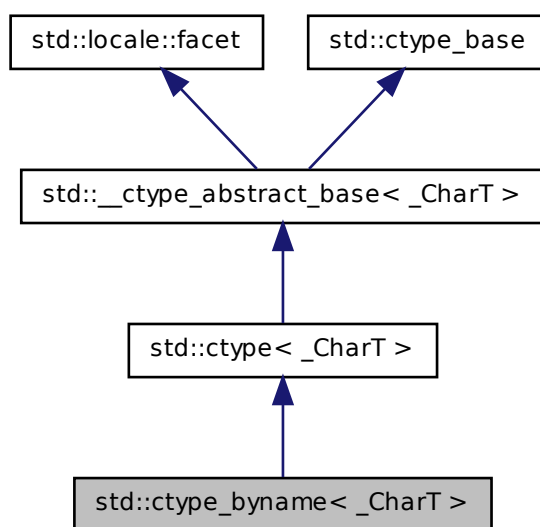
The documentation for this struct was generated from the following file:

- [ctype_base.h](#)

5.426 std::ctype_byname< _CharT > Class Template Reference

class [ctype_byname](#) [22.2.1.2].

Inheritance diagram for std::ctype_byname< _CharT >:



Public Types

- typedef const int * **__to_type**
- typedef _CharT [char_type](#)
- typedef [ctype](#)< _CharT >::mask **mask**

Public Member Functions

- **ctype_byname** (const char * __s, size_t __refs=0)
- bool **is** (mask __m, [char_type](#) __c) const
- const [char_type](#) * **is** (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __-vec) const

- const [char_type](#) * [narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __default, char * __to) const
- char [narrow](#) ([char_type](#) __c, char __default) const
- const [char_type](#) * [scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [tolower](#) ([char_type](#) __c) const
- const [char_type](#) * [tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- const [char_type](#) * [toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- [char_type](#) [toupper](#) ([char_type](#) __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, [char_type](#) * __to) const
- [char_type](#) [widen](#) (char __c) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool [do_is](#) (mask __m, [char_type](#) __c) const
- virtual const [char_type](#) * [do_is](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, char __default, char * __dest) const
- virtual char [do_narrow](#) ([char_type](#), char __default) const
- virtual const [char_type](#) * [do_scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual const [char_type](#) * [do_scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#) __c) const

- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#) __c) const
- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, [char_type](#) *__dest) const
- virtual [char_type](#) [do_widen](#) (char __c) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.426.1 Detailed Description

template<typename _CharT> class std::ctype_byname<_CharT>

class [ctype_byname](#) [22.2.1.2].

Definition at line 1466 of file locale_facets.h.

5.426.2 Member Typedef Documentation

5.426.2.1 **template<typename _CharT> typedef _CharT std::ctype<_CharT>::char_type [inherited]**

Typedef for the template parameter.

Reimplemented from [std::__ctype_abstract_base<_CharT>](#).

Definition at line 608 of file locale_facets.h.

5.426.3 Member Function Documentation

5.426.3.1 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const [protected, virtual, inherited]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

[do_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do_is\(\)](#) must always return the same result for the same input.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.426.3.2 `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is (mask __m, char_type __c) const [protected, virtual, inherited]`

Test char_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

[do_is\(\)](#) is a hook for a derived facet to change the behavior of classifying. [do_is\(\)](#) must always return the same result for the same input.

Parameters

- c* The char_type to find the mask of.
- m* The mask to compare against.

Returns

(M & m) != 0.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.426.3.3 `template<typename _CharT> virtual char std::ctype< _CharT >::do_narrow (char_type, char __dfault) const [protected, virtual, inherited]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The `char_type` to convert.
- dfault* Char to return if conversion fails.

Returns

The converted `char`.

Implements `std::__ctype_abstract_base< _CharT >`.

5.426.3.4 `template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __dest) const [protected, virtual, inherited]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.426.3.5 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Find char_type matching mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *hi*.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.426.3.6 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

[do_scan_is\(\)](#) is a hook for a derived facet to change the behavior of match searching. [do_is\(\)](#) must always return the same result for the same input.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *hi*.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.426.3.7 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower(char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Convert array to lowercase.

This virtual function converts each char_type in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.426.3.8 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower(char_type) const [protected, virtual, inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char_type to convert.

Returns

The lowercase `char_type` if convertible, else `c`.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.426.3.9 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper(char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.426.3.10 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper(char_type) const [protected, virtual, inherited]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else `c`.

Implements [std::__ctype_abstract_base< _CharT >](#).

5.426.3.11 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen (char) const [protected, virtual, inherited]`

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type

Implements [std::__ctype_abstract_base<_CharT>](#).

5.426.3.12 `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen (const char * __lo, const char * __hi, char_type * __dest) const [protected, virtual, inherited]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

5.426.3.13 `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is (mask __m, char_type __c) const [inline, inherited]`

Test char_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

- c* The char_type to compare the mask of.
- m* The mask to compare against.

Returns

(*M* & *m*) != 0.

Definition at line 161 of file locale_facets.h.

Referenced by `std::regex_traits<_Ch_type>::isctype()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

5.426.3.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is (const char_type * __lo, const char_type * __hi, mask * __vec) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char_type in the range [*lo*,*hi*) and successively writes it to *vec*. *vec* must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

Returns

hi.

Definition at line 178 of file locale_facets.h.

5.426.3.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow (char_type __c, char __dfault) const
[inline, inherited]`

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

- c* The char_type to convert.
- dfault* Char to return if conversion fails.

Returns

The converted char.

Definition at line 323 of file locale_facets.h.

Referenced by std::time_put< _CharT, _OutIter >::put().

5.426.3.16 `template<typename _CharT> const char_type*
std::__ctype_abstract_base< _CharT >::narrow (const char_type
* __lo, const char_type * __hi, char __dfault, char * __to) const
[inline, inherited]`

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char_type>::do_narrow(lo, hi, dfault, to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 345 of file locale_facets.h.

```
5.426.3.17  template<typename _CharT> const char_type*
             std::__ctype_abstract_base<_CharT>::scan_is ( mask __m,
             const char_type * __lo, const char_type * __hi ) const  [inline,
             inherited]
```

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning [ctype<char_type>::do_scan_is\(\)](#).

Parameters

- m* The mask to compare against.
- lo* Pointer to start of range.
- hi* Pointer to end of range.

Returns

Pointer to matching char_type if found, else *hi*.

Definition at line 194 of file locale_facets.h.

```
5.426.3.18  template<typename _CharT> const char_type*
             std::__ctype_abstract_base<_CharT>::scan_not ( mask __m,
             const char_type * __lo, const char_type * __hi ) const  [inline,
             inherited]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning [ctype<char_type>::do_scan_not\(\)](#).

Parameters

- m* The mask to compare against.
- lo* Pointer to first char in range.
- hi* Pointer to end of range.

Returns

Pointer to non-matching char if found, else *hi*.

Definition at line 210 of file locale_facets.h.

5.426.3.19 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::tolower (char_type *
__lo, const char_type * __hi) const [inline, inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 268 of file `locale_facets.h`.

5.426.3.20 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower (char_type __c) const [inline, inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

Parameters

c The `char_type` to convert.

Returns

The lowercase `char_type` if convertible, else *c*.

Definition at line 253 of file `locale_facets.h`.

5.426.3.21 `template<typename _CharT> const char_type*
std::__ctype_abstract_base<_CharT>::toupper (char_type *
__lo, const char_type * __hi) const [inline, inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

Definition at line 239 of file `locale_facets.h`.

5.426.3.22 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper (char_type __c) const [inline, inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

c The `char_type` to convert.

Returns

The uppercase `char_type` if convertible, else *c*.

Definition at line 224 of file `locale_facets.h`.

5.426.3.23 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen (char __c) const [inline, inherited]`

Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted char_type.

Definition at line 285 of file locale_facets.h.

Referenced by std::money_get<_CharT, _InIter>::do_get(), std::time_put<_CharT, _OutIter>::do_put(), std::money_put<_CharT, _OutIter>::do_put(), std::regex_traits<_Ch_type>::isctype(), and std::operator<<().

5.426.3.24 `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen (const char * __lo, const char * __hi, char_type * __to) const [inline, inherited]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 304 of file locale_facets.h.

5.426.4 Member Data Documentation

5.426.4.1 `template<typename _CharT> locale::id std::ctype<_CharT>::id [static, inherited]`

The facet id for ctype<char_type>

Definition at line 612 of file locale_facets.h.

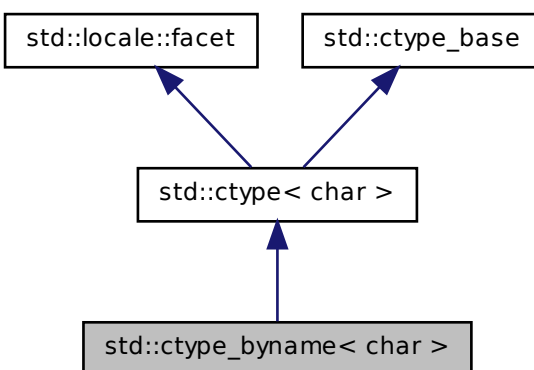
The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.427 `std::ctype_byname< char >` Class Template Reference

22.2.1.4 Class `ctype_byname` specializations.

Inheritance diagram for `std::ctype_byname< char >`:



Public Types

- typedef const int * `__to_type`
- typedef char `char_type`
- typedef unsigned short `mask`

Public Member Functions

- `ctype_byname` (const char * __s, size_t __refs=0)
- bool `is` (mask __m, char __c) const
- const char * `is` (const char * __lo, const char * __hi, mask * __vec) const
- const `char_type` * `narrow` (const `char_type` * __lo, const `char_type` * __hi, char __default, char * __to) const
- char `narrow` (`char_type` __c, char __default) const
- const char * `scan_is` (mask __m, const char * __lo, const char * __hi) const
- const char * `scan_not` (mask __m, const char * __lo, const char * __hi) const

- const mask * [table](#) () const throw ()
- const [char_type](#) * [tolower](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- [char_type](#) [tolower](#) ([char_type](#) __c) const
- [char_type](#) [toupper](#) ([char_type](#) __c) const
- const [char_type](#) * [toupper](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- const char * [widen](#) (const char *__lo, const char *__hi, [char_type](#) *__to) const
- [char_type](#) [widen](#) (char __c) const

Static Public Member Functions

- static const mask * [classic_table](#) () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t [table_size](#)
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual char [do_narrow](#) ([char_type](#) __c, char) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) *__lo, const [char_type](#) *__hi, char, char *__dest) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#)) const
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#)) const
- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) *__lo, const [char_type](#) *__hi) const
- virtual [char_type](#) [do_widen](#) (char __c) const
- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, [char_type](#) *__dest) const

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Protected Attributes

- `__c_locale _M_c_locale_ctype`
- `bool _M_del`
- `char _M_narrow [1+static_cast< unsigned char >(-1)]`
- `char _M_narrow_ok`
- `const mask * _M_table`
- `__to_type _M_tolower`
- `__to_type _M_toupper`
- `char _M_widen [1+static_cast< unsigned char >(-1)]`
- `char _M_widen_ok`

Friends

- `class locale::_Impl`

5.427.1 Detailed Description

`template<> class std::ctype_byname< char >`

22.2.1.4 Class [ctype_byname](#) specializations.

Definition at line 1481 of file `locale_facets.h`.

5.427.2 Member Typedef Documentation

5.427.2.1 `typedef char std::ctype< char >::char_type` [\[inherited\]](#)

Typedef for the template parameter `char`.

Definition at line 678 of file `locale_facets.h`.

5.427.3 Member Function Documentation

5.427.3.1 `static const mask* std::ctype< char >::classic_table () throw ()`
`[static, inherited]`

Returns a pointer to the C locale mask table.

5.427.3.2 `virtual char std::ctype< char >::do_narrow (char_type __c, char`
`)const [inline, protected, virtual, inherited]`

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

dfault Char to return if conversion fails.

Returns

The converted char.

Definition at line 1123 of file `locale_facets.h`.

5.427.3.3 `virtual const char_type* std::ctype< char >::do_narrow (const`
`char_type * __lo, const char_type * __hi, char, char * __dest)`
`const [inline, protected, virtual, inherited]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.
hi Pointer to end of range.
dfault Char to use if conversion fails.
to Pointer to the destination array.

Returns

hi.

Definition at line 1149 of file locale_facets.h.

5.427.3.4 `virtual const char_type* std::ctype< char >::do_tolower (char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.
hi Pointer to end of range.

Returns

hi.

5.427.3.5 `virtual char_type std::ctype< char >::do_tolower (char_type) const [protected, virtual, inherited]`

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The lowercase char if convertible, else *c*.

5.427.3.6 `virtual const char_type* std::ctype< char >::do_toupper (char_type * __lo, const char_type * __hi) const [protected, virtual, inherited]`

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

Returns

hi.

5.427.3.7 `virtual char_type std::ctype< char >::do_toupper (char_type) const [protected, virtual, inherited]`

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The uppercase char if convertible, else *c*.

5.427.3.8 **virtual char_type std::ctype< char >::do_widen (char __c) const** [inline, protected, virtual, inherited]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be returned unchanged.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted character.

Definition at line 1074 of file `locale_facets.h`.

5.427.3.9 **virtual const char* std::ctype< char >::do_widen (const char * __lo, const char * __hi, char_type * __dest) const** [inline, protected, virtual, inherited]

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived [ctype<char>](#) facet, the argument will be copied unchanged.

[do_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to start of range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 1097 of file `locale_facets.h`.

5.427.3.10 `const char * std::ctype< char >::is (const char * __lo, const char * __hi, mask * __vec) const [inline, inherited]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- vec* Pointer to an array of mask storage.

Returns

hi.

Definition at line 46 of file ctype_inline.h.

5.427.3.11 `bool std::ctype< char >::is (mask __m, char __c) const [inline, inherited]`

Test char classification.

This function compares the mask table[c] to *m*.

Parameters

- c* The char to compare the mask of.
- m* The mask to compare against.

Returns

True if *m* & table[c] is true, false otherwise.

Definition at line 41 of file ctype_inline.h.

5.427.3.12 `char std::ctype< char >::narrow (char_type __c, char __dfault) const [inline, inherited]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- c* The char to convert.
- dfault* Char to return if conversion fails.

Returns

The converted character.

Definition at line 922 of file `locale_facets.h`.

5.427.3.13 `const char_type* std::ctype<char>::narrow (const char_type *
__lo, const char_type * __hi, char __dfault, char * __to) const
[inline, inherited]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

- lo* Pointer to start of range.
- hi* Pointer to end of range.
- dfault* Char to use if conversion fails.
- to* Pointer to the destination array.

Returns

hi.

Definition at line 955 of file `locale_facets.h`.

5.427.3.14 `const char * std::ctype< char >::scan_is (mask __m, const char * __lo, const char * __hi) const [inline, inherited]`

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a matching char if found, else *hi*.

Definition at line 55 of file ctype_inline.h.

5.427.3.15 `const char * std::ctype< char >::scan_not (mask __m, const char * __lo, const char * __hi) const [inline, inherited]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [lo,hi) for which is(m,char) is false.

Parameters

m The mask to compare against.

lo Pointer to start of range.

hi Pointer to end of range.

Returns

Pointer to a non-matching char if found, else *hi*.

Definition at line 65 of file ctype_inline.h.

5.427.3.16 `const mask* std::ctype< char >::table () const throw () [inline, inherited]`

Returns a pointer to the mask table provided to the constructor, or /// the default from [classic_table\(\)](#) if none was provided.

Definition at line 973 of file locale_facets.h.

5.427.3.17 `char_type std::ctype< char >::tolower (char_type __c) const`
[inline, inherited]

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

[tolower\(\)](#) acts as if it returns `ctype<char>::do_tolower(c)`. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The lowercase char if convertible, else *c*.

Definition at line 827 of file locale_facets.h.

5.427.3.18 `const char_type* std::ctype< char >::tolower (char_type * __lo,`
`const char_type * __hi) const` **[inline, inherited]**

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

[tolower\(\)](#) acts as if it returns `ctype<char>::do_tolower(lo, hi)`. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

hi.

Definition at line 844 of file locale_facets.h.

5.427.3.19 `const char_type* std::ctype< char >::toupper (char_type * __lo,`
`const char_type * __hi) const` **[inline, inherited]**

Convert array to uppercase.

This function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

[toupper\(\)](#) acts as if it returns `ctype<char>::do_toupper(lo, hi)`. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

Returns

hi.

Definition at line 811 of file locale_facets.h.

5.427.3.20 char_type std::ctype< char >::toupper (char_type __c) const [inline, inherited]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[toupper\(\)](#) acts as if it returns `ctype<char>::do_toupper(c)`. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

c The char to convert.

Returns

The uppercase char if convertible, else *c*.

Definition at line 794 of file locale_facets.h.

5.427.3.21 char_type std::ctype< char >::widen (char __c) const [inline, inherited]

Widen char.

This function converts the char to char_type using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. [do_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

c The char to convert.

Returns

The converted character.

Definition at line 864 of file locale_facets.h.

5.427.3.22 `const char* std::ctype< char >::widen (const char * __lo, const char * __hi, char_type * __to) const [inline, inherited]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

lo Pointer to first char in range.

hi Pointer to end of range.

to Pointer to the destination array.

Returns

hi.

Definition at line 891 of file locale_facets.h.

5.427.4 Member Data Documentation

5.427.4.1 `locale::id std::ctype< char >::id [static, inherited]`

The facet id for `ctype<char>`

Definition at line 695 of file locale_facets.h.

5.427.4.2 `const size_t std::ctype< char >::table_size [static, inherited]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 697 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.428 `std::decay<_Tp>` Class Template Reference

`decay`

Public Types

- `typedef __decay_selector< __remove_type >::__type type`

5.428.1 Detailed Description

`template<typename _Tp> class std::decay<_Tp>`

`decay`

Definition at line 428 of file `type_traits`.

The documentation for this class was generated from the following file:

- [type_traits](#)

5.429 `std::decimal::decimal128` Class Reference

3.2.4 Class [decimal128](#).

Public Types

- `typedef float __decfloat128 __attribute__((mode(TD)))`

Public Member Functions

- `decimal128` ([decimal32](#) d32)
- `decimal128` (float __r)
- `decimal128` (unsigned int __z)
- `decimal128` (long __z)

- **decimal128** (double __r)
- **decimal128** (unsigned long __z)
- **decimal128** (long long __z)
- **decimal128** ([decimal64](#) d64)
- **decimal128** (long double __r)
- **decimal128** (unsigned long long __z)
- [decimal128](#) (__decfloat128 __z)
- **decimal128** (int __z)
- __decfloat128 **__getval** (void)
- void **__setval** (__decfloat128 __x)
- [decimal128](#) & **operator*=** ([decimal32](#) __rhs)
- [decimal128](#) & **operator*=** ([decimal64](#) __rhs)
- [decimal128](#) & **operator*=** ([decimal128](#) __rhs)
- [decimal128](#) & **operator*=** (int __rhs)
- [decimal128](#) & **operator*=** (unsigned int __rhs)
- [decimal128](#) & **operator*=** (long __rhs)
- [decimal128](#) & **operator*=** (unsigned long __rhs)
- [decimal128](#) & **operator*=** (unsigned long long __rhs)
- [decimal128](#) & **operator*=** (long long __rhs)
- [decimal128](#) & **operator++** ()
- [decimal128](#) **operator++** (int)
- [decimal128](#) & **operator+=** (int __rhs)
- [decimal128](#) & **operator+=** ([decimal32](#) __rhs)
- [decimal128](#) & **operator+=** ([decimal64](#) __rhs)
- [decimal128](#) & **operator+=** ([decimal128](#) __rhs)
- [decimal128](#) & **operator+=** (unsigned int __rhs)
- [decimal128](#) & **operator+=** (long __rhs)
- [decimal128](#) & **operator+=** (unsigned long __rhs)
- [decimal128](#) & **operator+=** (long long __rhs)
- [decimal128](#) & **operator+=** (unsigned long long __rhs)
- [decimal128](#) & **operator--** ()
- [decimal128](#) **operator--** (int)
- [decimal128](#) & **operator-=** (long long __rhs)
- [decimal128](#) & **operator-=** (long __rhs)
- [decimal128](#) & **operator-=** (int __rhs)
- [decimal128](#) & **operator-=** ([decimal32](#) __rhs)
- [decimal128](#) & **operator-=** (unsigned long __rhs)
- [decimal128](#) & **operator-=** ([decimal128](#) __rhs)
- [decimal128](#) & **operator-=** (unsigned long long __rhs)
- [decimal128](#) & **operator-=** (unsigned int __rhs)
- [decimal128](#) & **operator-=** ([decimal64](#) __rhs)
- [decimal128](#) & **operator/=** ([decimal64](#) __rhs)
- [decimal128](#) & **operator/=** (long __rhs)

- [decimal128](#) & **operator/=** (long long __rhs)
- [decimal128](#) & **operator/=** ([decimal32](#) __rhs)
- [decimal128](#) & **operator/=** (unsigned int __rhs)
- [decimal128](#) & **operator/=** (unsigned long __rhs)
- [decimal128](#) & **operator/=** (unsigned long long __rhs)
- [decimal128](#) & **operator/=** ([decimal128](#) __rhs)
- [decimal128](#) & **operator/=** (int __rhs)

5.429.1 Detailed Description

3.2.4 Class [decimal128](#).

Definition at line 391 of file decimal.

5.429.2 Constructor & Destructor Documentation

5.429.2.1 `std::decimal::decimal128::decimal128 (__decfloat128 __z)`
`[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 416 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

5.430 std::decimal::decimal32 Class Reference

3.2.2 Class [decimal32](#).

Public Types

- typedef float __decfloat32 **__attribute__** ((mode(SD)))

Public Member Functions

- **decimal32** ([decimal64](#) __d64)
- **decimal32** (float __r)
- **decimal32** (unsigned int __z)
- **decimal32** (long __z)

- **decimal32** (double __r)
- **decimal32** (unsigned long __z)
- **decimal32** (long long __z)
- **decimal32** ([decimal128](#) __d128)
- **decimal32** (long double __r)
- **decimal32** (unsigned long long __z)
- [decimal32](#) (__decfloat32 __z)
- **decimal32** (int __z)
- __decfloat32 **__getval** (void)
- void **__setval** (__decfloat32 __x)
- [decimal32](#) & **operator*=** ([decimal32](#) __rhs)
- [decimal32](#) & **operator*=** ([decimal64](#) __rhs)
- [decimal32](#) & **operator*=** ([decimal128](#) __rhs)
- [decimal32](#) & **operator*=** (int __rhs)
- [decimal32](#) & **operator*=** (unsigned int __rhs)
- [decimal32](#) & **operator*=** (long __rhs)
- [decimal32](#) & **operator*=** (unsigned long __rhs)
- [decimal32](#) & **operator*=** (unsigned long long __rhs)
- [decimal32](#) & **operator*=** (long long __rhs)
- [decimal32](#) & **operator++** ()
- [decimal32](#) **operator++** (int)
- [decimal32](#) & **operator+=** (int __rhs)
- [decimal32](#) & **operator+=** ([decimal32](#) __rhs)
- [decimal32](#) & **operator+=** ([decimal64](#) __rhs)
- [decimal32](#) & **operator+=** ([decimal128](#) __rhs)
- [decimal32](#) & **operator+=** (unsigned int __rhs)
- [decimal32](#) & **operator+=** (long __rhs)
- [decimal32](#) & **operator+=** (unsigned long __rhs)
- [decimal32](#) & **operator+=** (long long __rhs)
- [decimal32](#) & **operator+=** (unsigned long long __rhs)
- [decimal32](#) & **operator--** ()
- [decimal32](#) **operator--** (int)
- [decimal32](#) & **operator-=** (long long __rhs)
- [decimal32](#) & **operator-=** (long __rhs)
- [decimal32](#) & **operator-=** (int __rhs)
- [decimal32](#) & **operator-=** ([decimal32](#) __rhs)
- [decimal32](#) & **operator-=** (unsigned long __rhs)
- [decimal32](#) & **operator-=** ([decimal128](#) __rhs)
- [decimal32](#) & **operator-=** (unsigned long long __rhs)
- [decimal32](#) & **operator-=** (unsigned int __rhs)
- [decimal32](#) & **operator-=** ([decimal64](#) __rhs)
- [decimal32](#) & **operator/=** ([decimal64](#) __rhs)
- [decimal32](#) & **operator/=** (long __rhs)

- [decimal32](#) & **operator/=** (long long __rhs)
- [decimal32](#) & **operator/=** ([decimal32](#) __rhs)
- [decimal32](#) & **operator/=** (unsigned int __rhs)
- [decimal32](#) & **operator/=** (unsigned long __rhs)
- [decimal32](#) & **operator/=** (unsigned long long __rhs)
- [decimal32](#) & **operator/=** ([decimal128](#) __rhs)
- [decimal32](#) & **operator/=** (int __rhs)

5.430.1 Detailed Description

3.2.2 Class [decimal32](#).

Definition at line 225 of file decimal.

5.430.2 Constructor & Destructor Documentation

5.430.2.1 `std::decimal::decimal32::decimal32 (__decfloat32 __z)`
`[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 249 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

5.431 std::decimal::decimal64 Class Reference

3.2.3 Class [decimal64](#).

Public Types

- typedef float __decfloat64 **__attribute__** ((mode(DD)))

Public Member Functions

- **decimal64** ([decimal32](#) d32)
- **decimal64** (float __r)
- **decimal64** (unsigned int __z)
- **decimal64** (long __z)

- **decimal64** (double __r)
- **decimal64** (unsigned long __z)
- **decimal64** (long long __z)
- **decimal64** ([decimal128](#) d128)
- **decimal64** (long double __r)
- **decimal64** (unsigned long long __z)
- [decimal64](#) (__decfloat64 __z)
- **decimal64** (int __z)
- __decfloat64 **__getval** (void)
- void **__setval** (__decfloat64 __x)
- [decimal64](#) & **operator*=** ([decimal32](#) __rhs)
- [decimal64](#) & **operator*=** ([decimal64](#) __rhs)
- [decimal64](#) & **operator*=** ([decimal128](#) __rhs)
- [decimal64](#) & **operator*=** (int __rhs)
- [decimal64](#) & **operator*=** (unsigned int __rhs)
- [decimal64](#) & **operator*=** (long __rhs)
- [decimal64](#) & **operator*=** (unsigned long __rhs)
- [decimal64](#) & **operator*=** (unsigned long long __rhs)
- [decimal64](#) & **operator*=** (long long __rhs)
- [decimal64](#) & **operator++** ()
- [decimal64](#) **operator++** (int)
- [decimal64](#) & **operator+=** (int __rhs)
- [decimal64](#) & **operator+=** ([decimal32](#) __rhs)
- [decimal64](#) & **operator+=** ([decimal64](#) __rhs)
- [decimal64](#) & **operator+=** ([decimal128](#) __rhs)
- [decimal64](#) & **operator+=** (unsigned int __rhs)
- [decimal64](#) & **operator+=** (long __rhs)
- [decimal64](#) & **operator+=** (unsigned long __rhs)
- [decimal64](#) & **operator+=** (long long __rhs)
- [decimal64](#) & **operator+=** (unsigned long long __rhs)
- [decimal64](#) & **operator--** ()
- [decimal64](#) **operator--** (int)
- [decimal64](#) & **operator-=** (long long __rhs)
- [decimal64](#) & **operator-=** (long __rhs)
- [decimal64](#) & **operator-=** (int __rhs)
- [decimal64](#) & **operator-=** ([decimal32](#) __rhs)
- [decimal64](#) & **operator-=** (unsigned long __rhs)
- [decimal64](#) & **operator-=** ([decimal128](#) __rhs)
- [decimal64](#) & **operator-=** (unsigned long long __rhs)
- [decimal64](#) & **operator-=** (unsigned int __rhs)
- [decimal64](#) & **operator-=** ([decimal64](#) __rhs)
- [decimal64](#) & **operator/=** ([decimal64](#) __rhs)
- [decimal64](#) & **operator/=** (long __rhs)

- [decimal64](#) & `operator/=` (long long __rhs)
- [decimal64](#) & `operator/=` ([decimal32](#) __rhs)
- [decimal64](#) & `operator/=` (unsigned int __rhs)
- [decimal64](#) & `operator/=` (unsigned long __rhs)
- [decimal64](#) & `operator/=` (unsigned long long __rhs)
- [decimal64](#) & `operator/=` ([decimal128](#) __rhs)
- [decimal64](#) & `operator/=` (int __rhs)

5.431.1 Detailed Description

3.2.3 Class [decimal64](#).

Definition at line 308 of file decimal.

5.431.2 Constructor & Destructor Documentation

5.431.2.1 `std::decimal::decimal64::decimal64 (__decfloat64 __z)`
`[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 332 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

5.432 `std::default_delete<_Tp>` Struct Template Reference

Primary template, [default_delete](#).

Public Member Functions

- `template<typename _Up , typename = typename std::enable_if<std::is_convertible<_Up*, _Tp*>::value>::type>`
`default_delete (const default_delete<_Up> &)`
- `void operator() (_Tp *__ptr) const`

5.432.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp >`

Primary template, [default_delete](#).

Definition at line 48 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

5.433 `std::default_delete< _Tp[]>` Struct Template Reference

Specialization, [default_delete](#).

Public Member Functions

- `void operator() (_Tp *__ptr) const`

5.433.1 Detailed Description

`template<typename _Tp> struct std::default_delete< _Tp[] >`

Specialization, [default_delete](#).

Definition at line 69 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

5.434 `std::defer_lock_t` Struct Reference

Do not acquire ownership of the mutex.

5.434.1 Detailed Description

Do not acquire ownership of the mutex.

Definition at line 375 of file `mutex`.

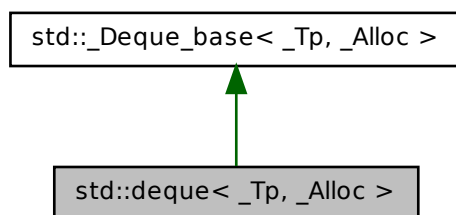
The documentation for this struct was generated from the following file:

- [mutex](#)

5.435 `std::deque< _Tp, _Alloc >` Class Template Reference

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Inheritance diagram for `std::deque< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `_Tp_alloc_type::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- [deque](#) ()
- [deque](#) (const allocator_type &__a)
- [deque](#) (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
[deque](#) (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- [deque](#) (const [deque](#) &__x)
- [deque](#) (size_type __n)
- [deque](#) ([deque](#) &&__x)
- [deque](#) (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- [~deque](#) ()
- void [assign](#) (size_type __n, const value_type &__val)
- template<typename _InputIterator >
void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) (initializer_list< value_type > __l)
- reference [at](#) (size_type __n)
- const_reference [at](#) (size_type __n) const
- reference [back](#) ()
- const_reference [back](#) () const
- iterator [begin](#) ()
- const_iterator [begin](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [cend](#) () const
- void [clear](#) ()
- const_reverse_iterator [crbegin](#) () const
- const_reverse_iterator [crend](#) () const
- template<typename... _Args>
[iterator](#) [emplace](#) (iterator __position, _Args &&...__args)
- template<typename... _Args>
void [emplace_back](#) (_Args &&...__args)
- template<typename... _Args>
void [emplace_front](#) (_Args &&...__args)
- bool [empty](#) () const
- iterator [end](#) ()
- const_iterator [end](#) () const
- iterator [erase](#) (iterator __position)
- iterator [erase](#) (iterator __first, iterator __last)
- const_reference [front](#) () const
- reference [front](#) ()

- allocator_type [get_allocator](#) () const
- iterator [insert](#) (iterator __position, value_type &&__x)
- void [insert](#) (iterator __p, initializer_list< value_type > __l)
- template<typename _InputIterator >
void [insert](#) (iterator __position, _InputIterator __first, _InputIterator __last)
- void [insert](#) (iterator __position, size_type __n, const value_type &__x)
- iterator [insert](#) (iterator __position, const value_type &__x)
- size_type [max_size](#) () const
- deque & [operator=](#) (initializer_list< value_type > __l)
- deque & [operator=](#) (const deque &__x)
- deque & [operator=](#) (deque &&__x)
- reference [operator\[\]](#) (size_type __n)
- const_reference [operator\[\]](#) (size_type __n) const
- void [pop_back](#) ()
- void [pop_front](#) ()
- void [push_back](#) (value_type &&__x)
- void [push_back](#) (const value_type &__x)
- void [push_front](#) (value_type &&__x)
- void [push_front](#) (const value_type &__x)
- reverse_iterator [rbegin](#) ()
- const_reverse_iterator [rbegin](#) () const
- reverse_iterator [rend](#) ()
- const_reverse_iterator [rend](#) () const
- void [resize](#) (size_type __new_size)
- void [resize](#) (size_type __new_size, const value_type &__x)
- void [shrink_to_fit](#) ()
- size_type [size](#) () const
- void [swap](#) (deque &__x)

Protected Types

- enum { **_S_initial_map_size** }
- typedef _Alloc::template rebind< _Tp * >::other **_Map_alloc_type**
- typedef pointer * **_Map_pointer**

Protected Member Functions

- _Tp ** **_M_allocate_map** (size_t __n)
- _Tp * **_M_allocate_node** ()
- template<typename _ForwardIterator >
void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last,
[std::forward_iterator_tag](#))

- `template<typename _InputIterator >`
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _Integer >`
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >`
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_create_nodes (_Tp **__nstart, _Tp **__nfinish)`
- `void _M_deallocate_map (_Tp **__p, size_t __n)`
- `void _M_deallocate_node (_Tp *__p)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize ()`
- `template<typename _Alloc1 >`
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator< _Tp > &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Tp **__nstart, _Tp **__nfinish)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Map_alloc_type _M_get_map_allocator () const`
- `_Tp_alloc_type & _M_get_Tp_allocator ()`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const`
- `template<typename _Integer >`
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_initialize_map (size_t)`
- `template<typename... _Args>`
`iterator _M_insert_aux (iterator __pos, _Args &&...__args)`
- `template<typename _ForwardIterator >`
`void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type &__x)`
- `template<typename _Integer >`
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`

- template<typename _InputIterator >
void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- void **_M_range_check** (size_type __n) const
- template<typename _ForwardIterator >
void **_M_range_insert_aux** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator >
void **_M_range_insert_aux** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator >
void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename... _Args>
void **_M_push_back_aux** (_Args &&...__args)
- template<typename... _Args>
void **_M_push_front_aux** (_Args &&...__args)
- void **_M_pop_back_aux** ()
- void **_M_pop_front_aux** ()
- iterator **_M_reserve_elements_at_front** (size_type __n)
- iterator **_M_reserve_elements_at_back** (size_type __n)
- void **_M_new_elements_at_front** (size_type __new_elements)
- void **_M_new_elements_at_back** (size_type __new_elements)
- void **_M_reserve_map_at_back** (size_type __nodes_to_add=1)
- void **_M_reserve_map_at_front** (size_type __nodes_to_add=1)
- void **_M_reallocate_map** (size_type __nodes_to_add, bool __add_at_front)

Static Protected Member Functions

- static size_t **_S_buffer_size** ()

Protected Attributes

- **_Deque_impl** **_M_impl**

5.435.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class
std::deque<_Tp, _Alloc>
```

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a deque<Tp> manages memory. Each deque has 4 members:

- Tp** _M_map
- size_t _M_map_size
- iterator _M_start, _M_finish

map_size is at least 8. map is an array of map_size pointers-to-. (The name map has nothing to do with the [std::map](#) class, and **nodes** should not be confused with [std::list](#)'s usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-Tp. If Tp is very large, there will be one Tp element per node (i.e., an *array* of one). For non-huge Tp's, node size is inversely related to Tp size: the larger the Tp, the fewer Tp's will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different Tp's, to improve allocator efficiency.

Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator i:
 - i.node points to a member of the map array. (Yes, you read that correctly: i.node does not actually point to a node.) The member of the map array is what actually points to the node.
 - i.first == *(i.node) (This points to the node (first Tp element).)
 - i.last == i.first + node_size

- i.cur is a pointer in the range [i.first, i.last). NOTE: the implication of this is that i.cur is always a dereferenceable pointer, even if i is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map_size).
- A pointer in the range [map, map + map_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, _Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 717 of file stl_deque.h.

5.435.2 Constructor & Destructor Documentation

5.435.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque () [inline]`

Default constructor creates no elements.

Definition at line 769 of file stl_deque.h.

5.435.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (const allocator_type & __a)
[inline, explicit]`

Creates a deque with no elements.

Parameters

a An allocator object.

Definition at line 777 of file `stl_deque.h`.

5.435.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (size_type __n) [inline,
explicit]`

Creates a deque with default constructed elements.

Parameters

n The number of elements to initially create.

This constructor fills the deque with *n* default constructed elements.

Definition at line 789 of file `stl_deque.h`.

5.435.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (size_type __n, const value_type
& __value, const allocator_type & __a = allocator_type())
[inline]`

Creates a deque with copies of an exemplar element.

Parameters

n The number of elements to initially create.

value An element to copy.

a An allocator.

This constructor fills the deque with *n* copies of *value*.

Definition at line 801 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::_M_fill_initialize()`.

5.435.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::deque (const deque<_Tp, _Alloc> &
__x) [inline]`

Deque copy constructor.

Parameters

x A deque of identical element and allocator types.

The newly-created deque uses a copy of the allocation object used by *x*.

Definition at line 828 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin(), and std::deque< _Tp, _Alloc >::end().

5.435.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (deque< _Tp, _Alloc > && __x
) [inline]`

Deque move constructor.

Parameters

x A deque of identical element and allocator types.

The newly-created deque contains the exact contents of *x*. The contents of *x* are a valid, but unspecified deque.

Definition at line 842 of file stl_deque.h.

5.435.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (initializer_list< value_type
> __l, const allocator_type & __a = allocator_type())
[inline]`

Builds a deque from an initializer list.

Parameters

l An [initializer_list](#).

a An allocator object.

Create a deque consisting of copies of the elements in the [initializer_list](#) *l*.

This will call the element type's copy constructor *N* times (where *N* is *l.size()*) and do no memory reallocation.

Definition at line 856 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::M_range_initialize().

```

5.435.2.8  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
            template<typename _InputIterator > std::deque< _Tp, _Alloc
            >::deque ( _InputIterator __first, _InputIterator __last, const
            allocator_type & __a = allocator_type() ) [inline]

```

Builds a deque from a range.

Parameters

first An input iterator.

last An input iterator.

a An allocator object.

Create a deque consisting of copies of the elements from [first, last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 881 of file stl_deque.h.

```

5.435.2.9  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
            std::deque< _Tp, _Alloc >::~~deque ( ) [inline]

```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 895 of file stl_deque.h.

References `std::deque< _Tp, _Alloc >::begin()`, and `std::deque< _Tp, _Alloc >::end()`.

5.435.3 Member Function Documentation

```

5.435.3.1  template<typename _Tp , typename _Alloc > void
            deque::_M_fill_initialize ( const value_type & __value )
            [protected]

```

Fills the deque with copies of value.

Parameters

value Initial value.

Returns

Nothing.

Precondition

_M_start and _M_finish have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

Definition at line 331 of file deque.tcc.

References std::_Destroy().

Referenced by std::deque< _Tp, _Alloc >::deque().

5.435.3.2 `template<typename _Tp , typename _Alloc > void
std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (size_t
__num_elements) [protected, inherited]`

Layout storage.

Parameters

num_elements The count of T's for which to allocate space at first.

Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 572 of file stl_deque.h.

References std::max().

Referenced by std::deque< _Tp, _Alloc >::_M_range_initialize().

5.435.3.3 `template<typename _Tp , typename _Alloc > void
deque::_M_new_elements_at_back (size_type __new_elements)
[protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 825 of file deque.tcc.

References std::deque< _Tp, _Alloc >::_M_reserve_map_at_back(), std::deque< _Tp, _Alloc >::max_size(), and std::deque< _Tp, _Alloc >::size().

Referenced by std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back().

5.435.3.4 `template<typename _Tp , typename _Alloc > void
deque::_M_new_elements_at_front (size_type __new_elements)
[protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 800 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::_M_reserve_map_at_front()`, `std::deque< _Tp, _Alloc >::max_size()`, and `std::deque< _Tp, _Alloc >::size()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front()`.

5.435.3.5 `template<typename _Tp , typename _Alloc > void
deque::_M_pop_back_aux () [protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 481 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::pop_back()`.

5.435.3.6 `template<typename _Tp , typename _Alloc > void
deque::_M_pop_front_aux () [protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 496 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::pop_front()`.

5.435.3.7 `template<typename _Tp , typename _Alloc > template<typename...
_Args> void deque::_M_push_back_aux (_Args &&... __args)
[protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 415 of file deque.tcc.

Referenced by `std::deque< _Tp, _Alloc >::push_back()`.

5.435.3.8 `template<typename _Tp , typename _Alloc > template<typename...
_Args> void deque::_M_push_front_aux (_Args &&... __args)
[protected]`

Helper functions for `push_*` and `pop_*`.

Definition at line 449 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::push_front().

5.435.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_range_check (size_type __n)
const [inline, protected]`

Safety check used only from [at\(\)](#).

Definition at line 1235 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::size().

Referenced by std::deque< _Tp, _Alloc >::at().

5.435.3.10 `template<typename _Tp , typename _Alloc > template<typename
_InputIterator > void deque::_M_range_initialize (_InputIterator
__first, _InputIterator __last, std::input_iterator_tag)
[protected]`

Fills the deque with whatever is in [first,last).

Parameters

first An input iterator.

last An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.

Definition at line 357 of file deque.tcc.

References std::_Deque_base< _Tp, _Alloc >::_M_initialize_map(), std::deque< _Tp, _Alloc >::clear(), and std::deque< _Tp, _Alloc >::push_back().

Referenced by std::deque< _Tp, _Alloc >::deque().

5.435.3.11 `template<typename _Tp , typename _Alloc > template<typename
_ForwardIterator > void deque::_M_range_initialize (
_ForwardIterator __first, _ForwardIterator __last,
std::forward_iterator_tag) [protected]`

Fills the deque with whatever is in [first,last).

Parameters

first An input iterator.

last An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 377 of file `deque.tcc`.

References `std::_Destroy()`, `std::_Deque_base< _Tp, _Alloc >::_M_initialize_map()`, `std::advance()`, and `std::distance()`.

5.435.3.12 `template<typename _Tp, typename _Alloc > void
deque::_M_reallocate_map (size_type __nodes_to_add, bool
__add_at_front) [protected]`

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 850 of file `deque.tcc`.

References `std::max()`.

Referenced by `std::deque< _Tp, _Alloc >::_M_reserve_map_at_back()`, and `std::deque< _Tp, _Alloc >::_M_reserve_map_at_front()`.

5.435.3.13 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back
(size_type __n) [inline, protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1856 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_new_elements_at_back()`.

5.435.3.14 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front
(size_type __n) [inline, protected]`

Memory-handling helpers for the previous internal insert functions.

Definition at line 1846 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_new_elements_at_front().

5.435.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (
size_type __nodes_to_add = 1) [inline, protected]`

Memory-handling helpers for the major map.

Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)

Definition at line 1882 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_reallocate_map().

Referenced by std::deque< _Tp, _Alloc >::_M_new_elements_at_back().

5.435.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (
size_type __nodes_to_add = 1) [inline, protected]`

Memory-handling helpers for the major map.

Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)

Definition at line 1890 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_reallocate_map().

Referenced by std::deque< _Tp, _Alloc >::_M_new_elements_at_front().

5.435.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (size_type __n, const
value_type & __val) [inline]`

Assigns a given value to a deque.

Parameters

n Number of elements to be assigned.

val Value to be assigned.

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 956 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::operator=()`.

```
5.435.3.18  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               template<typename _InputIterator > void std::deque<_Tp, _Alloc
               >::assign ( _InputIterator __first, _InputIterator __last )
               [inline]
```

Assigns a range to a deque.

Parameters

first An input iterator.

last An input iterator.

This function fills a deque with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 973 of file `stl_deque.h`.

```
5.435.3.19  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               void std::deque<_Tp, _Alloc>::assign ( initializer_list<
               value_type > __l ) [inline]
```

Assigns an initializer list to a deque.

Parameters

l An [initializer_list](#).

This function fills a deque with copies of the elements in the [initializer_list](#) *l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 992 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::assign()`.

Referenced by `std::deque<_Tp, _Alloc>::assign()`.

```
5.435.3.20  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               reference std::deque<_Tp, _Alloc>::at ( size_type __n )
               [inline]
```

Provides access to the data contained in the deque.

Parameters

n The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws *out_of_range* if the check fails.

Definition at line 1254 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_range_check().

```
5.435.3.21  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               const_reference std::deque< _Tp, _Alloc >::at ( size_type __n )
               const [inline]
```

Provides access to the data contained in the deque.

Parameters

n The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws *out_of_range* if the check fails.

Definition at line 1272 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_range_check().

```
5.435.3.22  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
               const_reference std::deque< _Tp, _Alloc >::back ( ) const
               [inline]
```

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1311 of file stl_deque.h.

References `std::deque< _Tp, _Alloc >::end()`.

5.435.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::back () [inline]`

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1299 of file stl_deque.h.

References `std::deque< _Tp, _Alloc >::end()`.

5.435.3.24 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::begin () [inline]`

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1007 of file stl_deque.h.

Referenced by `std::deque< _Tp, _Alloc >::clear()`, `std::deque< _Tp, _Alloc >::deque()`, `std::deque< _Tp, _Alloc >::erase()`, `std::deque< _Tp, _Alloc >::front()`, `std::deque< _Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::deque< _Tp, _Alloc >::~~deque()`.

5.435.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::begin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1015 of file stl_deque.h.

5.435.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cbegin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1078 of file stl_deque.h.

5.435.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cend () const
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1087 of file stl_deque.h.

5.435.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::clear () [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1577 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin().

Referenced by std::deque< _Tp, _Alloc >::_M_range_initialize(), std::deque< _Tp, _Alloc >::erase(), and std::deque< _Tp, _Alloc >::operator=().

5.435.3.29 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crbegin ()
const [inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1096 of file stl_deque.h.

5.435.3.30 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1105 of file stl_deque.h.

5.435.3.31 `template<typename _Tp, typename _Alloc > template<typename...
_Args> deque< _Tp, _Alloc >::iterator deque::emplace (iterator
__position, _Args &&... __args)`

Inserts an object in deque before specified iterator.

Parameters

position An iterator into the deque.

args Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location.`

Definition at line 172 of file `deque.tcc`.

References `std::deque<_Tp, _Alloc >::push_back()`, and `std::deque<_Tp, _Alloc >::push_front()`.

Referenced by `std::deque<_Tp, _Alloc >::insert()`.

5.435.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::deque<_Tp, _Alloc >::empty () const [inline]`

Returns true if the deque is empty. (Thus `begin()` would equal `end()`.)

Definition at line 1198 of file `stl_deque.h`.

5.435.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque<_Tp, _Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1024 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc >::back()`, `std::deque<_Tp, _Alloc >::deque()`, `std::deque<_Tp, _Alloc >::erase()`, `std::deque<_Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::deque<_Tp, _Alloc >::~~deque()`.

5.435.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque<_Tp, _Alloc >::end () const
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1033 of file `stl_deque.h`.

5.435.3.35 `template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase (iterator __position)`

Remove element at given position.

Parameters

position Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 194 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::end()`, `std::deque< _Tp, _Alloc >::pop_back()`, `std::deque< _Tp, _Alloc >::pop_front()`, and `std::deque< _Tp, _Alloc >::size()`.

5.435.3.36 `template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase (iterator __first, iterator __last)`

Remove a range of elements.

Parameters

first Iterator pointing to the first element to be erased.

last Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 217 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::begin()`, `std::deque< _Tp, _Alloc >::clear()`, `std::deque< _Tp, _Alloc >::end()`, and `std::deque< _Tp, _Alloc >::size()`.

5.435.3.37 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc>::front () [inline]`

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1283 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`.

5.435.3.38 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc>::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1291 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`.

5.435.3.39 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::deque<_Tp, _Alloc>::get_allocator () const
[inline]`

Get a copy of the memory allocation object.

Reimplemented from `std::_Deque_base<_Tp, _Alloc>`.

Definition at line 998 of file `stl_deque.h`.

5.435.3.40 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::deque<_Tp,
_Alloc>::insert (iterator __position, _InputIterator __first,
_InputIterator __last) [inline]`

Inserts a range into the deque.

Parameters

position An iterator into the deque.

first An input iterator.

last An input iterator.

This function will insert copies of the data in the range `[first,last)` into the deque before the location specified by *pos*. This is known as *range insert*.

Definition at line 1504 of file `stl_deque.h`.

5.435.3.41 `template<typename _Tp, typename _Alloc > deque< _Tp, _Alloc >::iterator deque::insert (iterator __position, const value_type & __x)`

Inserts given value into deque before specified iterator.

Parameters

position An iterator into the deque.

x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Definition at line 149 of file deque.tcc.

References `std::deque< _Tp, _Alloc >::push_back()`, and `std::deque< _Tp, _Alloc >::push_front()`.

Referenced by `std::deque< _Tp, _Alloc >::operator=()`, and `std::deque< _Tp, _Alloc >::resize()`.

5.435.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::insert (iterator __p, initializer_list< value_type > __l) [inline]`

Inserts an initializer list into the deque.

Parameters

p An iterator into the deque.

l An [initializer_list](#).

This function will insert copies of the data in the [initializer_list](#) *l* into the deque before the location specified by *p*. This is known as *list insert*.

Definition at line 1475 of file stl_deque.h.

References `std::deque< _Tp, _Alloc >::insert()`.

Referenced by `std::deque< _Tp, _Alloc >::insert()`.

5.435.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::insert (iterator __position, value_type && __x) [inline]`

Inserts given rvalue into deque before specified iterator.

Parameters

position An iterator into the deque.
x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1462 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::emplace().

```
5.435.3.44  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::deque< _Tp, _Alloc >::insert ( iterator __position,
             size_type __n, const value_type & __x ) [inline]
```

Inserts a number of copies of given data into the deque.

Parameters

position An iterator into the deque.
n Number of elements to be inserted.
x Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Definition at line 1489 of file stl_deque.h.

```
5.435.3.45  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             size_type std::deque< _Tp, _Alloc >::max_size ( ) const
             [inline]
```

Returns the [size\(\)](#) of the largest possible deque.

Definition at line 1117 of file stl_deque.h.

Referenced by std::deque< _Tp, _Alloc >::M_new_elements_at_back(), and std::deque< _Tp, _Alloc >::M_new_elements_at_front().

```
5.435.3.46  template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc
             > & deque::operator= ( const deque< _Tp, _Alloc > & __x )
```

Deque assignment operator.

Parameters

x A deque of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 93 of file deque.tcc.

References std::deque< _Tp, _Alloc >::begin(), std::deque< _Tp, _Alloc >::end(), std::deque< _Tp, _Alloc >::insert(), and std::deque< _Tp, _Alloc >::size().

5.435.3.47 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque< _Tp, _Alloc >::operator= (initializer_list<
value_type > __l) [inline]`

Assigns an initializer list to a deque.

Parameters

l An [initializer_list](#).

This function fills a deque with copies of the elements in the [initializer_list](#) *l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 938 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::assign().

5.435.3.48 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque< _Tp, _Alloc >::operator= (deque< _Tp,
_Alloc > && __x) [inline]`

Deque move assignment operator.

Parameters

x A deque of identical element and allocator types.

The contents of *x* are moved into this deque (without copying). *x* is a valid, but unspecified deque.

Definition at line 917 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::clear(), and std::deque< _Tp, _Alloc >::swap().

5.435.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque<_Tp, _Alloc>::operator[] (size_type __n)
[inline]`

Subscript access to the data contained in the deque.

Parameters

n The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 1214 of file `std_deque.h`.

5.435.3.50 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque<_Tp, _Alloc>::operator[] (size_type
__n) const [inline]`

Subscript access to the data contained in the deque.

Parameters

n The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 1229 of file `std_deque.h`.

5.435.3.51 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::pop_back () [inline]`

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before [pop_back\(\)](#) is called.

Definition at line 1412 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_pop_back_aux().

Referenced by std::deque< _Tp, _Alloc >::erase().

5.435.3.52 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_front () [inline]`

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before [pop_front\(\)](#) is called.

Definition at line 1391 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_pop_front_aux().

Referenced by std::deque< _Tp, _Alloc >::erase().

5.435.3.53 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_back (const value_type &
__x) [inline]`

Add data to the end of the deque.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1360 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_push_back_aux().

Referenced by std::deque< _Tp, _Alloc >::_M_range_initialize(), std::deque< _Tp, _Alloc >::emplace(), and std::deque< _Tp, _Alloc >::insert().

5.435.3.54 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_front (const value_type &
__x) [inline]`

Add data to the front of the deque.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1329 of file `stl_deque.h`.

References `std::deque< _Tp, _Alloc >::_M_push_front_aux()`.

Referenced by `std::deque< _Tp, _Alloc >::emplace()`, and `std::deque< _Tp, _Alloc >::insert()`.

5.435.3.55 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1051 of file `stl_deque.h`.

5.435.3.56 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rbegin ()
[inline]`

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1042 of file `stl_deque.h`.

5.435.3.57 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1060 of file `stl_deque.h`.

5.435.3.58 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1069 of file stl_deque.h.

```
5.435.3.59  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::deque< _Tp, _Alloc >::resize ( size_type __new_size )
             [inline]
```

Resizes the deque to the specified number of elements.

Parameters

new_size Number of elements the deque should contain.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1131 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::size().

```
5.435.3.60  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::deque< _Tp, _Alloc >::resize ( size_type __new_size,
             const value_type & __x ) [inline]
```

Resizes the deque to the specified number of elements.

Parameters

new_size Number of elements the deque should contain.

x Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1153 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::insert(), and std::deque< _Tp, _Alloc >::size().

```
5.435.3.61  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::deque< _Tp, _Alloc >::shrink_to_fit ( ) [inline]
```

A non-binding request to reduce memory use.

Definition at line 1189 of file stl_deque.h.

5.435.3.62 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque<_Tp, _Alloc>::size () const [inline]`

Returns the number of elements in the deque.

Definition at line 1112 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::_M_new_elements_at_back()`, `std::deque<_Tp, _Alloc>::_M_new_elements_at_front()`, `std::deque<_Tp, _Alloc>::_M_range_check()`, `std::deque<_Tp, _Alloc>::erase()`, `std::deque<_Tp, _Alloc>::operator=()`, `std::operator==()`, and `std::deque<_Tp, _Alloc>::resize()`.

5.435.3.63 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::swap (deque<_Tp, _Alloc> &
__x) [inline]`

Swaps data with another deque.

Parameters

x A deque of the same element and allocator types.

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Definition at line 1557 of file `stl_deque.h`.

Referenced by `std::deque<_Tp, _Alloc>::operator=()`, and `std::swap()`.

The documentation for this class was generated from the following files:

- [stl_deque.h](#)
- [deque.tcc](#)

5.436 `std::discard_block_engine<
RandomNumberEngine, __p, __r > Class
Template Reference`

Public Types

- `typedef RandomNumberEngine::result_type` [result_type](#)

Public Member Functions

- [discard_block_engine](#) ()
- [discard_block_engine](#) (const `_RandomNumberEngine` &__rne)
- [discard_block_engine](#) (result_type __s)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>`
[discard_block_engine](#) (_Sseq &__q)
- [discard_block_engine](#) (`_RandomNumberEngine` &&__rne)
- const `_RandomNumberEngine` & [base](#) () const
- void [discard](#) (unsigned long long __z)
- [result_type](#) [max](#) () const
- [result_type](#) [min](#) () const
- [result_type](#) [operator\(\)](#) ()
- void [seed](#) (result_type __s)
- `template<typename _Sseq >`
 void [seed](#) (_Sseq &__q)
- void [seed](#) ()

Static Public Attributes

- static const size_t **block_size**
- static const size_t **used_block**

Friends

- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits >`
[std::basic_ostream](#)<_CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)<_CharT, _Traits > &, const [std::discard_block_engine](#)<_RandomNumberEngine1, __p1, __r1 > &)
- bool [operator==](#) (const [discard_block_engine](#) &__lhs, const [discard_block_engine](#) &__rhs)
- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits >`
[std::basic_istream](#)<_CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)<_CharT, _Traits > &, [std::discard_block_engine](#)<_RandomNumberEngine1, __p1, __r1 > &)

5.436.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r> class
std::discard_block_engine< _RandomNumberEngine, __p, __r >
```

Produces random numbers from some base engine by discarding blocks of data.

0 <= __r <= __p

Definition at line 787 of file random.h.

5.436.2 Member Typedef Documentation

5.436.2.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> typedef _RandomNumberEngine::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::result_type`

The type of the generated random value.

Definition at line 794 of file random.h.

5.436.3 Constructor & Destructor Documentation

5.436.3.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine () [inline]`

Constructs a default discard_block_engine engine.

The underlying engine is default constructed as well.

Definition at line 805 of file random.h.

5.436.3.2 `template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (const _RandomNumberEngine & __rne) [inline, explicit]`

Copy constructs a discard_block_engine engine.

Copies an existing base class random number generator.

Parameters

rne An existing (base class) engine object.

Definition at line 815 of file random.h.

5.436.3.3 `template<typename _RandomNumberEngine, size_t __p, size_t
 __r> std::discard_block_engine<_RandomNumberEngine, __p, __r
 >::discard_block_engine (_RandomNumberEngine && __rne)
 [inline, explicit]`

Move constructs a discard_block_engine engine.

Copies an existing base class random number generator.

Parameters

rng An existing (base class) engine object.

Definition at line 825 of file random.h.

5.436.3.4 `template<typename _RandomNumberEngine, size_t __p, size_t
 __r> std::discard_block_engine<_RandomNumberEngine, __p,
 __r>::discard_block_engine (result_type __s) [inline,
 explicit]`

Seed constructs a discard_block_engine engine.

Constructs the underlying generator engine seeded with __s.

Parameters

__s A seed value for the base class engine.

Definition at line 835 of file random.h.

5.436.3.5 `template<typename _RandomNumberEngine, size_t __p,
 size_t __r> template<typename _Sseq, typename = typename
 std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value
 && !std::is_same<_Sseq, _RandomNumberEngine>::value>
 ::type> std::discard_block_engine<_RandomNumberEngine,
 __p, __r>::discard_block_engine (_Sseq & __q) [inline,
 explicit]`

Generator construct a discard_block_engine engine.

Parameters

__q A seed sequence.

Definition at line 848 of file random.h.

5.436.4 Member Function Documentation

5.436.4.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> const _RandomNumberEngine& std::discard_block_engine<_RandomNumberEngine, __p, __r>::base () const [inline]`

Gets a const reference to the underlying generator engine object.

Definition at line 892 of file random.h.

5.436.4.2 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard (unsigned long long __z) [inline]`

Discard a sequence of random numbers.

Todo

Look for a faster way to do discard.

Definition at line 919 of file random.h.

5.436.4.3 `template<typename _RandomNumberEngine, size_t __p, size_t __r> result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::max () const [inline]`

Gets the maximum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 910 of file random.h.

5.436.4.4 `template<typename _RandomNumberEngine, size_t __p, size_t __r> result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::min () const [inline]`

Gets the minimum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 901 of file random.h.

5.436.4.5 `template<typename _RandomNumberEngine, size_t __p, size_t
__r> discard_block_engine< _RandomNumberEngine, __p, __r
>::result_type std::discard_block_engine< _RandomNumberEngine,
__p, __r >::operator() ()`

Gets the next value in the generated random number sequence.

Definition at line 665 of file random.tcc.

5.436.4.6 `template<typename _RandomNumberEngine, size_t __p, size_t
__r> template<typename _Sseq > void std::discard_block_engine<
_RandomNumberEngine, __p, __r >::seed (_Sseq & __q)
[inline]`

Reseeds the discard_block_engine object with the given seed sequence.

Parameters

`__q` A seed generator function.

Definition at line 881 of file random.h.

5.436.4.7 `template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r
>::seed (result_type __s) [inline]`

Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

Definition at line 868 of file random.h.

5.436.4.8 `template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r
>::seed () [inline]`

Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

Definition at line 857 of file random.h.

5.436.5 Friends And Related Function Documentation

5.436.5.1 `template<typename _RandomNumberEngine, size_t __p,
size_t __r> template<typename _RandomNumberEngine1 ,
size_t __p1, size_t __r1, typename _CharT , typename _Traits
> std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 >
&) [friend]`

Inserts the current state of a discard_block_engine random number generator engine `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A discard_block_engine random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.436.5.2 `template<typename _RandomNumberEngine, size_t __p,
size_t __r> bool operator==(const discard_block_engine<
_RandomNumberEngine, __p, __r > & __lhs, const
discard_block_engine< _RandomNumberEngine, __p, __r > &
__rhs) [friend]`

Compares two discard_block_engine random number generator objects of the same type for equality.

Parameters

`__lhs` A discard_block_engine random number generator object.

`__rhs` Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 943 of file random.h.

5.436.5.3 `template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> &, std::discard_block_engine<_RandomNumberEngine1, __p1, __r1> &) [friend]`

Extracts the current state of a % `subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `discard_block_engine` random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.437 `std::discrete_distribution< _IntType >` Class Template Reference

A [discrete_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- `template<typename _InputIterator > discrete_distribution (_InputIterator __wbegin, _InputIterator __wend)`

- `template<typename _Func >`
discrete_distribution (size_t __nw, double __xmin, double __xmax, _Func __fw)
- **discrete_distribution** (const [param_type](#) &__p)
- **discrete_distribution** ([initializer_list](#)< double > __wl)
- [result_type](#) max () const
- [result_type](#) min () const
- `template<typename _UniformRandomNumberGenerator >`
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- `template<typename _UniformRandomNumberGenerator >`
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- `std::vector< double >` [probabilities](#) () const
- void [reset](#) ()

Friends

- `template<typename _IntType1 , typename _CharT , typename _Traits >`
[std::basic_ostream](#)< _CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::discrete_distribution](#)< _IntType1 > &)
- `template<typename _IntType1 , typename _CharT , typename _Traits >`
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &, [std::discrete_distribution](#)< _IntType1 > &)

5.437.1 Detailed Description

`template<typename _IntType = int> class std::discrete_distribution< _IntType >`

A [discrete_distribution](#) random number distribution. The formula for the discrete probability mass function is

Definition at line 4685 of file random.h.

5.437.2 Member Typedef Documentation

5.437.2.1 `template<typename _IntType = int> typedef _IntType std::discrete_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 4692 of file random.h.

5.437.3 Member Function Documentation

5.437.3.1 `template<typename _IntType = int> result_type
std::discrete_distribution< _IntType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4798 of file random.h.

References std::vector< _Tp, _Alloc >::size().

5.437.3.2 `template<typename _IntType = int> result_type
std::discrete_distribution< _IntType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4791 of file random.h.

5.437.3.3 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::discrete_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4806 of file random.h.

References std::discrete_distribution< _IntType >::operator()(), and std::discrete_distribution< _IntType >::param().

Referenced by std::discrete_distribution< _IntType >::operator()().

5.437.3.4 `template<typename _IntType = int> void std::discrete_distribution<
_IntType >::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 4784 of file random.h.

5.437.3.5 `template<typename _IntType = int> param_type
std::discrete_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4776 of file random.h.

Referenced by `std::discrete_distribution< _IntType >::operator()()`, and `std::operator==()`.

5.437.3.6 `template<typename _IntType = int> std::vector<double>
std::discrete_distribution< _IntType >::probabilities () const
[inline]`

Returns the probabilities of the distribution.

Definition at line 4769 of file random.h.

5.437.3.7 `template<typename _IntType = int> void std::discrete_distribution<
_IntType >::reset () [inline]`

Resets the distribution state.

Definition at line 4762 of file random.h.

5.437.4 Friends And Related Function Documentation

5.437.4.1 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_ostream< _CharT,
_Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & ,
const std::discrete_distribution< _IntType1 > &) [friend]`

Inserts a `discrete_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `discrete_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.438 `std::discrete_distribution< _IntType >::param_type` Struct Reference 2425

5.437.4.2 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> (std::basic_istream< _CharT, _Traits > & ,
std::discrete_distribution< _IntType1 > &) [friend]`

Extracts a `discrete_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `discrete_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.438 `std::discrete_distribution< _IntType >::param_type` Struct Reference

Public Types

- typedef [discrete_distribution](#)< `_IntType` > `distribution_type`

Public Member Functions

- `template<typename _InputIterator >`
`param_type` (`_InputIterator __wbegin`, `_InputIterator __wend`)
- `template<typename _Func >`
`param_type` (`size_t __nw`, `double __xmin`, `double __xmax`, `_Func __fw`)
- `param_type` ([initializer_list](#)< `double` > `__wil`)
- `std::vector`< `double` > `probabilities` () const

Friends

- class `discrete_distribution`< `_IntType` >
- bool `operator==` (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.438.1 Detailed Description

template<typename _IntType = int> struct std::discrete_distribution< _IntType >::param_type

Parameter type.

Definition at line 4694 of file random.h.

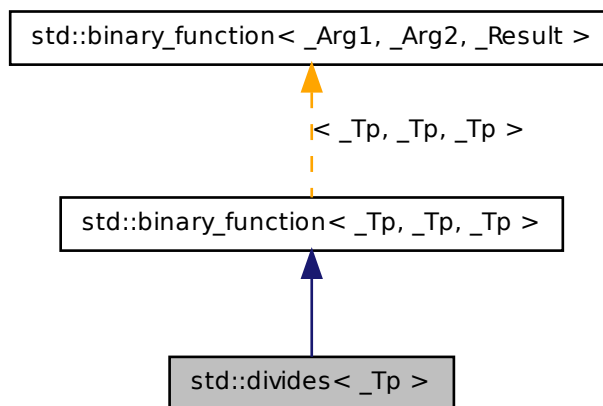
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.439 std::divides< _Tp > Struct Template Reference

One of the [math functions](#).

Inheritance diagram for std::divides< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)

- typedef _Tp [result_type](#)
- typedef _Tp [second_argument_type](#)

Public Member Functions

- _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

5.439.1 Detailed Description

template<typename _Tp> struct std::divides< _Tp >

One of the [math functors](#).

Definition at line 162 of file stl_function.h.

5.439.2 Member Typedef Documentation

5.439.2.1 **typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::first_argument_type** [**inherited**]

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.439.2.2 **typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type**
 [**inherited**]

type of the return type

Definition at line 118 of file stl_function.h.

5.439.2.3 **typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::second_argument_type** [**inherited**]

the type of the second argument

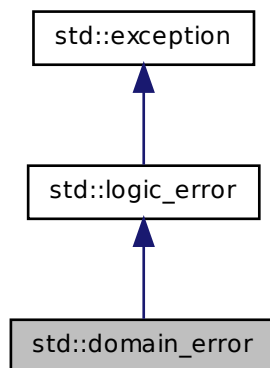
Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.440 std::domain_error Class Reference

Inheritance diagram for std::domain_error:



Public Member Functions

- **domain_error** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.440.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 73 of file stdexcept.

5.440.2 Member Function Documentation

5.440.2.1 virtual const char* std::logic_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

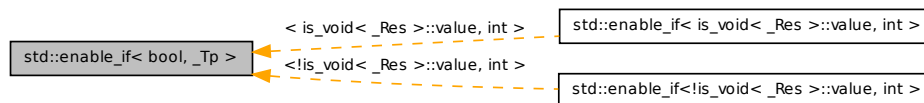
The documentation for this class was generated from the following file:

- [stdexcept](#)

5.441 `std::enable_if< bool, _Tp >` Struct Template Reference

[enable_if](#)

Inheritance diagram for `std::enable_if< bool, _Tp >`:



5.441.1 Detailed Description

```
template<bool, typename _Tp = void> struct std::enable_if< bool, _Tp >
```

[enable_if](#)

Definition at line 384 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.442 `std::enable_shared_from_this< _Tp >` Class Template Reference

Base class allowing use of member function `shared_from_this`.

Public Member Functions

- [shared_ptr](#)< _Tp > **shared_from_this** ()
- [shared_ptr](#)< const _Tp > **shared_from_this** () const

Protected Member Functions

- **enable_shared_from_this** (const [enable_shared_from_this](#) &)
- [enable_shared_from_this](#) & **operator=** (const [enable_shared_from_this](#) &)

Friends

- `template<typename _Tp1 >
void __enable_shared_from_this_helper (const __shared_count<> &__pn,
const enable_shared_from_this *__pe, const _Tp1 *__px)`

5.442.1 Detailed Description

`template<typename _Tp> class std::enable_shared_from_this< _Tp >`

Base class allowing use of member function `shared_from_this`.

Definition at line 466 of file `shared_ptr.h`.

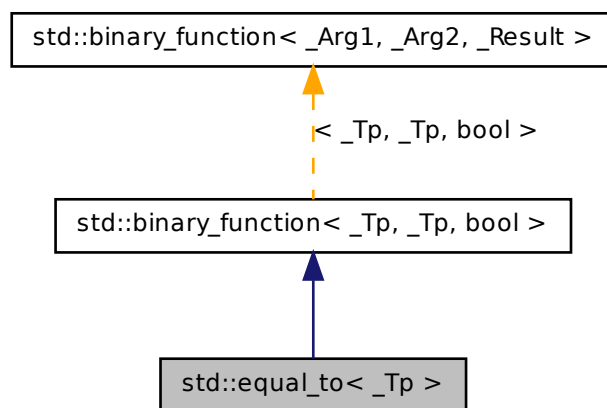
The documentation for this class was generated from the following file:

- [shared_ptr.h](#)

5.443 `std::equal_to< _Tp >` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::equal_to< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.443.1 Detailed Description

`template<typename _Tp> struct std::equal_to< _Tp >`

One of the [comparison functors](#).

Definition at line 199 of file `stl_function.h`.

5.443.2 Member Typedef Documentation

5.443.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.443.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.443.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.444 std::error_category Class Reference

[error_category](#)

Public Member Functions

- **error_category** (const [error_category](#) &)
- virtual [error_condition](#) **default_error_condition** (int __i) const
- virtual bool **equivalent** (const [error_code](#) &__code, int __i) const
- virtual bool **equivalent** (int __i, const [error_condition](#) &__cond) const
- virtual [string](#) **message** (int) const =0
- virtual const char * **name** () const =0
- bool **operator!=** (const [error_category](#) &__other) const
- bool **operator<** (const [error_category](#) &__other) const
- [error_category](#) & **operator=** (const [error_category](#) &)
- bool **operator==** (const [error_category](#) &__other) const

5.444.1 Detailed Description

[error_category](#)

Definition at line 64 of file `system_error`.

The documentation for this class was generated from the following file:

- [system_error](#)

5.445 std::error_code Struct Reference

[error_code](#)

Public Member Functions

- **error_code** (int __v, const [error_category](#) &__cat)
- template<typename _ErrorCodeEnum >
 error_code (_ErrorCodeEnum __e, typename [enable_if](#)< [is_error_code_enum](#)< _ErrorCodeEnum >::value >::type * = 0)
- void **assign** (int __v, const [error_category](#) &__cat)
- const [error_category](#) & **category** () const
- void **clear** ()
- [error_condition](#) **default_error_condition** () const
- [string](#) **message** () const
- **operator bool** () const
- template<typename _ErrorCodeEnum >
 [enable_if](#)< [is_error_code_enum](#)< _ErrorCodeEnum >::value, [error_code](#) & >::type **operator=** (_ErrorCodeEnum __e)
- int **value** () const

Friends

- class **hash**< [error_code](#) >

5.445.1 Detailed Description

[error_code](#)

Definition at line 116 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system_error](#)

5.446 std::error_condition Struct Reference

[error_condition](#)

Public Member Functions

- **error_condition** (int __v, const [error_category](#) &__cat)
- template<typename _ErrorConditionEnum >
error_condition (_ErrorConditionEnum __e, typename [enable_if](#)< [is_error_condition_enum](#)< _ErrorConditionEnum >::value >::type !=0)
- void **assign** (int __v, const [error_category](#) &__cat)
- const [error_category](#) & **category** () const
- void **clear** ()
- [string](#) **message** () const
- **operator bool** () const
- template<typename _ErrorConditionEnum >
[enable_if](#)< [is_error_condition_enum](#)< _ErrorConditionEnum >::value, [error_condition](#) & >::type **operator=** (_ErrorConditionEnum __e)
- int **value** () const

5.446.1 Detailed Description

[error_condition](#)

Definition at line 193 of file system_error.

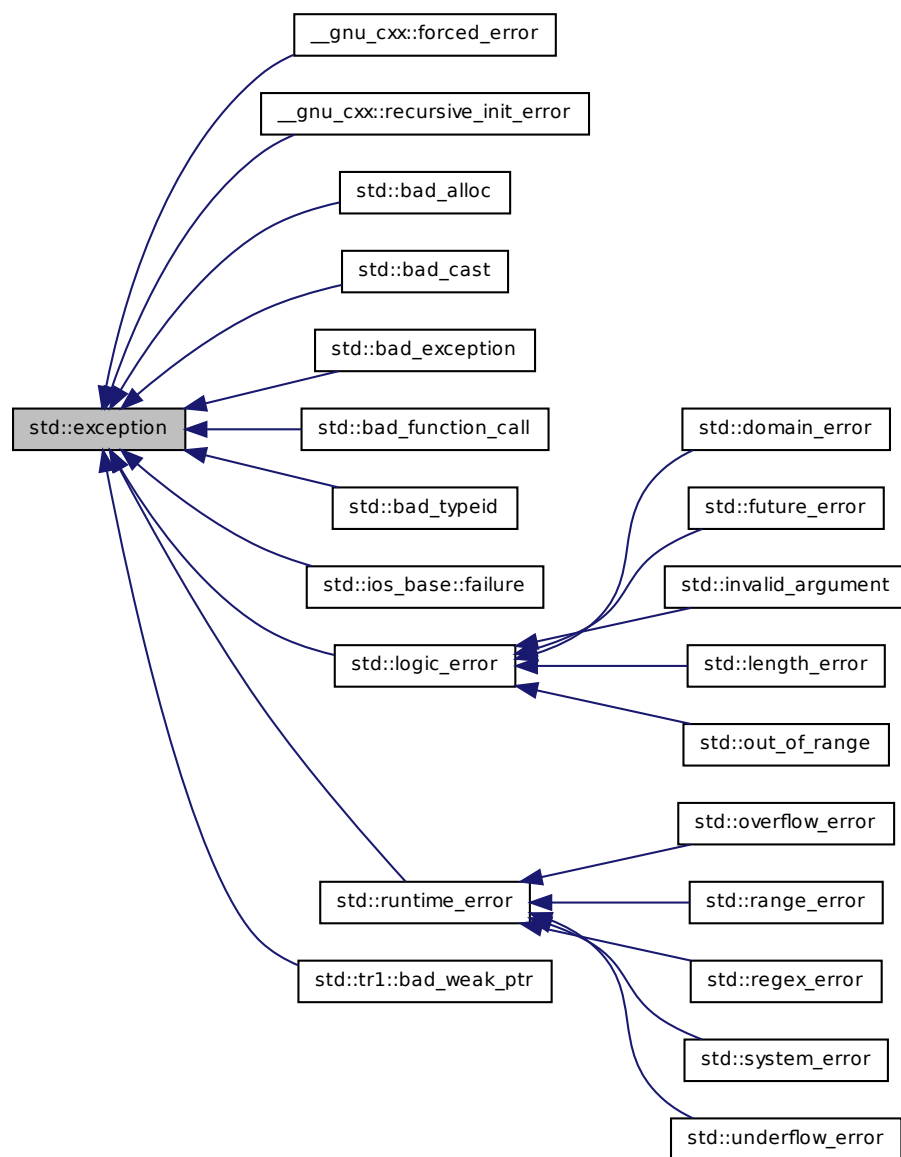
The documentation for this struct was generated from the following file:

- [system_error](#)

5.447 std::exception Class Reference

Base class for all library exceptions.

Inheritance diagram for std::exception:



Public Member Functions

- virtual const char * [what](#) () const throw ()

5.447.1 Detailed Description

Base class for all library exceptions. This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

Definition at line 61 of file exception.

5.447.2 Member Function Documentation

5.447.2.1 virtual const char* std::exception::what () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in [std::bad_exception](#), [std::bad_alloc](#), [std::bad_cast](#), [std::bad_typeid](#), [std::future_error](#), [std::logic_error](#), [std::runtime_error](#), [std::tr1::bad_weak_ptr](#), and [std::ios_base::failure](#).

The documentation for this class was generated from the following file:

- [exception](#)

5.448 std::exponential_distribution< _RealType > Class Template Reference

An exponential continuous distribution for random numbers.

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- `exponential_distribution` (const `result_type` &__lambda=`result_type`(1))
- `exponential_distribution` (const `param_type` &__p)
- `_RealType lambda` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (_UniformRandomNumberGenerator &__urng)
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (_UniformRandomNumberGenerator &__urng, const `param_type` &__p)
- `param_type param` () const
- `void param` (const `param_type` &__param)
- `void reset` ()

5.448.1 Detailed Description

`template<typename _RealType = double> class std::exponential_distribution<_RealType>`

An exponential continuous distribution for random numbers. The formula for the exponential probability density function is $p(x|\lambda) = \lambda e^{-\lambda x}$.

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	<i>zero</i>
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda^2}$

Table 5.1: Distribution Statistics

Definition at line 4164 of file `random.h`.

5.448.2 Member Typedef Documentation

5.448.2.1 `template<typename _RealType = double> typedef _RealType std::exponential_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4171 of file `random.h`.

5.448.3 Constructor & Destructor Documentation

5.448.3.1 `template<typename _RealType = double> std::exponential_distribution< _RealType >::exponential_distribution (const result_type & __lambda = result_type (1)) [inline, explicit]`

Constructs an exponential distribution with inverse scale parameter λ .

Definition at line 4202 of file random.h.

5.448.4 Member Function Documentation

5.448.4.1 `template<typename _RealType = double> _RealType std::exponential_distribution< _RealType >::lambda () const [inline]`

Returns the inverse scale parameter of the distribution.

Definition at line 4223 of file random.h.

5.448.4.2 `template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4252 of file random.h.

5.448.4.3 `template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4245 of file random.h.

5.448.4.4 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::exponential_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4260 of file random.h.

5.448 `std::exponential_distribution<_RealType>` Class Template Reference 2439

References `std::exponential_distribution<_RealType>::operator()()`, and `std::exponential_distribution<_RealType>::param()`.

Referenced by `std::exponential_distribution<_RealType>::operator()()`.

5.448.4.5 `template<typename _RealType = double> param_type
std::exponential_distribution<_RealType>::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4230 of file `random.h`.

Referenced by `std::exponential_distribution<_RealType>::operator()()`, `std::operator==()`, and `std::operator>>()`.

5.448.4.6 `template<typename _RealType = double> void
std::exponential_distribution<_RealType>::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 4238 of file `random.h`.

5.448.4.7 `template<typename _RealType = double> void
std::exponential_distribution<_RealType>::reset () [inline]`

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4217 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

5.449 `std::exponential_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [exponential_distribution<_RealType>](#) `distribution_type`

Public Member Functions

- `param_type` (`_RealType __lambda=_RealType(1)`)
- `_RealType lambda` () const

Friends

- bool `operator==` (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.449.1 Detailed Description

`template<typename _RealType = double> struct std::exponential_distribution<_RealType>::param_type`

Parameter type.

Definition at line 4173 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.450 `std::extreme_value_distribution<_RealType>` Class Template Reference

A [extreme_value_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` `result_type`

Public Member Functions

- **`extreme_value_distribution`** (`_RealType` `__a`=`_RealType(0)`, `_RealType` `__b`=`_RealType(1)`)
- **`extreme_value_distribution`** (const `param_type` &`__p`)
- `_RealType` `a` () const
- `_RealType` `b` () const
- `result_type` `max` () const
- `result_type` `min` () const
- template<typename `_UniformRandomNumberGenerator` >
`result_type` **`operator()`** (`_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- template<typename `_UniformRandomNumberGenerator` >
`result_type` **`operator()`** (`_UniformRandomNumberGenerator` &`__urng`)
- void `param` (const `param_type` &`__param`)
- `param_type` `param` () const
- void `reset` ()

5.450.1 Detailed Description

template<typename `_RealType` = `double`> class `std::extreme_value_distribution<_RealType>`

A `extreme_value_distribution` random number distribution. The formula for the normal probability mass function is

$$p(x|a, b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Definition at line 4513 of file `random.h`.

5.450.2 Member Typedef Documentation

5.450.2.1 **template<typename `_RealType` = `double`> typedef `_RealType` `std::extreme_value_distribution<_RealType>::result_type`**

The type of the range of the distribution.

Definition at line 4520 of file `random.h`.

5.450.3 Member Function Documentation

5.450.3.1 `template<typename _RealType = double> _RealType
std::extreme_value_distribution< _RealType >::a () const
[inline]`

Return the a parameter of the distribution.

Definition at line 4571 of file random.h.

5.450.3.2 `template<typename _RealType = double> _RealType
std::extreme_value_distribution< _RealType >::b () const
[inline]`

Return the b parameter of the distribution.

Definition at line 4578 of file random.h.

5.450.3.3 `template<typename _RealType = double> result_type
std::extreme_value_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4607 of file random.h.

5.450.3.4 `template<typename _RealType = double> result_type
std::extreme_value_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4600 of file random.h.

5.450.3.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::extreme_value_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4615 of file random.h.

References `std::extreme_value_distribution< _RealType >::operator()()`, and `std::extreme_value_distribution< _RealType >::param()`.

5.451 `std::extreme_value_distribution<_RealType>::param_type` Struct Reference 2443

Referenced by `std::extreme_value_distribution<_RealType>::operator()`.

5.450.3.6 `template<typename _RealType = double> void
std::extreme_value_distribution<_RealType>::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 4593 of file `random.h`.

5.450.3.7 `template<typename _RealType = double> param_type
std::extreme_value_distribution<_RealType>::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4585 of file `random.h`.

Referenced by `std::extreme_value_distribution<_RealType>::operator()`, `std::operator==()`, and `std::operator>>()`.

5.450.3.8 `template<typename _RealType = double> void
std::extreme_value_distribution<_RealType>::reset ()
[inline]`

Resets the distribution state.

Definition at line 4564 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.451 `std::extreme_value_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `extreme_value_distribution<_RealType>` `distribution_type`

Public Member Functions

- **param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.451.1 Detailed Description

template<typename _RealType = double> struct std::extreme_value_distribution< _RealType >::param_type

Parameter type.

Definition at line 4522 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.452 std::fisher_f_distribution< _RealType > Class Template Reference

A [fisher_f_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **fisher_f_distribution** (_RealType __m=_RealType(1), _RealType __n=_RealType(1))
- **fisher_f_distribution** (const [param_type](#) &__p)

- `_RealType m () const`
- `result_type max () const`
- `result_type min () const`
- `_RealType n () const`
- `template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng, const
param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `void param (const param_type &__param)`
- `param_type param () const`
- `void reset ()`

Friends

- `template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _-
CharT, _Traits > &, const std::fisher_f_distribution< _RealType1 > &)`
- `template<typename _RealType1 >
bool operator== (const std::fisher_f_distribution< _RealType1 > &__d1, const
std::fisher_f_distribution< _RealType1 > &__d2)`
- `template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _-
CharT, _Traits > &, std::fisher_f_distribution< _RealType1 > &)`

5.452.1 Detailed Description

`template<typename _RealType = double> class std::fisher_f_distribution< _-
RealType >`

A `fisher_f_distribution` random number distribution. The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 2880 of file `random.h`.

5.452.2 Member Typedef Documentation

5.452.2.1 `template<typename _RealType = double> typedef _RealType
std::fisher_f_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2887 of file random.h.

5.452.3 Member Function Documentation

5.452.3.1 `template<typename _RealType = double> result_type
std::fisher_f_distribution<_RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2974 of file random.h.

5.452.3.2 `template<typename _RealType = double> result_type
std::fisher_f_distribution<_RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2967 of file random.h.

5.452.3.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::fisher_f_distribution<_RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2982 of file random.h.

5.452.3.4 `template<typename _RealType = double> void
std::fisher_f_distribution<_RealType >::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 2960 of file random.h.

5.452.3.5 `template<typename _RealType = double> param_type
std::fisher_f_distribution<_RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2952 of file random.h.

5.452.3.6 `template<typename _RealType = double> void
std::fisher_f_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2931 of file random.h.

References std::gamma_distribution< _RealType >::reset().

5.452.4 Friends And Related Function Documentation

5.452.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::fisher_f_distribution< _RealType1 > &) [friend]`

Inserts a fisher_f_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A fisher_f_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.452.4.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::fisher_f_distribution<
_RealType1 > & __d1, const std::fisher_f_distribution< _RealType1
> & __d2) [friend]`

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3003 of file random.h.

5.452.4.3 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream< _CharT, _Traits > & ,
std::fisher_f_distribution< _RealType1 > &) [friend]`

Extracts a fisher_f_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A fisher_f_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

5.453 `std::fisher_f_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef [fisher_f_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m=_RealType(1), _RealType __n=_RealType(1))
- _RealType **m** () const
- _RealType **n** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.453.1 Detailed Description

```
template<typename _RealType = double> struct std::fisher_f_distribution< _-  
_RealType >::param_type
```

Parameter type.

Definition at line 2889 of file random.h.

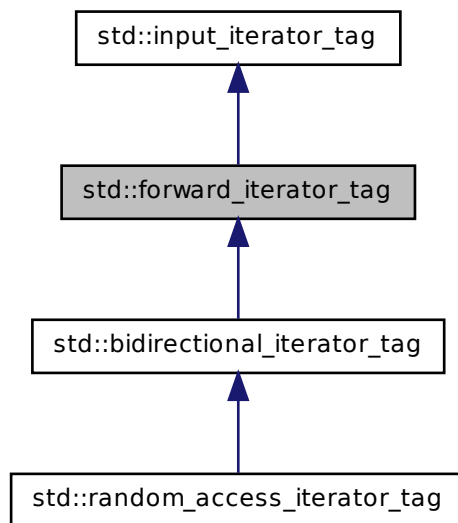
The documentation for this struct was generated from the following file:

- [random.h](#)

5.454 std::forward_iterator_tag Struct Reference

Forward iterators support a superset of input iterator operations.

Inheritance diagram for std::forward_iterator_tag:



5.454.1 Detailed Description

Forward iterators support a superset of input iterator operations.

Definition at line 94 of file `stl_iterator_base_types.h`.

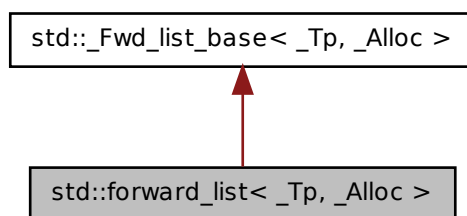
The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.455 `std::forward_list< _Tp, _Alloc >` Class Template Reference

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Inheritance diagram for `std::forward_list< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Fwd_list_const_iterator< _Tp >` **const_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `_Tp_alloc_type::const_reference` **const_reference**
- typedef `std::ptrdiff_t` **difference_type**
- typedef `_Fwd_list_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**

- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [forward_list](#) (const _Alloc &__al=_Alloc())
- [forward_list](#) (const [forward_list](#) &__list, const _Alloc &__al)
- [forward_list](#) (size_type __n)
- [forward_list](#) ([forward_list](#) &&__list)
- [forward_list](#) (std::initializer_list< _Tp > __il, const _Alloc &__al=_Alloc())
- [forward_list](#) (size_type __n, const _Tp &__value, const _Alloc &__al=_Alloc())
- [forward_list](#) ([forward_list](#) &&__list, const _Alloc &__al)
- template<typename _InputIterator >
[forward_list](#) (_InputIterator __first, _InputIterator __last, const _Alloc &__al=_Alloc())
- [forward_list](#) (const [forward_list](#) &__list)
- [~forward_list](#) ()
- void [assign](#) (std::initializer_list< _Tp > __il)
- template<typename _InputIterator >
void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) (size_type __n, const _Tp &__val)
- [iterator before_begin](#) ()
- [const_iterator before_begin](#) () const
- [iterator begin](#) ()
- [const_iterator begin](#) () const
- [const_iterator cbefore_begin](#) () const
- [const_iterator cbegin](#) () const
- [const_iterator cend](#) () const
- void [clear](#) ()
- template<typename... _Args>
[iterator emplace_after](#) (const_iterator __pos, _Args &&...__args)
- template<typename... _Args>
void [emplace_front](#) (_Args &&...__args)
- bool [empty](#) () const
- [const_iterator end](#) () const
- [iterator end](#) ()
- void [erase_after](#) (const_iterator __pos)
- void [erase_after](#) (const_iterator __pos, const_iterator __last)
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const

- `template<typename _InputIterator >`
`iterator insert_after (const_iterator __pos, _InputIterator __first, _InputIterator`
`__last)`
- `iterator insert_after (const_iterator __pos, const _Tp &&__val)`
- `iterator insert_after (const_iterator __pos, const _Tp &&__val)`
- `iterator insert_after (const_iterator __pos, size_type __n, const _Tp &&__val)`
- `iterator insert_after (const_iterator __pos, std::initializer_list< _Tp > __il)`
- `size_type max_size () const`
- `template<typename _Comp >`
`void merge (forward_list &&__list, _Comp __comp)`
- `void merge (forward_list &&__list)`
- `forward_list & operator= (std::initializer_list< _Tp > __il)`
- `forward_list & operator= (const forward_list &&__list)`
- `forward_list & operator= (forward_list &&__list)`
- `void pop_front ()`
- `void push_front (const _Tp &&__val)`
- `void push_front (_Tp &&__val)`
- `void remove (const _Tp &&__val)`
- `template<typename _Pred >`
`void remove_if (_Pred __pred)`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const value_type &&__val)`
- `void reverse ()`
- `template<typename _Comp >`
`void sort (_Comp __comp)`
- `void sort ()`
- `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __-`
`__before, const_iterator __last)`
- `void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __-`
`__i)`
- `void splice_after (const_iterator __pos, forward_list &&__list)`
- `void swap (forward_list &&__list)`
- `template<typename _BinPred >`
`void unique (_BinPred __binary_pred)`
- `void unique ()`

Private Types

- `typedef _Alloc::template rebind< _Fwd_list_node< _Tp > >::other _Node_-`
`alloc_type`

Private Member Functions

- template<typename... _Args>
_Node * _M_create_node (_Args &&...__args)
- void _M_erase_after (_Fwd_list_node_base *__pos)
- void _M_erase_after (_Fwd_list_node_base *__pos, _Fwd_list_node_base *___last)
- _Node * _M_get_node ()
- const _Node_alloc_type & _M_get_Node_allocator () const
- _Node_alloc_type & _M_get_Node_allocator ()
- template<typename... _Args>
_Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&...__args)
- void _M_put_node (_Node *__p)

Private Attributes

- _Fwd_list_impl _M_impl

5.455.1 Detailed Description

template<typename _Tp, typename _Alloc = allocator<_Tp>> class std::forward_list<_Tp, _Alloc>

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence. Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike [std::vector](#) and [std::deque](#), random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, [std::forward_list](#) provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `forward_list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `Fwd_list_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `forward_list<X, Alloc1>` are spliced into `forward_list<X, Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Definition at line 407 of file `forward_list.h`.

5.455.2 Constructor & Destructor Documentation

5.455.2.1 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (const _Alloc & __al
= _Alloc()) [inline, explicit]`

Creates a forward_list with no elements.

Parameters

al An allocator object.

Definition at line 436 of file forward_list.h.

5.455.2.2 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (const forward_list<
_Tp, _Alloc> & __list, const _Alloc & __al) [inline]`

Copy constructor with allocator argument.

Parameters

list Input list to copy.

al An allocator object.

Definition at line 445 of file forward_list.h.

5.455.2.3 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (forward_list<_Tp,
_Alloc> && __list, const _Alloc & __al) [inline]`

Move constructor with allocator argument.

Parameters

list Input list to move.

al An allocator object.

Definition at line 454 of file forward_list.h.

5.455.2.4 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (size_type __n)
[inline, explicit]`

Creates a forward_list with default constructed elements.

Parameters

n The number of elements to initially create.

This constructor creates the forward_list with *n* default constructed elements.

Definition at line 466 of file forward_list.h.

```
5.455.2.5  template<typename _Tp, typename _Alloc = allocator<_Tp>>
            std::forward_list< _Tp, _Alloc >::forward_list ( size_type __n,
            const _Tp & __value, const _Alloc & __al = _Alloc() )
            [inline]
```

Creates a forward_list with copies of an exemplar element.

Parameters

n The number of elements to initially create.

value An element to copy.

al An allocator object.

This constructor fills the forward_list with *n* copies of *value*.

Definition at line 479 of file forward_list.h.

```
5.455.2.6  template<typename _Tp, typename _Alloc = allocator<_Tp>>
            template<typename _InputIterator > std::forward_list< _Tp, _Alloc
            >::forward_list ( _InputIterator __first, _InputIterator __last,
            const _Alloc & __al = _Alloc() ) [inline]
```

Builds a forward_list from a range.

Parameters

first An input iterator.

last An input iterator.

al An allocator object.

Create a forward_list consisting of copies of the elements from [*first,last*). This is linear in N (where N is distance(*first,last*)).

Definition at line 495 of file forward_list.h.

5.455.2.7 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (const forward_list<
_Tp, _Alloc> & __list) [inline]`

The forward_list copy constructor.

Parameters

list A forward_list of identical element and allocator types.

The newly-created forward_list uses a copy of the allocation object used by *list*.

Definition at line 512 of file forward_list.h.

References std::forward_list<_Tp, _Alloc>::begin(), and std::forward_list<_Tp, _Alloc>::end().

5.455.2.8 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (forward_list<_Tp,
_Alloc> && __list) [inline]`

The forward_list move constructor.

Parameters

list A forward_list of identical element and allocator types.

The newly-created forward_list contains the exact contents of *forward_list*. The contents of *list* are a valid, but unspecified forward_list.

Definition at line 525 of file forward_list.h.

5.455.2.9 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (std::initializer_list<
_Tp> __il, const _Alloc & __al = _Alloc()) [inline]`

Builds a forward_list from an [initializer_list](#).

Parameters

il An [initializer_list](#) of value_type.

al An allocator object.

Create a forward_list consisting of copies of the elements in the [initializer_list](#) *il*. This is linear in il.size().

Definition at line 536 of file forward_list.h.

5.455.2.10 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::~~forward_list () [inline]`

The `forward_list` dtor.

Definition at line 544 of file `forward_list.h`.

5.455.3 Member Function Documentation

5.455.3.1 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator > void std::forward_list< _Tp,
_Alloc >::assign (_InputIterator __first, _InputIterator __last)
[inline]`

Assigns a range to a `forward_list`.

Parameters

first An input iterator.

last An input iterator.

This function fills a `forward_list` with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the `forward_list` and that the resulting `forward_list`'s size is the same as the number of elements assigned. Old data may be lost.

Definition at line 606 of file `forward_list.h`.

5.455.3.2 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::assign (size_type __n, const _Tp
& __val) [inline]`

Assigns a given value to a `forward_list`.

Parameters

n Number of elements to be assigned.

val Value to be assigned.

This function fills a `forward_list` with *n* copies of the given value. Note that the assignment completely changes the `forward_list` and that the resulting `forward_list`'s size is the same as the number of elements assigned. Old data may be lost.

Definition at line 623 of file `forward_list.h`.

5.455.3.3 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list<_Tp, _Alloc>::assign (std::initializer_list<_Tp
> __il) [inline]`

Assigns an [initializer_list](#) to a `forward_list`.

Parameters

il An [initializer_list](#) of `value_type`.

Replace the contents of the `forward_list` with copies of the elements in the [initializer_list](#) *il*. This is linear in `il.size()`.

Definition at line 638 of file `forward_list.h`.

5.455.3.4 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::before_begin ()
[inline]`

Returns a read/write iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 656 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::operator=()`, and `std::forward_list<_Tp, _Alloc>::resize()`.

5.455.3.5 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list<_Tp, _Alloc>::before_begin ()
const [inline]`

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 665 of file `forward_list.h`.

5.455.3.6 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::begin () [inline]`

Returns a read/write iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 673 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::forward_list()`, `std::forward_list<_Tp, _Alloc>::operator=()`, and `std::forward_list<_Tp, _Alloc>::unique()`.

5.455.3.7 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::begin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 682 of file forward_list.h.

5.455.3.8 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin ()
const [inline]`

Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 718 of file forward_list.h.

5.455.3.9 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbegin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 709 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=(), and std::operator==().

5.455.3.10 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cend () const
[inline]`

Returns a read-only (constant) iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 727 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=(), and std::operator==().

5.455.3.11 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::clear () [inline]`

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves

are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1016 of file forward_list.h.

```
5.455.3.12  template<typename _Tp, typename _Alloc = allocator<_Tp>>
               template<typename... _Args> iterator std::forward_list<_Tp,
               _Alloc>::emplace_after( const_iterator __pos, _Args &&...
               __args ) [inline]
```

Constructs object in forward_list after the specified iterator.

Parameters

pos A const_iterator into the forward_list.

args Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 840 of file forward_list.h.

```
5.455.3.13  template<typename _Tp, typename _Alloc = allocator<_Tp>>
               template<typename... _Args> void std::forward_list<_Tp, _Alloc
               >::emplace_front( _Args &&... __args ) [inline]
```

Constructs object in forward_list at the front of the list.

Parameters

args Arguments.

This function will insert an object of type Tp constructed with Tp(std::forward<Args>(args)...) at the front of the list. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 784 of file forward_list.h.

5.455.3.14 `template<typename _Tp, typename _Alloc = allocator<_Tp>> bool
std::forward_list< _Tp, _Alloc >::empty () const [inline]`

Returns true if the forward_list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 735 of file forward_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::insert_after()`.

5.455.3.15 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 691 of file forward_list.h.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::forward_list< _Tp, _Alloc >::resize()`, and `std::forward_list< _Tp, _Alloc >::unique()`.

5.455.3.16 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::end () const
[inline]`

Returns a read-only iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 700 of file forward_list.h.

5.455.3.17 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::erase_after (const_iterator
__pos, const_iterator __last) [inline]`

Remove a range of elements.

Parameters

pos Iterator pointing before the first element to be erased.

last Iterator pointing to one past the last element to be erased.

This function will erase the elements in the range (pos,last) and shorten the forward_list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only

erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 960 of file forward_list.h.

5.455.3.18 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list<_Tp, _Alloc>::erase_after (const_iterator
__pos) [inline]`

Removes the element pointed to by the iterator following `pos`.

Parameters

pos Iterator pointing before element to be erased.

This function will erase the element at the given position and thus shorten the forward_list by one.

Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 939 of file forward_list.h.

Referenced by `std::forward_list<_Tp, _Alloc>::operator=()`, `std::forward_list<_Tp, _Alloc>::resize()`, and `std::forward_list<_Tp, _Alloc>::unique()`.

5.455.3.19 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
reference std::forward_list<_Tp, _Alloc>::front () [inline]`

Returns a read/write reference to the data at the first element of the forward_list.

Definition at line 752 of file forward_list.h.

5.455.3.20 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_reference std::forward_list<_Tp, _Alloc>::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the forward_list.

Definition at line 763 of file forward_list.h.

5.455.3.21 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
allocator_type std::forward_list< _Tp, _Alloc >::get_allocator ()
const [inline]`

Get a copy of the memory allocation object.

Definition at line 646 of file forward_list.h.

5.455.3.22 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
const_iterator __pos, const _Tp & __val) [inline]`

Inserts given value into forward_list after specified iterator.

Parameters

pos An iterator into the forward_list.

val Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 857 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=(), and std::forward_list< _Tp, _Alloc >::resize().

5.455.3.23 `template<typename _Tp, typename _Alloc > forward_list< _Tp,
_Alloc >::iterator std::forward_list< _Tp, _Alloc >::insert_after (
const_iterator __pos, size_type __n, const _Tp & __val)`

Inserts a number of copies of given data into the forward_list.

Parameters

pos An iterator into the forward_list.

n Number of elements to be inserted.

val Data to be inserted.

Returns

An iterator pointing to the last inserted copy of *val* or *pos* if *n* == 0.

This function will insert a specified number of copies of the given data after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 244 of file forward_list.tcc.

5.455.3.24 `template<typename _Tp, typename _Alloc > template<typename
_InputIterator > forward_list< _Tp, _Alloc >::iterator
std::forward_list< _Tp, _Alloc >::insert_after (const_iterator
__pos, _InputIterator __first, _InputIterator __last)`

Inserts a range into the forward_list.

Parameters

position An iterator into the forward_list.

first An input iterator.

last An input iterator.

Returns

An iterator pointing to the last inserted element or *pos* if *first* == *last*.

This function will insert copies of the data in the range [*first*,*last*) into the forward_list after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 259 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::empty().

5.455.3.25 `template<typename _Tp, typename _Alloc > forward_list< _Tp,
_Alloc >::iterator std::forward_list< _Tp, _Alloc >::insert_after (
const_iterator __pos, std::initializer_list< _Tp > __il)`

Inserts the contents of an [initializer_list](#) into forward_list after the specified iterator.

Parameters

pos An iterator into the forward_list.

il An [initializer_list](#) of value_type.

Returns

An iterator pointing to the last inserted element or *pos* if *il* is empty.

This function will insert copies of the data in the [initializer_list](#) *il* into the `forward_list` before the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 272 of file `forward_list.tcc`.

5.455.3.26 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
size_type std::forward_list< _Tp, _Alloc >::max_size () const
[inline]`

Returns the largest possible size of `forward_list`.

Definition at line 742 of file `forward_list.h`.

5.455.3.27 `template<typename _Tp , typename _Alloc > template<typename
_Comp > void std::forward_list< _Tp, _Alloc >::merge (
forward_list< _Tp, _Alloc > && __list, _Comp __comp)`

Merge sorted lists according to comparison function.

Parameters

list Sorted list to merge.

comp Comparison function defining sort order.

Assumes that both *list* and this list are sorted according to *comp*. Merges elements of *list* into this list in sorted order, leaving *list* empty when complete. Elements in this list precede elements in *list* that are equivalent according to *comp*().

Definition at line 352 of file `forward_list.tcc`.

5.455.3.28 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::merge (forward_list< _Tp,
_Alloc > && __list) [inline]`

Merge sorted lists.

Parameters

list Sorted list to merge.

Assumes that both *list* and this list are sorted according to operator<(). Merges elements of *list* into this list in sorted order, leaving *list* empty when complete. Elements in this list precede elements in *list* that are equal.

Definition at line 1147 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::merge().

Referenced by std::forward_list< _Tp, _Alloc >::merge().

5.455.3.29 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list& std::forward_list< _Tp, _Alloc >::operator= (
forward_list< _Tp, _Alloc > && __list) [inline]`

The forward_list move assignment operator.

Parameters

list A forward_list of identical element and allocator types.

The contents of *list* are moved into this forward_list (without copying). *list* is a valid, but unspecified forward_list

Definition at line 568 of file forward_list.h.

5.455.3.30 `template<typename _Tp, typename _Alloc > forward_list< _Tp,
_Alloc > & std::forward_list< _Tp, _Alloc >::operator= (const
forward_list< _Tp, _Alloc > & __list)`

The forward_list assignment operator.

Parameters

list A forward_list of identical element and allocator types.

All the elements of *list* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 140 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::before_begin(), std::forward_list< _Tp, _Alloc >::begin(), std::forward_list< _Tp, _Alloc >::cbegin(), std::forward_list< _Tp, _Alloc >::cend(), std::forward_list< _Tp, _Alloc >::end(), std::forward_list< _Tp, _Alloc >::erase_after(), and std::forward_list< _Tp, _Alloc >::insert_after().

5.455.3.31 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list& std::forward_list< _Tp, _Alloc >::operator= (
std::initializer_list< _Tp > __il) [inline]`

The forward_list initializer list assignment operator.

Parameters

il An [initializer_list](#) of value_type.

Replace the contents of the forward_list with copies of the elements in the [initializer_list](#) *il*. This is linear in *il.size()*.

Definition at line 586 of file forward_list.h.

5.455.3.32 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::pop_front () [inline]`

Removes first element.

This is a typical stack operation. It shrinks the forward_list by one. Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before [pop_front\(\)](#) is called.

Definition at line 822 of file forward_list.h.

5.455.3.33 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::push_front (const _Tp & __val
) [inline]`

Add data to the front of the forward_list.

Parameters

val Data to be added.

This is a typical stack operation. The function creates an element at the front of the forward_list and assigns the given data to it. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 799 of file forward_list.h.

5.455.3.34 `template<typename _Tp, typename _Alloc > void
std::forward_list< _Tp, _Alloc >::remove (const _Tp & __val)`

Remove all elements equal to value.

Parameters

val The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 286 of file forward_list.tcc.

5.455.3.35 `template<typename _Tp, typename _Alloc> template<typename
_Pred> void std::forward_list< _Tp, _Alloc>::remove_if (_Pred
__pred)`

Remove all elements satisfying a predicate.

Parameters

pred Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 315 of file forward_list.tcc.

5.455.3.36 `template<typename _Tp, typename _Alloc> void
std::forward_list< _Tp, _Alloc>::resize (size_type __sz)`

Resizes the forward_list to the specified number of elements.

Parameters

sz Number of elements the forward_list should contain.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current size the forward_list is truncated, otherwise the forward_list is extended and the new elements are default constructed.

Definition at line 185 of file forward_list.tcc.

References `std::forward_list< _Tp, _Alloc>::before_begin()`, `std::forward_list< _Tp, _Alloc>::end()`, and `std::forward_list< _Tp, _Alloc>::erase_after()`.

5.455.3.37 `template<typename _Tp, typename _Alloc> void
std::forward_list< _Tp, _Alloc>::resize (size_type __sz, const
value_type & __val)`

Resizes the forward_list to the specified number of elements.

Parameters

sz Number of elements the forward_list should contain.

val Data with which new elements should be populated.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current size the forward_list is truncated, otherwise the forward_list is extended and new elements are populated with given data.

Definition at line 204 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::before_begin(), std::forward_list< _Tp, _Alloc >::end(), std::forward_list< _Tp, _Alloc >::erase_after(), and std::forward_list< _Tp, _Alloc >::insert_after().

5.455.3.38 **template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::reverse () [inline]**

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1191 of file forward_list.h.

5.455.3.39 **template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::sort () [inline]**

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1172 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::sort().

Referenced by std::forward_list< _Tp, _Alloc >::sort().

5.455.3.40 **template<typename _Tp, class _Alloc > template<typename _Comp > void std::forward_list< _Tp, _Alloc >::sort (_Comp __comp)**

Sort the [forward_list](#) using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 398 of file forward_list.tcc.

5.455.3.41 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::splice_after (const_iterator
__pos, forward_list< _Tp, _Alloc > && __list) [inline]`

Insert contents of another forward_list.

Parameters

pos Iterator referencing the element to insert after.

list Source list.

The elements of *list* are inserted in constant time after the element referenced by *pos*.
list becomes an empty list.

Requires this != *x*.

Definition at line 1033 of file forward_list.h.

5.455.3.42 `template<typename _Tp , typename _Alloc > void
std::forward_list< _Tp, _Alloc >::splice_after (const_iterator
__pos, forward_list< _Tp, _Alloc > && __list, const_iterator
__before, const_iterator __last)`

Insert range from another forward_list.

Parameters

pos Iterator referencing the element to insert after.

list Source list.

before Iterator referencing before the start of range in list.

last Iterator referencing the end of range in list.

Removes elements in the range (before,last) and inserts them after *pos* in constant time.

Undefined if *pos* is in (before,last).

Definition at line 233 of file forward_list.tcc.

5.455.3.43 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::splice_after (const_iterator
__pos, forward_list< _Tp, _Alloc > && __list, const_iterator __i
) [inline]`

Insert element from another forward_list.

Parameters

pos Iterator referencing the element to insert after.

list Source list.

i Iterator referencing the element before the element to move.

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

Definition at line 1050 of file forward_list.h.

5.455.3.44 `template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::swap (forward_list< _Tp,
_Alloc > & __list) [inline]`

Swaps data with another forward_list.

Parameters

list A forward_list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global [std::swap\(\)](#) function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 975 of file forward_list.h.

Referenced by `std::swap()`.

5.455.3.45 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void
std::forward_list< _Tp, _Alloc >::unique () [inline]`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1118 of file forward_list.h.

References `std::forward_list< _Tp, _Alloc >::unique()`.

Referenced by `std::forward_list< _Tp, _Alloc >::unique()`.

5.455.3.46 `template<typename _Tp , typename _Alloc > template<typename
_BinPred > void std::forward_list< _Tp, _Alloc >::unique (
_BinPred __binary_pred)`

Remove consecutive elements satisfying a predicate.

Parameters

binary_pred Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 331 of file forward_list.tcc.

References `std::forward_list< _Tp, _Alloc >::begin()`, `std::forward_list< _Tp, _Alloc >::end()`, and `std::forward_list< _Tp, _Alloc >::erase_after()`.

The documentation for this class was generated from the following files:

- [forward_list.h](#)
- [forward_list.tcc](#)

5.456 `std::fpos< _StateT >` Class Template Reference

Class representing stream positions.

Public Member Functions

- [fpos](#) (streamoff __off)
- [operator streamoff](#) () const
- [fpos operator+](#) (streamoff __off) const
- [fpos & operator+=](#) (streamoff __off)
- [fpos operator-](#) (streamoff __off) const
- [streamoff operator-](#) (const [fpos](#) &__other) const
- [fpos & operator-=](#) (streamoff __off)
- void [state](#) (_StateT __st)
- _StateT [state](#) () const

5.456.1 Detailed Description

`template<typename _StateT> class std::fpos< _StateT >`

Class representing stream positions. The standard places no requirements upon the template parameter StateT. In this implementation StateT must be DefaultConstructible, CopyConstructible and Assignable. The standard only requires that fpos should contain a member of type StateT. In this implementation it also contains an offset stored as a signed integer.

Parameters

StateT Type passed to and returned from [state\(\)](#).

Definition at line 112 of file postypes.h.

5.456.2 Constructor & Destructor Documentation

5.456.2.1 `template<typename _StateT> std::fpos< _StateT >::fpos (streamoff __off) [inline]`

Construct position from offset.

Definition at line 133 of file postypes.h.

5.456.3 Member Function Documentation

5.456.3.1 `template<typename _StateT> std::fpos< _StateT >::operator streamoff () const [inline]`

Convert to streamoff.

Definition at line 137 of file postypes.h.

5.456.3.2 `template<typename _StateT> fpos std::fpos< _StateT >::operator+ (streamoff __off) const [inline]`

Add position and offset.

Definition at line 178 of file postypes.h.

5.456.3.3 `template<typename _StateT> fpos& std::fpos< _StateT >::operator+=(streamoff __off) [inline]`

Add offset to this position.

Definition at line 154 of file postypes.h.

5.456.3.4 `template<typename _StateT> streamoff std::fpos< _StateT >::operator- (const fpos< _StateT > & __other) const [inline]`

Subtract position to return offset.

Definition at line 205 of file postypes.h.

5.456.3.5 `template<typename _StateT> fpos std::fpos< _StateT >::operator- (streamoff __off) const [inline]`

Subtract offset from position.

Definition at line 192 of file postypes.h.

5.456.3.6 `template<typename _StateT> fpos& std::fpos< _StateT >::operator-= (streamoff __off) [inline]`

Subtract offset from this position.

Definition at line 165 of file postypes.h.

5.456.3.7 `template<typename _StateT> _StateT std::fpos< _StateT >::state () const [inline]`

Return the last set value of *st*.

Definition at line 146 of file postypes.h.

5.456.3.8 `template<typename _StateT> void std::fpos< _StateT >::state (_StateT __st) [inline]`

Remember the value of *st*.

Definition at line 141 of file postypes.h.

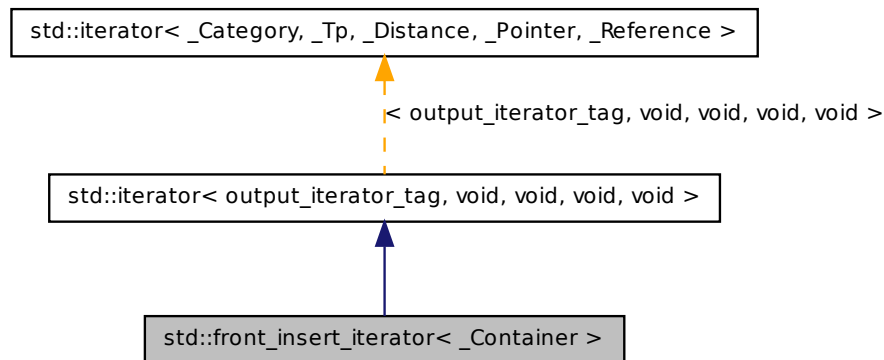
The documentation for this class was generated from the following file:

- [postypes.h](#)

5.457 `std::front_insert_iterator< _Container >` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::front_insert_iterator<_Container>`:



Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

Public Member Functions

- `front_insert_iterator` (`_Container &__x`)
- `front_insert_iterator` & `operator*` ()
- `front_insert_iterator` & `operator++` ()
- `front_insert_iterator` `operator++` (int)
- `front_insert_iterator` & `operator=` (const typename `_Container::value_type` &__value)
- `front_insert_iterator` & `operator=` (typename `_Container::value_type` &&__value)

Protected Attributes

- `_Container * container`

5.457.1 Detailed Description

template<typename _Container> class std::front_insert_iterator<_Container >

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using push_front.

Tip: Using the front_inserter function to create these iterators can save typing.

Definition at line 485 of file stl_iterator.h.

5.457.2 Member Typedef Documentation

**5.457.2.1 template<typename _Container > typedef _Container
std::front_insert_iterator<_Container >::container_type**

A nested typedef for the type of whatever container you used.

Definition at line 493 of file stl_iterator.h.

**5.457.2.2 typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::difference_type [inherited]**

Distance between iterators is represented as this type.

Definition at line 124 of file stl_iterator_base_types.h.

**5.457.2.3 typedef output_iterator_tag std::iterator< output_iterator_tag , void
, void , void , void >::iterator_category [inherited]**

One of the [tag types](#).

Definition at line 120 of file stl_iterator_base_types.h.

**5.457.2.4 typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::pointer [inherited]**

This type represents a pointer-to-value_type.

Definition at line 126 of file stl_iterator_base_types.h.

5.457.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 128 of file stl_iterator_base_types.h.

5.457.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file stl_iterator_base_types.h.

5.457.3 Constructor & Destructor Documentation

5.457.3.1 `template<typename _Container > std::front_insert_iterator< _Container >::front_insert_iterator (_Container & __x) [inline, explicit]`

The only way to create this iterator is with a container.

Definition at line 496 of file stl_iterator.h.

5.457.4 Member Function Documentation

5.457.4.1 `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator* () [inline]`

Simply returns *this.

Definition at line 534 of file stl_iterator.h.

5.457.4.2 `template<typename _Container > front_insert_iterator std::front_insert_iterator< _Container >::operator++ (int) [inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 544 of file stl_iterator.h.

5.457.4.3 `template<typename _Container > front_insert_iterator&
std::front_insert_iterator< _Container >::operator++ ()
[inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 539 of file stl_iterator.h.

5.457.4.4 `template<typename _Container > front_insert_iterator&
std::front_insert_iterator< _Container >::operator= (const
typename _Container::value_type & __value) [inline]`

Parameters

value An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

Definition at line 518 of file stl_iterator.h.

The documentation for this class was generated from the following file:

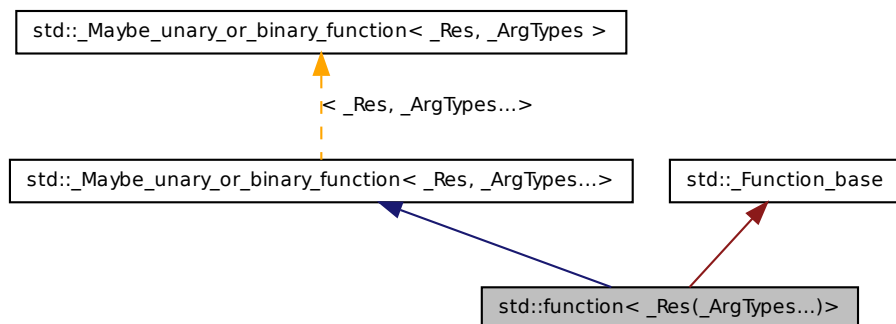
- [stl_iterator.h](#)

5.458 `std::function< _Res(_ArgTypes...)>` Class Template Reference

Primary class template for `std::function`.

Polymorphic function wrapper.

Inheritance diagram for std::function< _Res(_ArgTypes...)>:



Public Types

- typedef `_Res` **result_type**

Public Member Functions

- `function` ()
- `function` (nullptr_t)
- `function` (function &&__x)
- template<typename _Functor >
`function` (_Functor __f, typename `enable_if`< !is_integral< _Functor >::value, _Useless >::type=_Useless())
- `function` (const function &&__x)
- `operator bool` () const
- template<typename _Res2 , typename... _ArgTypes2>
void **operator!=** (const function< _Res2(_ArgTypes2...)> &) const
- `_Res` `operator()` (_ArgTypes...__args) const
- function & `operator=` (nullptr_t)
- template<typename _Functor >
`enable_if`<!is_integral< _Functor >::value, function & >::type `operator=` (_Functor &&__f)
- template<typename _Functor >
`enable_if`<!is_integral< _Functor >::value, function & >::type `operator=` (reference_wrapper< _Functor > __f)

- function & [operator=](#) (const function &__x)
- function & [operator=](#) (function &&__x)
- template<typename _Res2, typename... _ArgTypes2>
void **operator==** (const function< _Res2(_ArgTypes2...)> &) const
- void [swap](#) (function &__x)
- template<typename _Functor >
_Functor * [target](#) ()
- template<typename _Functor >
const _Functor * [target](#) () const
- const [type_info](#) & [target_type](#) () const

Private Types

- typedef bool(* **_Manager_type**)(_Any_data &, const _Any_data &, _Manager_operation)

Private Member Functions

- bool **_M_empty** () const

Private Attributes

- _Any_data **_M_functor**
- _Manager_type **_M_manager**

Static Private Attributes

- static const std::size_t **_M_max_align**
- static const std::size_t **_M_max_size**

5.458.1 Detailed Description

template<typename _Res, typename... _ArgTypes> class std::function< _Res(_ArgTypes...)>

Primary class template for std::function.

Polymorphic function wrapper.

Definition at line 1859 of file functional.

5.458.2 Constructor & Destructor Documentation

5.458.2.1 `template<typename _Res , typename... _ArgTypes> std::function<
_Res(_ArgTypes...)>::function () [inline]`

Default construct creates an empty function call wrapper.

Postcondition

`!(bool)*this`

Definition at line 1876 of file functional.

5.458.2.2 `template<typename _Res , typename... _ArgTypes> std::function<
_Res(_ArgTypes...)>::function (nullptr_t) [inline]`

Creates an empty function call wrapper.

Postcondition

`!(bool)*this`

Definition at line 1882 of file functional.

5.458.2.3 `template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (const function<
_Res(_ArgTypes...)> & __x)`

Function copy constructor.

Parameters

x A function object with identical call signature.

Postcondition

`(bool)*this == (bool)x`

The newly-created function contains a copy of the target of *x* (if it has one).

Definition at line 2113 of file functional.

5.458.2.4 `template<typename _Res , typename... _ArgTypes> std::function<
_Res(_ArgTypes...)>::function (function< _Res(_ArgTypes...)> &&
__x) [inline]`

Function move constructor.

Parameters

x A function object rvalue with identical call signature.

The newly-created function contains the target of *x* (if it has one).

Definition at line 1901 of file functional.

```
5.458.2.5 template<typename _Res , typename... _ArgTypes>
            template<typename _Function > std::function<
              _Res(_ArgTypes...)>::function ( _Function __f, typename enable_if<
                !is_integral< _Function >::value, _Useless >::type = _Useless() )
```

Builds a function that targets a copy of the incoming function object.

Parameters

f A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to *Res*.

The newly-created function object will target a copy of *f*. If *f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `f.get()`. If *f* is a NULL function pointer or NULL pointer-to-member, the newly-created object will be empty.

If *f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2127 of file functional.

5.458.3 Member Function Documentation

```
5.458.3.1 template<typename _Res , typename... _ArgTypes> std::function<
            _Res(_ArgTypes...)>::operator bool ( ) const [inline,
            explicit]
```

Determine if the function wrapper has a target.

Returns

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw an exception.

Definition at line 2056 of file functional.

5.458.3.2 `template<typename _Res , typename... _ArgTypes> _Res
std::function< _Res(_ArgTypes...)>::operator() (_ArgTypes...
__args) const`

Invokes the function targeted by `*this`.

Returns

the result of the target.

Exceptions

bad_function_call when `!(bool)*this`

The function call operator invokes the target function object stored by `this`.

Definition at line 2145 of file functional.

5.458.3.3 `template<typename _Res , typename... _ArgTypes> function&
std::function< _Res(_ArgTypes...)>::operator= (function<
_Res(_ArgTypes...)> && __x) [inline]`

Function move-assignment operator.

Parameters

x A function rvalue with identical call signature.

Returns

`*this`

The target of `x` is moved to `*this`. If `x` has no target, then `*this` will be empty.

If `x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 1961 of file functional.

5.458.3.4 `template<typename _Res , typename... _ArgTypes>
template<typename _Functor > enable_if<!is_integral<_
_Functor>::value, function&>::type std::function<
_Res(_ArgTypes...)>::operator= (_Functor && __f) [inline]`

Function assignment to a new target.

Parameters

f A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to Res.

Returns

*this

This function object wrapper will target a copy of *f*. If *f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `f.get()`. If *f* is a NULL function pointer or NULL pointer-to-member, `this` object will be empty.

If *f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2004 of file functional.

```
5.458.3.5 template<typename _Res , typename... _ArgTypes>
template<typename _Functor > enable_if<!is_integral<_Functor>::value, function<>::type std::function<
_Res(_ArgTypes...)>::operator= ( reference_wrapper< _Functor >
__f ) [inline]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2013 of file functional.

```
5.458.3.6 template<typename _Res , typename... _ArgTypes> function&
std::function< _Res(_ArgTypes...)>::operator= ( const function<
_Res(_ArgTypes...)> & __x ) [inline]
```

Function assignment operator.

Parameters

x A function with identical call signature.

Postcondition

`(bool)*this == (bool)x`

Returns

*this

The target of *x* is copied to **this*. If *x* has no target, then **this* will be empty.

If *x* targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 1943 of file functional.

```
5.458.3.7  template<typename _Res , typename... _ArgTypes> function&
            std::function< _Res(_ArgTypes...)>::operator= ( nullptr_t )
            [inline]
```

Function assignment to zero.

Postcondition

!(bool)*this

Returns

*this

The target of **this* is deallocated, leaving it empty.

Definition at line 1975 of file functional.

```
5.458.3.8  template<typename _Res , typename... _ArgTypes> void
            std::function< _Res(_ArgTypes...)>::swap ( function<
            _Res(_ArgTypes...)> & __x ) [inline]
```

Swap the targets of two function objects.

Parameters

f A function with identical call signature.

Swap the targets of *this* function object and *f*. This function will not throw an exception.

Definition at line 2028 of file functional.

References std::swap().

```
5.458.3.9  template<typename _Res , typename... _ArgTypes>
            template<typename _Functor > const _Functor * std::function<
            _Res(_ArgTypes...)>::target ( ) const
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2191 of file functional.

5.458.3.10 `template<typename _Res , typename... _ArgTypes>
template<typename _Functor > _Functor * std::function<
_Res(_ArgTypes...)>::target ()`

Access the stored target function object.

Returns

Returns a pointer to the stored target function object, if `typeid(Functor).equals(target_type())`; otherwise, a NULL pointer.

This function will not throw an exception.

Definition at line 2172 of file functional.

5.458.3.11 `template<typename _Res , typename... _ArgTypes> const type_info
& std::function< _Res(_ArgTypes...)>::target_type () const`

Determine the type of the target of this function object wrapper.

Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw an exception.

Definition at line 2156 of file functional.

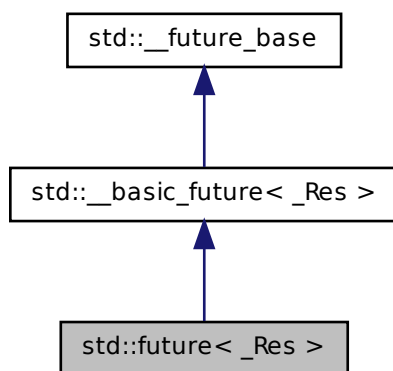
The documentation for this class was generated from the following file:

- [functional](#)

5.459 `std::future< _Res >` Class Template Reference

Primary template for future.

Inheritance diagram for std::future< _Res >:



Public Member Functions

- [future](#) ([future](#) &&__uf)
- **future** (const [future](#) &)
- [_Res](#) [get](#) ()
- [future](#) & **operator=** (const [future](#) &)
- [future](#) & **operator=** ([future](#) &&__fut)
- bool **valid** () const
- void **wait** () const
- template<typename _Rep , typename _Period >
bool **wait_for** (const [chrono::duration](#)< _Rep, _Period > &__rel) const
- template<typename _Clock , typename _Duration >
bool **wait_until** (const [chrono::time_point](#)< _Clock, _Duration > &__abs)
const

Static Public Member Functions

- template<typename _Res , typename _Allocator >
static [Ptr](#)< [Result_alloc](#)< _Res, _Allocator > >::type [_S_allocate_result](#)
(const _Allocator &__a)

Protected Types

- typedef `__future_base::_Result` < _Res > & `__result_type`

Protected Member Functions

- `__result_type` `_M_get_result` ()
- void `_M_swap` (`__basic_future` &__that)

Friends

- template<typename _Fn, typename... _Args>
`future` < typename result_of< _Fn(_Args...)>::type > `async` (launch, _Fn &&, _Args &&...)
- class `packaged_task`
- class `promise` < _Res >

5.459.1 Detailed Description

template<typename _Res> class std::future< _Res >

Primary template for future.

Definition at line 554 of file future.

5.459.2 Constructor & Destructor Documentation

5.459.2.1 template<typename _Res > std::future< _Res >::future (future< _Res > && __uf) [inline]

Move constructor.

Definition at line 572 of file future.

5.459.3 Member Function Documentation

5.459.3.1 template<typename _Res> __result_type std::__basic_future< _Res >::_M_get_result () [inline, protected, inherited]

Wait for the state to be ready and rethrow any stored exception.

Definition at line 508 of file future.

Referenced by `std::shared_future<_Res>::get()`, `std::future<void>::get()`, and `std::future<_Res>::get()`.

5.459.3.2 `template<typename _Res> _Res std::future<_Res>::get ()`
`[inline]`

Retrieving the value.

Definition at line 586 of file `future`.

References `std::__basic_future<_Res>::_M_get_result()`.

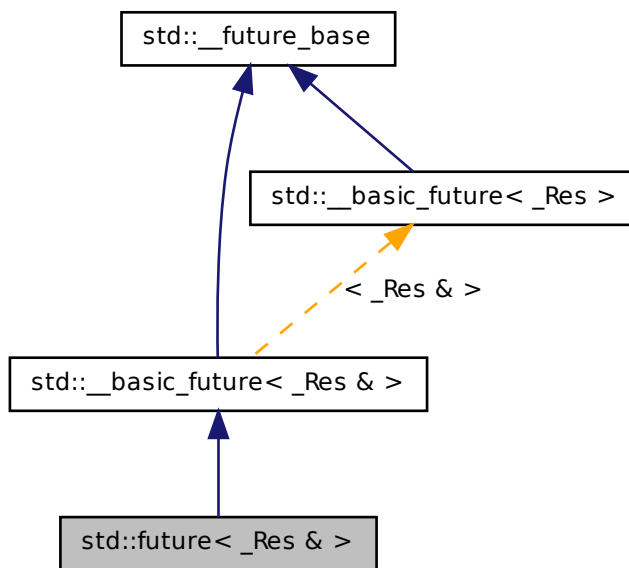
The documentation for this class was generated from the following file:

- [future](#)

5.460 `std::future<_Res &>` Class Template Reference

Partial specialization for `future<R&>`

Inheritance diagram for `std::future< _Res & >`:



Public Member Functions

- `future` (`future` &&__uf)
- `future` (const `future` &)
- `_Res` & `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&__fut)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`< `_Rep`, `_Period` > &__rel) const
- bool `wait_until` (const `chrono::time_point`< `_Clock`, `_Duration` > &__abs) const

Static Public Member Functions

- `template<typename _Res , typename _Allocator >`

```
static _Ptr< _Result_alloc< _Res, _Allocator > >::type _S_allocate_result
(const _Allocator &__a)
```

Protected Types

- typedef `__future_base::_Result< _Res & > & __result_type`

Protected Member Functions

- `__result_type M_get_result ()`
- `void M_swap (__basic_future &__that)`

Friends

- `template<typename _Fn , typename... _Args>`
`future< typename result_of< _Fn(_Args...)>::type > async (launch, _Fn &&, _Args &&...)`
- class `packaged_task`
- class `promise< _Res & >`

5.460.1 Detailed Description

```
template<typename _Res> class std::future< _Res & >
```

Partial specialization for future<R&>

Definition at line 595 of file future.

5.460.2 Constructor & Destructor Documentation

5.460.2.1 `template<typename _Res > std::future< _Res & >::future (future< _Res & > && __uf) [inline]`

Move constructor.

Definition at line 613 of file future.

5.460.3 Member Function Documentation

5.460.3.1 `__result_type std::__basic_future<_Res & >::_M_get_result ()`
`[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 508 of file future.

References `std::rethrow_exception()`.

5.460.3.2 `template<typename _Res > _Res& std::future<_Res & >::get ()`
`[inline]`

Retrieving the value.

Definition at line 627 of file future.

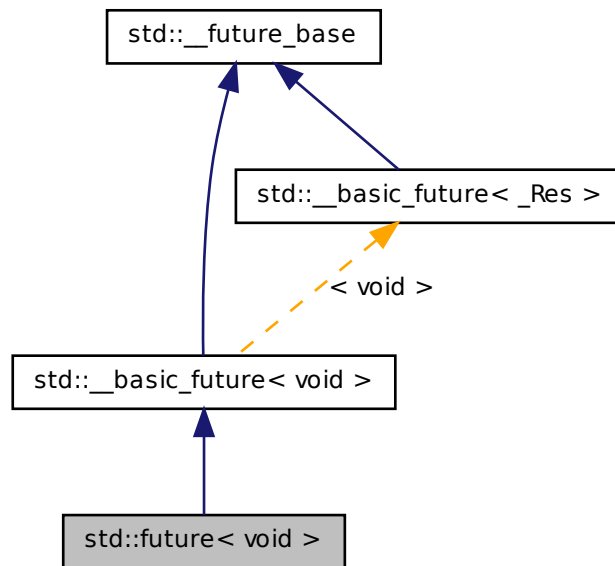
The documentation for this class was generated from the following file:

- [future](#)

5.461 `std::future< void >` Class Template Reference

Explicit specialization for [future<void>](#)

Inheritance diagram for std::future< void >:



Public Member Functions

- `future` (`future` &&__uf)
- `future` (const `future` &)
- void `get` ()
- `future` & `operator=` (const `future` &)
- `future` & `operator=` (`future` &&__fut)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration`< _Rep, _Period > &__rel) const
- bool `wait_until` (const `chrono::time_point`< _Clock, _Duration > &__abs) const

Static Public Member Functions

- `template<typename _Res, typename _Allocator >`

```
static \_Ptr< \_Result\_alloc< \_Res, \_Allocator > >::type \_S\_allocate\_result
(const \_Allocator &__a)
```

Protected Types

- typedef [__future_base::_Result](#)< void > & [__result_type](#)

Protected Member Functions

- [__result_type](#) [_M_get_result](#) ()
- void [_M_swap](#) ([__basic_future](#) &__that)

Friends

- template<typename [_Fn](#) , typename... [_Args](#)>
[future](#)< typename result_of< [_Fn](#)([_Args](#)...)>::type > [async](#) (launch, [_Fn](#) &&, [_Args](#) &&...)
- class [packaged_task](#)
- class [promise](#)< void >

5.461.1 Detailed Description

template<> class std::future< void >

Explicit specialization for [future<void>](#)

Definition at line 636 of file future.

5.461.2 Constructor & Destructor Documentation

5.461.2.1 [std::future< void >::future](#) ([future< void >](#) && [__uf](#))
[inline]

Move constructor.

Definition at line 654 of file future.

5.461.3 Member Function Documentation

5.461.3.1 `__result_type std::__basic_future< void >::_M_get_result ()` [inline, protected, inherited]

Wait for the state to be ready and rethrow any stored exception.

Definition at line 508 of file future.

5.461.3.2 `void std::future< void >::get ()` [inline]

Retrieving the value.

Definition at line 668 of file future.

References `std::__basic_future< _Res >::_M_get_result()`.

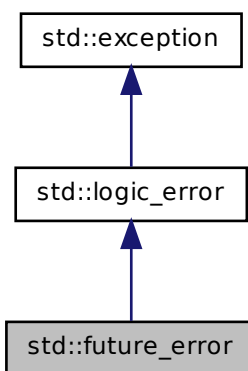
The documentation for this class was generated from the following file:

- [future](#)

5.462 std::future_error Class Reference

Exception type thrown by futures.

Inheritance diagram for std::future_error:



Public Member Functions

- **future_error** ([error_code](#) __ec)
- const [error_code](#) & **code** () const throw ()
- virtual const char * [what](#) () const throw ()

5.462.1 Detailed Description

Exception type thrown by futures.

Definition at line 85 of file future.

5.462.2 Member Function Documentation

5.462.2.1 virtual const char* std::future_error::what () const throw () [**virtual**]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::logic_error](#).

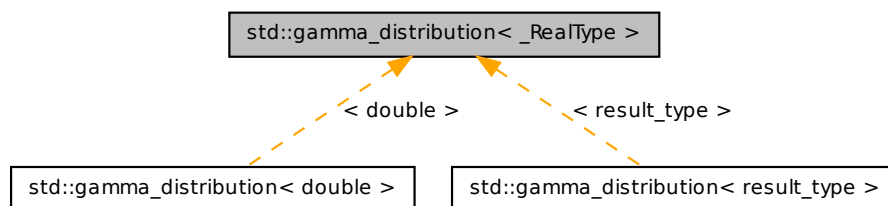
The documentation for this class was generated from the following file:

- [future](#)

5.463 std::gamma_distribution< _RealType > Class Template Reference

A gamma continuous distribution for random numbers.

Inheritance diagram for `std::gamma_distribution<_RealType>`:



Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- [gamma_distribution](#) (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- [gamma_distribution](#) (const [param_type](#) &__p)
- `_RealType alpha () const`
- `_RealType beta () const`
- [result_type max \(\) const](#)
- [result_type min \(\) const](#)
- template<typename `_UniformRandomNumberGenerator` >
[result_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >
[result_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (const [param_type](#) &__param)
- [param_type param \(\) const](#)
- void [reset \(\)](#)

Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::gamma_distribution< _RealType1 > &)`
- `template<typename _RealType1 >
bool operator== (const std::gamma_distribution< _RealType1 > &__d1, const std::gamma_distribution< _RealType1 > &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::gamma_distribution< _RealType1 > &)`

5.463.1 Detailed Description

`template<typename _RealType = double> class std::gamma_distribution< _RealType >`

A gamma continuous distribution for random numbers. The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Definition at line 2353 of file random.h.

5.463.2 Member Typedef Documentation

5.463.2.1 `template<typename _RealType = double> typedef _RealType
std::gamma_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2360 of file random.h.

5.463.3 Constructor & Destructor Documentation

5.463.3.1 `template<typename _RealType = double> std::gamma_distribution<
_RealType >::gamma_distribution (_RealType __alpha_val
= _RealType(1), _RealType __beta_val = _RealType(1))
[inline, explicit]`

Constructs a gamma distribution with parameters α and β .

Definition at line 2405 of file random.h.

5.463.4 Member Function Documentation

5.463.4.1 `template<typename _RealType = double> _RealType
std::gamma_distribution< _RealType >::alpha () const
[inline]`

Returns the α of the distribution.

Definition at line 2426 of file random.h.

5.463.4.2 `template<typename _RealType = double> _RealType
std::gamma_distribution< _RealType >::beta () const [inline]`

Returns the β of the distribution.

Definition at line 2433 of file random.h.

5.463.4.3 `template<typename _RealType = double> result_type
std::gamma_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2462 of file random.h.

Referenced by std::gamma_distribution< result_type >::max().

5.463.4.4 `template<typename _RealType = double> result_type
std::gamma_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2455 of file random.h.

5.463.4.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::gamma_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2470 of file random.h.

Referenced by std::gamma_distribution< result_type >::operator()().

5.463.4.6 `template<typename _RealType > template<typename
_UniformRandomNumberGenerator > gamma_distribution<
_RealType >::result_type std::gamma_distribution< _RealType
>::operator() (_UniformRandomNumberGenerator & __urng,
const param_type & __param)`

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables"
ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

Definition at line 2012 of file random.tcc.

References `std::log()`, and `std::pow()`.

5.463.4.7 `template<typename _RealType = double> void
std::gamma_distribution< _RealType >::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 2448 of file random.h.

5.463.4.8 `template<typename _RealType = double> param_type
std::gamma_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2440 of file random.h.

Referenced by `std::gamma_distribution< result_type >::operator()`.

5.463.4.9 `template<typename _RealType = double> void
std::gamma_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2419 of file random.h.

Referenced by `std::negative_binomial_distribution< _IntType >::reset()`,
`std::student_t_distribution< _RealType >::reset()`, `std::fisher_f_distribution< _
RealType >::reset()`, and `std::chi_squared_distribution< _RealType >::reset()`.

5.463.5 Friends And Related Function Documentation

5.463.5.1 `template<typename _RealType = double> template<typename
_RealType1, typename _CharT, typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream<_CharT, _Traits> &, const
std::gamma_distribution<_RealType1> &) [friend]`

Inserts a gamma_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A gamma_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.463.5.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::gamma_distribution<
_RealType1> & __d1, const std::gamma_distribution<_RealType1
> & __d2) [friend]`

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2485 of file random.h.

5.463.5.3 `template<typename _RealType = double> template<typename
_RealType1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream<_CharT, _Traits> &,
std::gamma_distribution<_RealType1> &) [friend]`

Extracts a gamma_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A gamma_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.464 `std::gamma_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [gamma_distribution<_RealType>](#) **distribution_type**

Public Member Functions

- **param_type** (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- `_RealType` **alpha** () const
- `_RealType` **beta** () const

Friends

- class [gamma_distribution<_RealType>](#)
- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.464.1 Detailed Description

`template<typename _RealType = double> struct std::gamma_distribution<_RealType>::param_type`

Parameter type.

Definition at line 2362 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.465 std::geometric_distribution< _IntType > Class Template Reference

A discrete geometric random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef _IntType [result_type](#)

Public Member Functions

- **geometric_distribution** (double __p=0.5)
- **geometric_distribution** (const [param_type](#) &__p)
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- double **p** () const
- void **param** (const [param_type](#) &__param)
- [param_type](#) **param** () const
- void **reset** ()

5.465.1 Detailed Description

```
template<typename _IntType = int> class std::geometric_distribution< _-
_IntType >
```

A discrete geometric random number distribution. The formula for the geometric probability density function is $p(i|p) = (1 - p)p^{i-1}$ where p is the parameter of the distribution.

Definition at line 3622 of file random.h.

5.465.2 Member Typedef Documentation

5.465.2.1 `template<typename _IntType = int> typedef _IntType std::geometric_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3629 of file random.h.

5.465.3 Member Function Documentation

5.465.3.1 `template<typename _IntType = int> result_type std::geometric_distribution< _IntType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3715 of file random.h.

5.465.3.2 `template<typename _IntType = int> result_type std::geometric_distribution< _IntType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3708 of file random.h.

5.465.3.3 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type std::geometric_distribution< _IntType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3723 of file random.h.

References `std::geometric_distribution< _IntType >::operator()()`, and `std::geometric_distribution< _IntType >::param()`.

Referenced by `std::geometric_distribution< _IntType >::operator()()`.

5.465.3.4 `template<typename _IntType = int> double std::geometric_distribution< _IntType >::p () const [inline]`

Returns the distribution parameter p.

5.466 `std::geometric_distribution< _IntType >::param_type` Struct Reference 2505

Definition at line 3686 of file random.h.

5.465.3.5 `template<typename _IntType = int> void std::geometric_distribution< _IntType >::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 3701 of file random.h.

5.465.3.6 `template<typename _IntType = int> param_type std::geometric_distribution< _IntType >::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 3693 of file random.h.

Referenced by `std::geometric_distribution< _IntType >::operator()()`, `std::operator==()`, and `std::operator>>()`.

5.465.3.7 `template<typename _IntType = int> void std::geometric_distribution< _IntType >::reset () [inline]`

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 3680 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.466 `std::geometric_distribution< _IntType >::param_type` Struct Reference

Public Types

- typedef [geometric_distribution< _IntType >](#) `distribution_type`

Public Member Functions

- **param_type** (double __p=0.5)
- double **p** () const

Friends

- class **geometric_distribution**< _IntType >
- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.466.1 Detailed Description

template<typename _IntType = int> struct std::geometric_distribution< _IntType >::param_type

Parameter type.

Definition at line 3631 of file random.h.

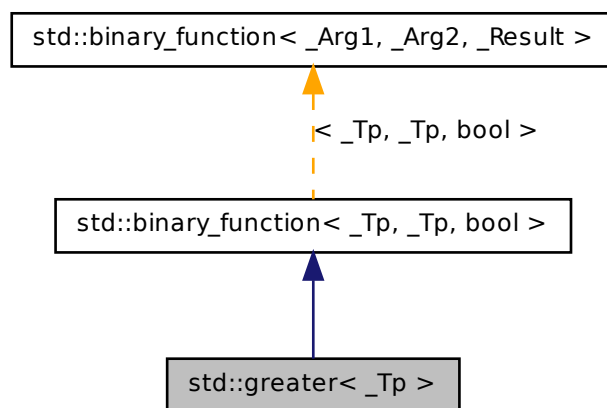
The documentation for this struct was generated from the following file:

- [random.h](#)

5.467 std::greater< _Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::greater< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.467.1 Detailed Description

```
template<typename _Tp> struct std::greater< _Tp >
```

One of the [comparison functors](#).

Definition at line 217 of file `stl_function.h`.

5.467.2 Member Typedef Documentation

5.467.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.467.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.467.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

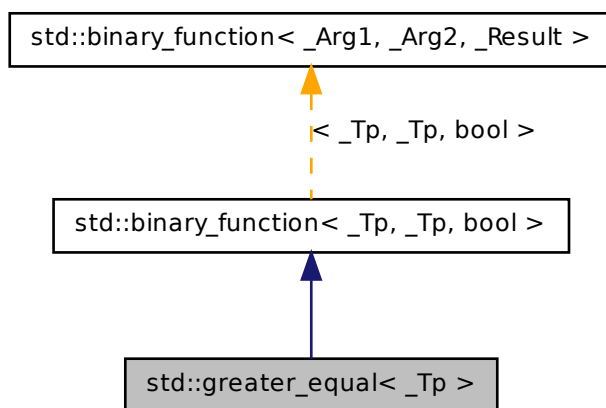
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.468 `std::greater_equal< _Tp >` Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::greater_equal< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.468.1 Detailed Description

```
template<typename _Tp> struct std::greater_equal< _Tp >
```

One of the [comparison functors](#).

Definition at line 235 of file `stl_function.h`.

5.468.2 Member Typedef Documentation

5.468.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.468.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.468.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.469 std::gslice Class Reference

Class defining multi-dimensional subset of an array.

Public Member Functions

- [gslice](#) ()
- [gslice](#) (size_t, const [valarray](#)< size_t > &, const [valarray](#)< size_t > &)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- [valarray](#)< size_t > [size](#) () const
- size_t [start](#) () const
- [valarray](#)< size_t > [stride](#) () const

Friends

- class `valarray`

5.469.1 Detailed Description

Class defining multi-dimensional subset of an array. The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 63 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

5.470 `std::gslice_array< _Tp >` Class Template Reference

Reference to multi-dimensional subset of an array.

Public Types

- typedef `_Tp value_type`

Public Member Functions

- [gslice_array](#) (const [gslice_array](#) &)
- template<class `_Dom` >
void **operator%=** (const `_Expr< _Dom, _Tp >` &) const
- void **operator%=** (const [valarray](#)< `_Tp` > &) const
- void **operator&=** (const [valarray](#)< `_Tp` > &) const
- template<class `_Dom` >
void **operator&=** (const `_Expr< _Dom, _Tp >` &) const

- `template<class _Dom >`
`void operator*= (const _Expr< _Dom, _Tp > &) const`
- `void operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator+= (const _Expr< _Dom, _Tp > &) const`
- `void operator+= (const valarray< _Tp > &) const`
- `void operator-= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator-= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void operator/= (const _Expr< _Dom, _Tp > &) const`
- `void operator/= (const valarray< _Tp > &) const`
- `void operator<<= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator<<= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const _Tp &) const`
- `gslice_array & operator= (const gslice_array &)`
- `template<class _Dom >`
`void operator= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator|= (const _Expr< _Dom, _Tp > &) const`

Friends

- `class valarray< _Tp >`

5.470.1 Detailed Description

`template<typename _Tp> class std::gslice_array< _Tp >`

Reference to multi-dimensional subset of an array. A [gslice_array](#) is a reference to the actual elements of an array specified by a [gslice](#). The way to get a [gslice_array](#) is to call `operator[]`([gslice](#)) on a [valarray](#). The returned [gslice_array](#) then permits carrying operations out on the referenced subset of elements in the original [valarray](#). For example, `operator+=(valarray)` will add values to the subset of elements in the underlying [valarray](#) this [gslice_array](#) refers to.

5.471 `std::has_nothrow_copy_assign< _Tp >` Struct Template Reference 2513

Parameters

Tp Element type.

Definition at line 59 of file `gslice_array.h`.

The documentation for this class was generated from the following file:

- [gslice_array.h](#)

5.471 `std::has_nothrow_copy_assign< _Tp >` Struct Template Reference

[has_nothrow_copy_assign](#)

Inherits `integral_constant< bool, __has_nothrow_assign(_Tp)>`.

5.471.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_copy_assign< _Tp >
```

[has_nothrow_copy_assign](#)

Definition at line 297 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.472 `std::has_nothrow_copy_constructor< _Tp >` Struct Template Reference

[has_nothrow_copy_constructor](#)

Inherits `integral_constant< bool, __has_nothrow_copy(_Tp)>`.

5.472.1 Detailed Description

```
template<typename _Tp> struct std::has_nothrow_copy_constructor< _Tp >
```

[has_nothrow_copy_constructor](#)

Definition at line 291 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.473 `std::has_nothrow_default_constructor< _Tp >` Struct Template Reference

[has_nothrow_default_constructor](#)

Inherits `integral_constant< bool, __has_nothrow_constructor(_Tp)>`.

5.473.1 Detailed Description

`template<typename _Tp> struct std::has_nothrow_default_constructor< _Tp >`

[has_nothrow_default_constructor](#)

Definition at line 285 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.474 `std::has_trivial_copy_assign< _Tp >` Struct Template Reference

[has_trivial_copy_assign](#)

Inherits `integral_constant< bool, __has_trivial_assign(_Tp)>`.

5.474.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_copy_assign< _Tp >`

[has_trivial_copy_assign](#)

Definition at line 273 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.475 std::has_trivial_copy_constructor< _Tp > Struct Template Reference 2515

5.475 std::has_trivial_copy_constructor< _Tp > Struct Template Reference

[has_trivial_copy_constructor](#)

Inherits `integral_constant< bool, __has_trivial_copy(_Tp)>`.

5.475.1 Detailed Description

template<typename _Tp> struct std::has_trivial_copy_constructor< _Tp >

[has_trivial_copy_constructor](#)

Definition at line 267 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.476 std::has_trivial_default_constructor< _Tp > Struct Template Reference

[has_trivial_default_constructor](#)

Inherits `integral_constant< bool, __has_trivial_constructor(_Tp)>`.

5.476.1 Detailed Description

template<typename _Tp> struct std::has_trivial_default_constructor< _Tp >

[has_trivial_default_constructor](#)

Definition at line 261 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.477 std::has_trivial_destructor< _Tp > Struct Tem- plate Reference

[has_trivial_destructor](#)

Inherits `integral_constant< bool, __has_trivial_destructor(_Tp)>`.

5.477.1 Detailed Description

`template<typename _Tp> struct std::has_trivial_destructor< _Tp >`

[has_trivial_destructor](#)

Definition at line 279 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.478 `std::hash< _Tp >` Struct Template Reference

Primary class template `hash`.

Inherits `std::__hash_base< size_t, _Tp >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (_Tp __val) const`
- `template<> size_t operator() (double __val) const`
- `template<> size_t operator() (float __val) const`
- `template<> size_t operator() (unsigned long long __val) const`
- `template<> size_t operator() (unsigned long __val) const`
- `template<> size_t operator() (unsigned int __val) const`
- `template<> size_t operator() (unsigned short __val) const`
- `template<> size_t operator() (long long __val) const`

- `template<>`
`size_t operator() (long __val) const`
- `template<>`
`size_t operator() (int __val) const`
- `template<>`
`size_t operator() (short __val) const`
- `template<>`
`size_t operator() (char32_t __val) const`
- `template<>`
`size_t operator() (char16_t __val) const`
- `template<>`
`size_t operator() (wchar_t __val) const`
- `template<>`
`size_t operator() (unsigned char __val) const`
- `template<>`
`size_t operator() (signed char __val) const`
- `template<>`
`size_t operator() (char __val) const`
- `template<>`
`size_t operator() (bool __val) const`

5.478.1 Detailed Description

`template<typename _Tp> struct std::hash< _Tp >`

Primary class template hash.

Definition at line 56 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

5.479 `std::hash< __debug::bitset< _Nb > >` Struct Template Reference

`std::hash` specialization for `bitset`.

Inherits `std::__hash_base< size_t, __debug::bitset< _Nb > >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (const __debug::bitset< _Nb > &__b) const`

5.479.1 Detailed Description

`template<size_t _Nb> struct std::hash< __debug::bitset< _Nb > >`

[std::hash](#) specialization for `bitset`.

Definition at line 387 of file `debug/bitset`.

The documentation for this struct was generated from the following file:

- [debug/bitset](#)

5.480 `std::hash< __debug::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

Inherits `std::__hash_base< size_t, __debug::vector< bool, _Alloc > >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (const __debug::vector< bool, _Alloc > &__b) const`

5.480.1 Detailed Description

`template<typename _Alloc> struct std::hash< __debug::vector< bool, _Alloc > >`

[std::hash](#) specialization for `vector<bool>`.

Definition at line 586 of file `debug/vector`.

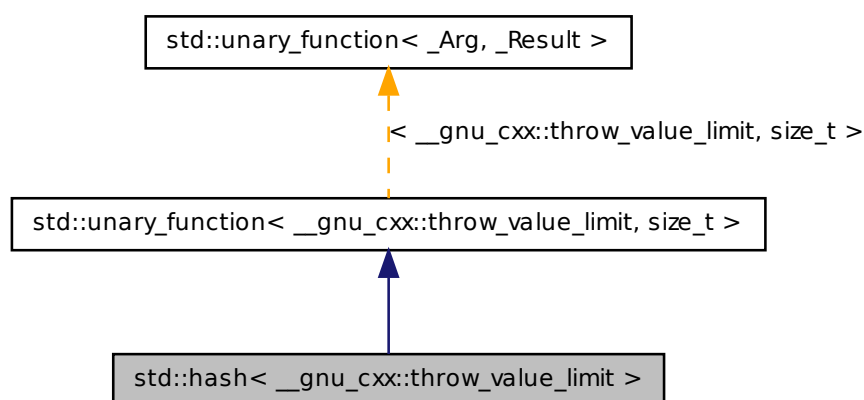
The documentation for this struct was generated from the following file:

- [debug/vector](#)

5.481 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:



Public Types

- typedef `__gnu_cxx::throw_value_limit` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const __gnu_cxx::throw_value_limit &__val) const`

5.481.1 Detailed Description

`template<> struct std::hash< __gnu_cxx::throw_value_limit >`

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 734 of file `throw_allocator.h`.

5.481.2 Member Typedef Documentation

5.481.2.1 `typedef __gnu_cxx::throw_value_limit std::unary_function<
__gnu_cxx::throw_value_limit, size_t >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.481.2.2 `typedef size_t std::unary_function< __gnu_cxx::throw_value_limit,
size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

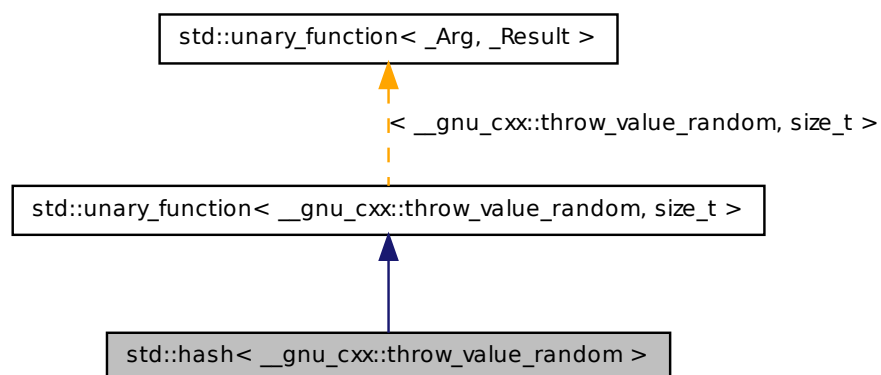
- [throw_allocator.h](#)

5.482 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Explicit specialization of [std::hash](#) for [__gnu_cxx::throw_value_limit](#).

5.482 std::hash< __gnu_cxx::throw_value_random > Struct Template Reference

Inheritance diagram for std::hash< __gnu_cxx::throw_value_random >:



Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const __gnu_cxx::throw_value_random &__val) const`

5.482.1 Detailed Description

template<> struct std::hash< __gnu_cxx::throw_value_random >

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 748 of file `throw_allocator.h`.

5.482.2 Member Typedef Documentation

5.482.2.1 `typedef __gnu_cxx::throw_value_random std::unary_function<
__gnu_cxx::throw_value_random , size_t >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.482.2.2 `typedef size_t std::unary_function< __gnu_-
cxx::throw_value_random , size_t >::result_type
[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

5.483 std::hash< __profile::bitset< _Nb > > Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inherits `std::__hash_base< size_t, __profile::bitset< _Nb > >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (const __profile::bitset< _Nb > &__b) const`

5.483.1 Detailed Description

`template<size_t _Nb> struct std::hash< __profile::bitset< _Nb > >`

[std::hash](#) specialization for `bitset`.

5.484 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference 2523

Definition at line 361 of file `profile/bitset`.

The documentation for this struct was generated from the following file:

- [profile/bitset](#)

5.484 `std::hash< __profile::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

Inherits `std::__hash_base< size_t, __profile::vector< bool, _Alloc > >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator() (const __profile::vector< bool, _Alloc > &__b) const`

5.484.1 Detailed Description

```
template<typename _Alloc> struct std::hash< __profile::vector< bool, _Alloc > >
```

[std::hash](#) specialization for `vector<bool>`.

Definition at line 507 of file `profile/vector`.

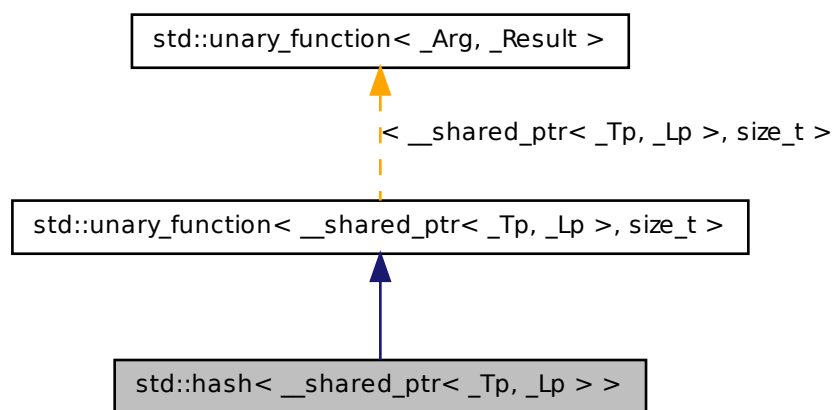
The documentation for this struct was generated from the following file:

- [profile/vector](#)

5.485 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference

[std::hash](#) specialization for `__shared_ptr`.

Inheritance diagram for `std::hash< __shared_ptr< _Tp, _Lp > >`:



Public Types

- typedef `__shared_ptr< _Tp, _Lp >` [argument_type](#)
- typedef `size_t` [result_type](#)

Public Member Functions

- `size_t operator() (const __shared_ptr< _Tp, _Lp > &__s) const`

5.485.1 Detailed Description

```
template<typename _Tp, _Lock_policy _Lp> struct std::hash< __shared_ptr<
_Tp, _Lp > >
```

[std::hash](#) specialization for `__shared_ptr`.

Definition at line 1189 of file `shared_ptr_base.h`.

5.485.2 Member Typedef Documentation

5.485.2.1 `typedef __shared_ptr< _Tp, _Lp > std::unary_function<
__shared_ptr< _Tp, _Lp > , size_t >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.485.2.2 `typedef size_t std::unary_function< __shared_ptr< _Tp, _Lp > ,
size_t >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr_base.h](#)

5.486 std::hash< _Tp * > Struct Template Reference

Partial specializations for pointer types.

Inherits `std::__hash_base< size_t, _Tp * >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (_Tp *__p) const`

5.486.1 Detailed Description

`template<typename _Tp> struct std::hash< _Tp * >`

Partial specializations for pointer types.

Definition at line 64 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

5.487 `std::hash< error_code >` Struct Template Reference

[std::hash](#) specialization for [error_code](#).

Inherits `std::__hash_base< size_t, error_code >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator() (const error_code &__e) const`

5.487.1 Detailed Description

`template<> struct std::hash< error_code >`

[std::hash](#) specialization for [error_code](#).

Definition at line 355 of file `system_error`.

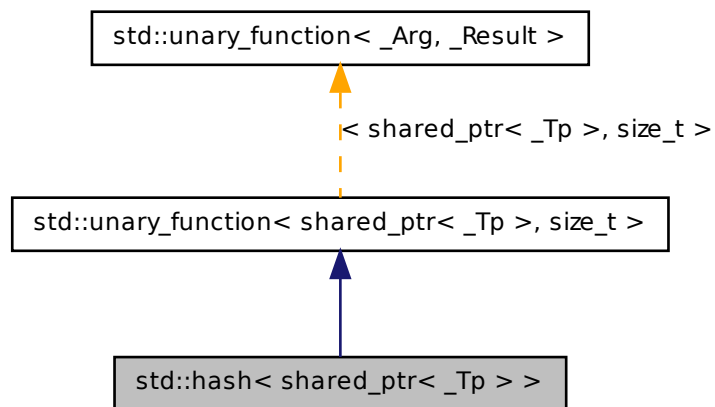
The documentation for this struct was generated from the following file:

- [system_error](#)

5.488 `std::hash< shared_ptr< _Tp > >` Struct Template Reference

[std::hash](#) specialization for [shared_ptr](#).

Inheritance diagram for std::hash< shared_ptr< _Tp > >:



Public Types

- typedef `shared_ptr< _Tp >` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator()` (`const shared_ptr< _Tp > &__s`) `const`

5.488.1 Detailed Description

`template<typename _Tp> struct std::hash< shared_ptr< _Tp > >`

`std::hash` specialization for `shared_ptr`.

Definition at line 544 of file `shared_ptr.h`.

5.488.2 Member Typedef Documentation

5.488.2.1 `typedef shared_ptr< _Tp > std::unary_function< shared_ptr< _Tp >, size_t>::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.488.2.2 `typedef size_t std::unary_function< shared_ptr< _Tp >, size_t>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

5.489 std::hash< string > Struct Template Reference

[std::hash](#) specialization for string.

Inherits `std::__hash_base< size_t, string >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (const string &__s) const`

5.489.1 Detailed Description

`template<> struct std::hash< string >`

[std::hash](#) specialization for string.

Definition at line 2927 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

5.490 `std::hash< thread::id >` Struct Template Reference

[std::hash](#) specialization for [thread::id](#).

Inherits `std::__hash_base< size_t, thread::id >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator() (const thread::id &__id) const`

5.490.1 Detailed Description

`template<> struct std::hash< thread::id >`

[std::hash](#) specialization for [thread::id](#).

Definition at line 221 of file `thread`.

The documentation for this struct was generated from the following file:

- [thread](#)

5.491 `std::hash< u16string >` Struct Template Reference

[std::hash](#) specialization for `u16string`.

Inherits `std::__hash_base< size_t, u16string >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator() (const u16string &__s) const`

5.491.1 Detailed Description

`template<> struct std::hash< u16string >`

[std::hash](#) specialization for [u16string](#).

Definition at line 2952 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

5.492 `std::hash< u32string >` Struct Template Reference

[std::hash](#) specialization for [u32string](#).

Inherits `std::__hash_base< size_t, u32string >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (const u32string &__s) const`

5.492.1 Detailed Description

`template<> struct std::hash< u32string >`

[std::hash](#) specialization for [u32string](#).

Definition at line 2963 of file `basic_string.h`.

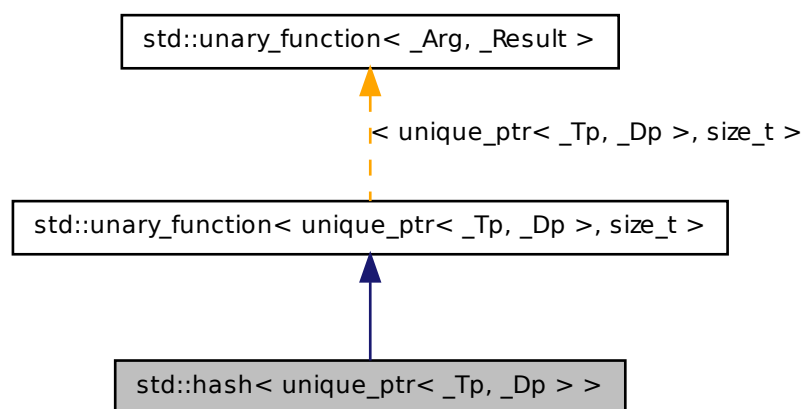
The documentation for this struct was generated from the following file:

- [basic_string.h](#)

5.493 `std::hash< unique_ptr< _Tp, _Dp > >` Struct Template Reference

`std::hash` specialization for `unique_ptr`.

Inheritance diagram for `std::hash< unique_ptr< _Tp, _Dp > >`:



Public Types

- typedef `unique_ptr< _Tp, _Dp >` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator() (const unique_ptr< _Tp, _Dp > &__u) const`

5.493.1 Detailed Description

```
template<typename _Tp, typename _Dp> struct std::hash< unique_ptr< _Tp,
_Dp > >
```

`std::hash` specialization for `unique_ptr`.

Definition at line 492 of file `unique_ptr.h`.

5.493.2 Member Typedef Documentation

5.493.2.1 `typedef unique_ptr< _Tp, _Dp > std::unary_function< unique_ptr< _Tp, _Dp >, size_t>::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.493.2.2 `typedef size_t std::unary_function< unique_ptr< _Tp, _Dp >, size_t>::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

5.494 std::hash< wstring > Struct Template Reference

[std::hash](#) specialization for `wstring`.

Inherits `std::__hash_base< size_t, wstring >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (const wstring &__s) const`

5.494.1 Detailed Description

`template<> struct std::hash< wstring >`

[std::hash](#) specialization for `wstring`.

Definition at line 2938 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

5.495 `std::hash<::bitset< _Nb > >` Struct Template Reference

[std::hash](#) specialization for `bitset`.

Inherits `std::__hash_base< size_t, ::bitset< _Nb > >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator() (const ::bitset< _Nb > &__b) const`

5.495.1 Detailed Description

`template<size_t _Nb> struct std::hash<::bitset< _Nb > >`

[std::hash](#) specialization for `bitset`.

Definition at line 1497 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

5.496 `std::hash<::vector< bool, _Alloc > >` Struct Template Reference

[std::hash](#) specialization for `vector<bool>`.

Inherits `std::__hash_base< size_t, ::vector< bool, _Alloc > >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (const `::vector`< `bool`, `_Alloc` > &__b) const

5.496.1 Detailed Description

`template<typename _Alloc> struct std::hash<::vector< bool, _Alloc > >`

`std::hash` specialization for `vector<bool>`.

Definition at line 1040 of file `stl_bvector.h`.

The documentation for this struct was generated from the following files:

- `stl_bvector.h`
- `vector.tcc`

5.497 `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >` Class Template Reference

Public Types

- typedef `_UIntType` **result_type**

Public Member Functions

- `independent_bits_engine` ()
- `independent_bits_engine` (const `_RandomNumberEngine` &__rne)
- `independent_bits_engine` (**result_type** __s)
- `template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`
`independent_bits_engine` (_Sseq &__q)
- `independent_bits_engine` (_RandomNumberEngine &&__rne)
- const `_RandomNumberEngine` & **base** () const

- void [discard](#) (unsigned long long __z)
- [result_type max](#) () const
- [result_type min](#) () const
- [result_type operator\(\)](#) ()
- void [seed](#) ([result_type](#) __s)
- template<typename _Sseq >
void [seed](#) (_Sseq &__q)
- void [seed](#) ()

Friends

- bool [operator==](#) (const [independent_bits_engine](#) &__lhs, const [independent_bits_engine](#) &__rhs)
- template<typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &__is, [std::independent_bits_engine](#)< _RandomNumberEngine, __w, _UIntType > &__x)

5.497.1 Detailed Description

template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>
>

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits __w.

Definition at line 1013 of file random.h.

5.497.2 Member Typedef Documentation

5.497.2.1 **template<typename _RandomNumberEngine, size_t __w, typename _UIntType> typedef _UIntType std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::result_type**

The type of the generated random value.

Definition at line 1022 of file random.h.

5.497.3 Constructor & Destructor Documentation

5.497.3.1 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> std::independent_bits_engine<_
RandomNumberEngine, __w, _UIntType >::independent_bits_engine
() [inline]`

Constructs a default independent_bits_engine engine.

The underlying engine is default constructed as well.

Definition at line 1029 of file random.h.

5.497.3.2 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> std::independent_bits_engine<_
_RandomNumberEngine, __w, _UIntType >::independent_bits_
engine (const _RandomNumberEngine & __rne) [inline,
explicit]`

Copy constructs a independent_bits_engine engine.

Copies an existing base class random number generator.

Parameters

rne An existing (base class) engine object.

Definition at line 1039 of file random.h.

5.497.3.3 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> std::independent_bits_engine<_
RandomNumberEngine, __w, _UIntType >::independent_bits_engine
(_RandomNumberEngine && __rne) [inline, explicit]`

Move constructs a independent_bits_engine engine.

Copies an existing base class random number generator.

Parameters

rne An existing (base class) engine object.

Definition at line 1049 of file random.h.

5.497.3.4 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> std::independent_bits_engine<_-
RandomNumberEngine, __w, _UIntType >::independent_bits_engine
(result_type __s) [inline, explicit]`

Seed constructs a independent_bits_engine engine.

Constructs the underlying generator engine seeded with __s.

Parameters

__s A seed value for the base class engine.

Definition at line 1059 of file random.h.

5.497.3.5 `template<typename _RandomNumberEngine, size_t
__w, typename _UIntType> template<typename _Sseq ,
typename = typename std::enable_if<!std::is_same<_Sseq,
independent_bits_engine>::value && !std::is_same<_Sseq,
RandomNumberEngine>::value> ::type> std::independent
bits_engine<_RandomNumberEngine, __w, _UIntType
>::independent_bits_engine (_Sseq & __q) [inline,
explicit]`

Generator construct a independent_bits_engine engine.

Parameters

__q A seed sequence.

Definition at line 1072 of file random.h.

5.497.4 Member Function Documentation

5.497.4.1 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> const _RandomNumberEngine&
std::independent_bits_engine<_RandomNumberEngine, __w,
_UIntType >::base () const [inline]`

Gets a const reference to the underlying generator engine object.

Definition at line 1107 of file random.h.

5.497.4.2 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::discard (unsigned
long long __z) [inline]`

Discard a sequence of random numbers.

Todo

Look for a faster way to do discard.

Definition at line 1134 of file random.h.

5.497.4.3 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::max () const
[inline]`

Gets the maximum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 1125 of file random.h.

5.497.4.4 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::min () const
[inline]`

Gets the minimum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 1116 of file random.h.

5.497.4.5 `template<typename _RandomNumberEngine , size_t __w, typename
_UIntType > independent_bits_engine< _RandomNumberEngine,
__w, _UIntType >::result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::operator() ()`

Gets the next value in the generated random number sequence.

**5.497 std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType
> Class Template Reference** **2539**

Definition at line 722 of file random.tcc.

References std::log().

5.497.4.6 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> template<typename _Sseq > void
std::independent_bits_engine<_RandomNumberEngine, __w,
_UIntType >::seed (_Sseq & __q) [inline]`

Reseeds the independent_bits_engine object with the given seed sequence.

Parameters

`__q` A seed generator function.

Definition at line 1099 of file random.h.

5.497.4.7 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::seed (result_type __s
) [inline]`

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

Definition at line 1089 of file random.h.

5.497.4.8 `template<typename _RandomNumberEngine, size_t __w,
typename _UIntType> void std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::seed () [inline]`

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

Definition at line 1081 of file random.h.

5.497.5 Friends And Related Function Documentation

5.497.5.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool operator==(const independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __lhs, const independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __rhs) [friend]`

Compares two `independent_bits_engine` random number generator objects of the same type for equality.

Parameters

`__lhs` A `independent_bits_engine` random number generator object.

`__rhs` Another `independent_bits_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1159 of file `random.h`.

5.497.5.2 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits> & operator>>(std::basic_istream<_CharT, _Traits> & __is, std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __x) [friend]`

Extracts the current state of a `% subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `independent_bits_engine` random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

Definition at line 1177 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.498 `std::indirect_array<_Tp>` Class Template Reference

Reference to arbitrary subset of an array.

Public Types

- `typedef _Tp value_type`

Public Member Functions

- `indirect_array` (const `indirect_array` &)
- `template<class _Dom >`
void **`operator%=`** (const `_Expr<_Dom, _Tp>` &) const
- void `operator%=` (const `valarray<_Tp>` &) const
- void `operator&=` (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **`operator&=`** (const `_Expr<_Dom, _Tp>` &) const
- `template<class _Dom >`
void **`operator*=`** (const `_Expr<_Dom, _Tp>` &) const
- void `operator*=` (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **`operator+=`** (const `_Expr<_Dom, _Tp>` &) const
- void `operator+=` (const `valarray<_Tp>` &) const
- void `operator-=` (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **`operator-=`** (const `_Expr<_Dom, _Tp>` &) const
- `template<class _Dom >`
void **`operator/=`** (const `_Expr<_Dom, _Tp>` &) const
- void `operator/=` (const `valarray<_Tp>` &) const
- void `operator<=<=` (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **`operator<=<=`** (const `_Expr<_Dom, _Tp>` &) const
- void `operator=` (const `_Tp` &) const
- `indirect_array` & `operator=` (const `indirect_array` &)
- `template<class _Dom >`
void **`operator=`** (const `_Expr<_Dom, _Tp>` &) const
- void `operator=` (const `valarray<_Tp>` &) const
- `template<class _Dom >`
void **`operator>>=`** (const `_Expr<_Dom, _Tp>` &) const
- void `operator>>=` (const `valarray<_Tp>` &) const

- `template<class _Dom >`
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator|= (const _Expr< _Dom, _Tp > &) const`

Friends

- `class gslice_array< _Tp >`
- `class valarray< _Tp >`

5.498.1 Detailed Description

`template<class _Tp> class std::indirect_array< _Tp >`

Reference to arbitrary subset of an array. An `indirect_array` is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an `indirect_array` is to call `operator[]`(`valarray<size_t>`) on a `valarray`. The returned `indirect_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

Parameters

Tp Element type.

Definition at line 61 of file `indirect_array.h`.

5.498.2 Member Function Documentation

5.498.2.1 `template<class _Tp> void std::indirect_array< _Tp >::operator%= (const valarray< _Tp > &) const`

Modulo slice elements by corresponding elements of *v*.

5.498.2.2 `template<class _Tp> void std::indirect_array< _Tp >::operator&= (const valarray< _Tp > &) const`

Logical and slice elements with corresponding elements of *v*.

5.498.2.3 `template<class _Tp> void std::indirect_array< _Tp >::operator*= (const valarray< _Tp > &) const`

Multiply slice elements by corresponding elements of *v*.

5.498.2.4 `template<class _Tp> void std::indirect_array< _Tp >::operator+= (const valarray< _Tp > &) const`

Add corresponding elements of *v* to slice elements.

5.498.2.5 `template<class _Tp> void std::indirect_array< _Tp >::operator-= (const valarray< _Tp > &) const`

Subtract corresponding elements of *v* from slice elements.

5.498.2.6 `template<class _Tp> void std::indirect_array< _Tp >::operator/= (const valarray< _Tp > &) const`

Divide slice elements by corresponding elements of *v*.

5.498.2.7 `template<class _Tp> void std::indirect_array< _Tp >::operator<<= (const valarray< _Tp > &) const`

Left shift slice elements by corresponding elements of *v*.

5.498.2.8 `template<class _Tp> void std::indirect_array< _Tp >::operator>>= (const valarray< _Tp > &) const`

Right shift slice elements by corresponding elements of *v*.

5.498.2.9 `template<class _Tp> void std::indirect_array< _Tp >::operator^= (const valarray< _Tp > &) const`

Logical xor slice elements with corresponding elements of *v*.

5.498.2.10 `template<class _Tp> void std::indirect_array< _Tp >::operator|= (const valarray< _Tp > &) const`

Logical or slice elements with corresponding elements of *v*.

The documentation for this class was generated from the following file:

- [indirect_array.h](#)

5.499 `std::initializer_list<_E>` Class Template Reference

[initializer_list](#)

Public Types

- typedef const _E * **const_iterator**
- typedef const _E & **const_reference**
- typedef const _E * **iterator**
- typedef const _E & **reference**
- typedef size_t **size_type**
- typedef _E **value_type**

Public Member Functions

- const_iterator **begin** () const
- const_iterator **end** () const
- size_type **size** () const

5.499.1 Detailed Description

`template<class _E> class std::initializer_list<_E>`

[initializer_list](#)

Definition at line 45 of file `initializer_list`.

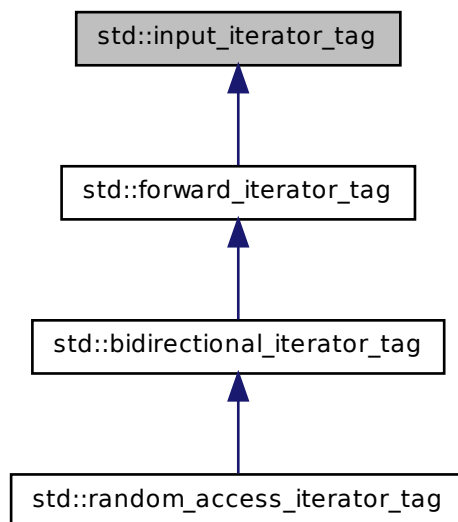
The documentation for this class was generated from the following file:

- [initializer_list](#)

5.500 `std::input_iterator_tag` Struct Reference

Marking input iterators.

Inheritance diagram for `std::input_iterator_tag`:



5.500.1 Detailed Description

Marking input iterators.

Definition at line 88 of file `stl_iterator_base_types.h`.

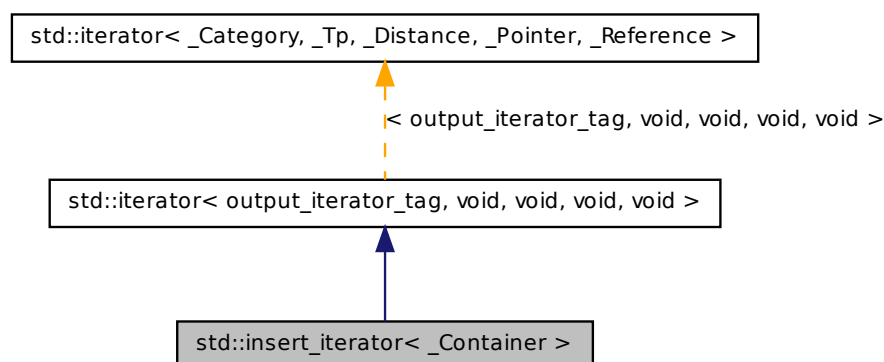
The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.501 `std::insert_iterator<_Container>` Class Template Reference

Turns assignment into insertion.

Inheritance diagram for `std::insert_iterator< _Container >`:



Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

Public Member Functions

- `insert_iterator` (`_Container &__x`, `typename _Container::iterator __i`)
- `insert_iterator & operator*` ()
- `insert_iterator & operator++` ()
- `insert_iterator & operator++` (int)
- `insert_iterator & operator=` (const `typename _Container::value_type &__value`)
- `insert_iterator & operator=` (`typename _Container::value_type &&__value`)

Protected Attributes

- `_Container * container`
- `_Container::iterator iter`

5.501.1 Detailed Description

template<typename _Container> class std::insert_iterator< _Container >

Turns assignment into insertion. These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 579 of file stl_iterator.h.

5.501.2 Member Typedef Documentation

**5.501.2.1 template<typename _Container> typedef _Container
 std::insert_iterator< _Container >::container_type**

A nested typedef for the type of whatever container you used.

Definition at line 588 of file stl_iterator.h.

**5.501.2.2 typedef void std::iterator< output_iterator_tag , void , void , void ,
 void >::difference_type [inherited]**

Distance between iterators is represented as this type.

Definition at line 124 of file stl_iterator_base_types.h.

**5.501.2.3 typedef output_iterator_tag std::iterator< output_iterator_tag , void
 , void , void , void >::iterator_category [inherited]**

One of the [tag types](#).

Definition at line 120 of file stl_iterator_base_types.h.

**5.501.2.4 typedef void std::iterator< output_iterator_tag , void , void , void ,
 void >::pointer [inherited]**

This type represents a pointer-to-value_type.

Definition at line 126 of file stl_iterator_base_types.h.

5.501.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 128 of file stl_iterator_base_types.h.

5.501.2.6 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file stl_iterator_base_types.h.

5.501.3 Constructor & Destructor Documentation

5.501.3.1 `template<typename _Container> std::insert_iterator< _Container >::insert_iterator (_Container & __x, typename _Container::iterator __i) [inline]`

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 594 of file stl_iterator.h.

5.501.4 Member Function Documentation

5.501.4.1 `template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator* () [inline]`

Simply returns *this.

Definition at line 648 of file stl_iterator.h.

5.501.4.2 `template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator++ (int) [inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 658 of file stl_iterator.h.

5.501.4.3 `template<typename _Container> insert_iterator&
std::insert_iterator< _Container >::operator++ () [inline]`

Simply returns *this. (This iterator does not *move*.).

Definition at line 653 of file stl_iterator.h.

5.501.4.4 `template<typename _Container> insert_iterator&
std::insert_iterator< _Container >::operator= (const typename
_Container::value_type & __value) [inline]`

Parameters

value An instance of whatever type `container_type::const_reference` is; presumably a reference-to-const T for `container<T>`.

Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z  
  
insert_iterator i (v, ++v.begin());  
i = 1;  
i = 2;  
i = 3;  
  
// vector v contains A, 1, 2, 3, and Z
```

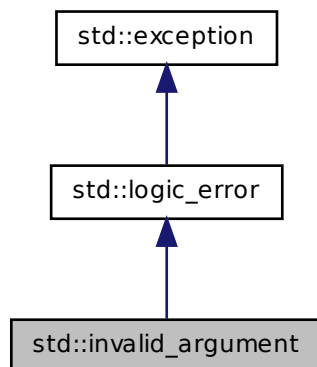
Definition at line 630 of file stl_iterator.h.

The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

5.502 std::invalid_argument Class Reference

Inheritance diagram for std::invalid_argument:



Public Member Functions

- **invalid_argument** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.502.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 80 of file stdexcept.

5.502.2 Member Function Documentation

5.502.2.1 virtual const char* std::logic_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

The documentation for this class was generated from the following file:

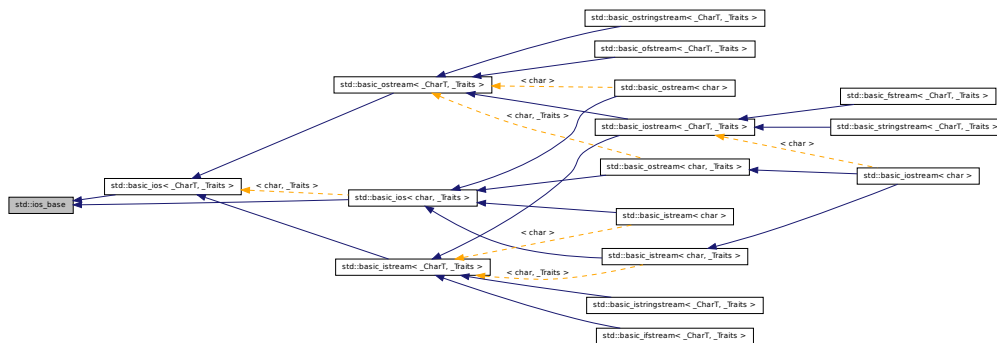
- [stdexcept](#)

5.503 std::ios_base Class Reference

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Inheritance diagram for `std::ios_base`:



Classes

- class [failure](#)

These are thrown to indicate problems with io.
 27.4.2.1.1 Class `ios_base::failure`.

Public Types

- enum [event](#) { `erase_event`, `imbue_event`, `copyfmt_event` }
- typedef void(* [event_callback](#))(event, `ios_base` &, int)
- typedef `_Ios_Fmtflags` [fmtflags](#)
- typedef int `io_state`

- typedef `_Ios_Iostate` [iostate](#)
- typedef int `open_mode`
- typedef `_Ios_Openmode` [openmode](#)
- typedef int `seek_dir`
- typedef `_Ios_Seekdir` [seekdir](#)
- typedef [std::streamoff](#) `streamoff`
- typedef [std::streampos](#) `streampos`

Public Member Functions

- virtual `~ios_base` ()
- const `locale` & `_M_getloc` () const
- `fmtflags` `flags` () const
- `fmtflags` `flags` (`fmtflags` __fmtfl)
- `locale` `getloc` () const
- `locale imbue` (const `locale` & __loc) throw ()
- long & `word` (int __ix)
- `streamsize` `precision` (`streamsize` __prec)
- `streamsize` `precision` () const
- void *& `pword` (int __ix)
- void `register_callback` (`event_callback` __fn, int __index)
- `fmtflags` `setf` (`fmtflags` __fmtfl)
- `fmtflags` `setf` (`fmtflags` __fmtfl, `fmtflags` __mask)
- void `unsetf` (`fmtflags` __mask)
- `streamsize` `width` () const
- `streamsize` `width` (`streamsize` __wide)

Static Public Member Functions

- static bool `sync_with_stdio` (bool __sync=true)
- static int `xalloc` () throw ()

Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iostate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`

- static const [fmtflags boolalpha](#)
- static const [seekdir cur](#)
- static const [fmtflags dec](#)
- static const [seekdir end](#)
- static const [iostate eofbit](#)
- static const [iostate failbit](#)
- static const [fmtflags fixed](#)
- static const [fmtflags floatfield](#)
- static const [iostate goodbit](#)
- static const [fmtflags hex](#)
- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- void [_M_call_callbacks](#) ([event __ev](#)) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- [_Words & _M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()

Protected Attributes

- `_Callback_list * _M_callbacks`
- `iosstate _M_exception`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `streamsize _M_precision`
- `iosstate _M_streambuf_state`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

5.503.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 198 of file `ios_base.h`.

5.503.2 Member Typedef Documentation

5.503.2.1 `typedef void(* std::ios_base::event_callback)(event, ios_base &, int)`

The type of an event callback function.

Parameters

event One of the members of the event enum.

ios_base Reference to the `ios_base` object.

int The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 435 of file `ios_base.h`.

5.503.2.2 typedef _Ios_Fmtflags std::ios_base::fmtflags

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 254 of file `ios_base.h`.

5.503.2.3 `typedef _Ios_Iostate std::ios_base::iostate`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 329 of file `ios_base.h`.

5.503.2.4 `typedef _Ios_Openmode std::ios_base::openmode`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 360 of file `ios_base.h`.

5.503.2.5 `typedef _Ios_Seekdir std::ios_base::seekdir`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.

- end, equivalent to SEEK_END in the C standard library.

Definition at line 392 of file ios_base.h.

5.503.3 Member Enumeration Documentation

5.503.3.1 enum std::ios_base::event

The set of events that may be passed to an event callback.

erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 418 of file ios_base.h.

5.503.4 Constructor & Destructor Documentation

5.503.4.1 virtual std::ios_base::~ios_base () [virtual]

Invokes each callback with erase_event. Destroys local storage.

Note that the ios_base object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with erase_event (unless copyfmt is used).

5.503.5 Member Function Documentation

5.503.5.1 const locale& std::ios_base::_M_getloc () const [inline]

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 705 of file ios_base.h.

Referenced by std::money_get<_CharT, _InIter>::do_get(), std::num_get<_CharT, _InIter>::do_get(), std::time_get<_CharT, _InIter>::do_get_date(), std::time_get<_CharT, _InIter>::do_get_monthname(), std::time_get<_CharT, _InIter>::do_get_time(), std::time_get<_CharT, _InIter>::do_get_weekday(), std::time_get<_CharT, _InIter>::do_get_year(), std::time_put<_CharT, _OutIter>::do_put(), std::num_put<_CharT, _OutIter>::do_put(), and std::time_put<_CharT, _OutIter>::put().

5.503.5.2 fmtflags std::ios_base::flags () const [inline]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 550 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), and std::operator>>().

5.503.5.3 fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline]

Setting new format flags all at once.

Parameters

fmtfl The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *fmtfl*.

Definition at line 561 of file ios_base.h.

5.503.5.4 locale std::ios_base::getloc () const [inline]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 694 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Traits >::imbue(), std::operator>>(), and std::ws().

5.503.5.5 locale std::ios_base::imbue (const locale & __loc) throw ()

Setting a new locale.

Parameters

loc The new locale.

Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with imbue_event.

Reimplemented in [std::basic_ios< _CharT, _Traits >](#), and [std::basic_ios< char, _Traits >](#).

5.503.5.6 long& std::ios_base::iword (int __ix) [inline]

Access to integer array.

Parameters

__ix Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 740 of file ios_base.h.

5.503.5.7 streamsize std::ios_base::precision () const [inline]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 620 of file ios_base.h.

Referenced by std::basic_ios<_CharT, _Traits>::copyfmt(), and std::operator<<().

5.503.5.8 streamsize std::ios_base::precision (streamsize __prec) [inline]

Changing flags.

Parameters

prec The new precision value.

Returns

The previous value of [precision\(\)](#).

Definition at line 629 of file ios_base.h.

5.503.5.9 void*& std::ios_base::pword (int __ix) [inline]

Access to void pointer array.

Parameters

__ix Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 761 of file ios_base.h.

5.503.5.10 void std::ios_base::register_callback (event_callback __fn, int __index)

Add the callback __fn with parameter __index.

Parameters

- __fn* The function to add.
__index The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**5.503.5.11 fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask)
[inline]**

Setting new format flags.

Parameters

- fmtfl* Additional flags to set.
mask The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 594 of file `ios_base.h`.

5.503.5.12 fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline]

Setting new format flags.

Parameters

- fmtfl* Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 577 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

5.503.5.13 `static bool std::ios_base::sync_with_stdio (bool __sync = true) [static]`

Interaction with the standard C I/O objects.

Parameters

sync Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch28s02.html>

5.503.5.14 `void std::ios_base::unsetf (fmtflags __mask) [inline]`

Clearing format flags.

Parameters

mask The flags to unset.

This function clears *mask* in the format flags.

Definition at line 609 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

5.503.5.15 `streamsize std::ios_base::width () const [inline]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 643 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::copyfmt()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::operator>>()`.

5.503.5.16 streamsize std::ios_base::width (streamsize __*wide*) [inline]

Changing flags.

Parameters

wide The new width value.

Returns

The previous value of [width\(\)](#).

Definition at line 652 of file ios_base.h.

5.503.5.17 static int std::ios_base::xalloc () throw () [static]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

5.503.6 Member Data Documentation**5.503.6.1** const fmtflags std::ios_base::adjustfield [static]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 309 of file ios_base.h.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

5.503.6.2 const openmode std::ios_base::app [static]

Seek to end before each write.

Definition at line 363 of file ios_base.h.

5.503.6.3 const openmode std::ios_base::ate [static]

Open and seek to end immediately after opening.

Definition at line 366 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.503.6.4 const iostate std::ios_base::badbit [static]

Indicates a loss of integrity in an input or output sequence (such /// as an irrecoverable read error from a file).

Definition at line 333 of file ios_base.h.

Referenced by std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::operator<<(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_istream< _CharT, _Traits >::unget().

5.503.6.5 const fmtflags std::ios_base::basefield [static]

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 312 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.503.6.6 const seekdir std::ios_base::beg [static]

Request a seek relative to the beginning of the stream.

Definition at line 395 of file ios_base.h.

5.503.6.7 const openmode std::ios_base::binary [static]

Perform input and output in binary mode (as opposed to text mode). /// This is probably not what you think it is; see ///

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 371 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

5.503.6.8 const fmtflags std::ios_base::boolalpha [static]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 257 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), and std::num_put< _CharT, _OutIter >::do_put().

5.503.6.9 const seekdir std::ios_base::cur [static]

Request a seek relative to the current position within the sequence.

Definition at line 398 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

5.503.6.10 const fmtflags std::ios_base::dec [static]

Converts integer input or generates integer output in decimal base.

Definition at line 260 of file ios_base.h.

5.503.6.11 const seekdir std::ios_base::end [static]

Request a seek relative to the current end of the sequence.

Definition at line 401 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

5.503.6.12 const iostate std::ios_base::eofbit [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 336 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter

>::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), and std::ws().

5.503.6.13 `const iostate std::ios_base::failbit` **[static]**

Indicates that an input operation failed to read the expected /// characters, or that an output operation failed to generate the /// desired characters.

Definition at line 341 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), and std::basic_ostream< _CharT, _Traits >::seekp().

5.503.6.14 `const fmtflags std::ios_base::fixed` **[static]**

Generate floating-point output in fixed-point notation.

Definition at line 263 of file ios_base.h.

5.503.6.15 `const fmtflags std::ios_base::floatfield` **[static]**

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 315 of file ios_base.h.

5.503.6.16 `const iostate std::ios_base::goodbit` **[static]**

Indicates all is well.

Definition at line 344 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),

std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ios< _CharT, _Traits >::init(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

5.503.6.17 const fmtflags std::ios_base::hex [static]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 266 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::basic_ostream< _CharT, _Traits >::operator<<().

5.503.6.18 const openmode std::ios_base::in [static]

Open for input. Default for ifstream and fstream.

Definition at line 374 of file ios_base.h.

Referenced by std::basic_istream< _CharT, _Traits >::seekg(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::underflow().

5.503.6.19 const fmtflags std::ios_base::internal [static]

Adds fill characters at a designated internal point in certain /// generated output, or identical to right if no such point is /// designated.

Definition at line 271 of file ios_base.h.

5.503.6.20 const fmtflags std::ios_base::left [static]

Adds fill characters on the right (final positions) of certain /// generated output. (I.e., the thing you print is flush left.).

Definition at line 275 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put().

5.503.6.21 `const fmtflags std::ios_base::oct [static]`

Converts integer input or generates integer output in octal base.

Definition at line 278 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::operator<<()`.

5.503.6.22 `const openmode std::ios_base::out [static]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 377 of file `ios_base.h`.

Referenced by `std::basic_ostream< _CharT, _Traits >::seekp()`, and `std::basic_ostream< _CharT, _Traits >::tellp()`.

5.503.6.23 `const fmtflags std::ios_base::right [static]`

Adds fill characters on the left (initial positions) of certain /// generated output. (I.e., the thing you print is flush right.).

Definition at line 282 of file `ios_base.h`.

5.503.6.24 `const fmtflags std::ios_base::scientific [static]`

Generates floating-point output in scientific notation.

Definition at line 285 of file `ios_base.h`.

5.503.6.25 `const fmtflags std::ios_base::showbase [static]`

Generates a prefix indicating the numeric base of generated integer /// output.

Definition at line 289 of file `ios_base.h`.

5.503.6.26 `const fmtflags std::ios_base::showpoint [static]`

Generates a decimal-point character unconditionally in generated /// floating-point output.

Definition at line 293 of file `ios_base.h`.

5.503.6.27 `const fmtflags std::ios_base::showpos [static]`

Generates a + sign in non-negative generated numeric output.

Definition at line 296 of file `ios_base.h`.

5.503.6.28 `const fmtflags std::ios_base::skipws` `[static]`

Skips leading white space before certain input operations.

Definition at line 299 of file `ios_base.h`.

5.503.6.29 `const openmode std::ios_base::trunc` `[static]`

Open for input. Default for `ofstream`.

Definition at line 380 of file `ios_base.h`.

5.503.6.30 `const fmtflags std::ios_base::unitbuf` `[static]`

Flushes output after each output operation.

Definition at line 302 of file `ios_base.h`.

5.503.6.31 `const fmtflags std::ios_base::uppercase` `[static]`

Replaces certain lowercase letters with their uppercase equivalents /// in generated output.

Definition at line 306 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`.

The documentation for this class was generated from the following file:

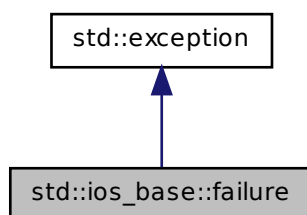
- [ios_base.h](#)

5.504 `std::ios_base::failure` Class Reference

These are thrown to indicate problems with io.

27.4.2.1.1 Class `ios_base::failure`.

Inheritance diagram for `std::ios_base::failure`:



Public Member Functions

- **failure** (const [string](#) &__str) throw ()
- virtual const char * **what** () const throw ()

5.504.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class [ios_base::failure](#).

Definition at line 208 of file `ios_base.h`.

5.504.2 Member Function Documentation

5.504.2.1 virtual const char* std::ios_base::failure::what () const throw () [virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [ios_base.h](#)

5.505 `std::is_base_of< _Base, _Derived >` Struct Template Reference

[is_base_of](#)

Inherits `integral_constant< bool, __is_base_of(_Base, _Derived)>`.

5.505.1 Detailed Description

```
template<typename _Base, typename _Derived> struct std::is_base_of< _Base,
_Derived >
```

[is_base_of](#)

Definition at line 305 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.506 `std::is_bind_expression< _Tp >` Struct Template Reference

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].

Inherits `false_type`.

5.506.1 Detailed Description

```
template<typename _Tp> struct std::is_bind_expression< _Tp >
```

Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].

Definition at line 822 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.507 `std::is_bind_expression< _Bind< _Signature > >` Struct Template Reference

Class template `_Bind` is always a bind expression.

Inherits `true_type`.

5.507.1 Detailed Description

```
template<typename _Signature> struct std::is_bind_expression< _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

Definition at line 1403 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.508 `std::is_bind_expression< _Bind_result< _Result, _Signature > >` Struct Template Reference

Class template `_Bind` is always a bind expression.

Inherits `true_type`.

5.508.1 Detailed Description

```
template<typename _Result, typename _Signature> struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
```

Class template `_Bind` is always a bind expression.

Definition at line 1411 of file `functional`.

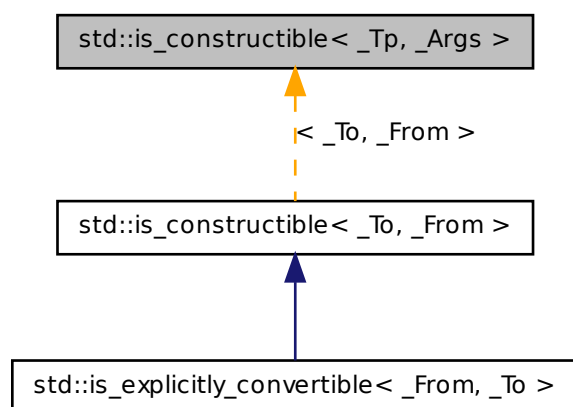
The documentation for this struct was generated from the following file:

- [functional](#)

5.509 std::is_constructible< _Tp, _Args > Struct Template Reference

[is_constructible](#)

Inheritance diagram for std::is_constructible< _Tp, _Args >:



5.509.1 Detailed Description

```
template<typename _Tp, typename... _Args> struct std::is_constructible< _Tp,  
_Args >
```

[is_constructible](#)

Definition at line 231 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.510 `std::is_convertible< _From, _To >` Struct Template Reference

[is_convertible](#)

Inherits `integral_constant< bool, __is_convertible_helper< _From, _To >::__value >`.

5.510.1 Detailed Description

```
template<typename _From, typename _To> struct std::is_convertible< _From,
_To >
```

[is_convertible](#)

Definition at line 337 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.511 `std::is_error_code_enum< _Tp >` Struct Template Reference

[is_error_code_enum](#)

Inherits `false_type`.

5.511.1 Detailed Description

```
template<typename _Tp> struct std::is_error_code_enum< _Tp >
```

[is_error_code_enum](#)

Definition at line 52 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system_error](#)

5.512 `std::is_error_condition_enum< _Tp >` Struct Template Reference

[is_error_condition_enum](#)

Inherits `false_type`.

5.512.1 Detailed Description

```
template<typename _Tp> struct std::is_error_condition_enum< _Tp >
```

[is_error_condition_enum](#)

Definition at line 56 of file `system_error`.

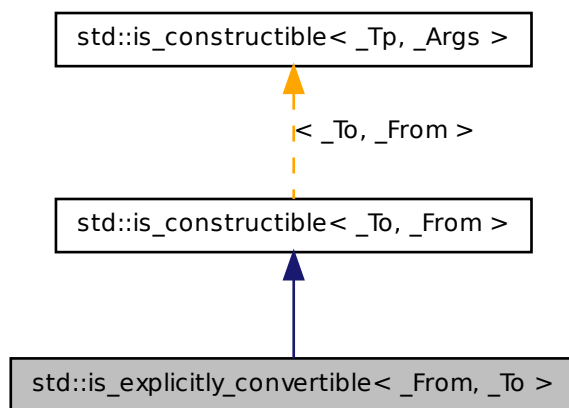
The documentation for this struct was generated from the following file:

- [system_error](#)

5.513 `std::is_explicitly_convertible< _From, _To >` Struct Template Reference

[is_explicitly_convertible](#)

Inheritance diagram for `std::is_explicitly_convertible<_From, _To>`:



5.513.1 Detailed Description

```
template<typename _From,   typename _To> struct std::is_explicitly_
convertible<_From, _To>
```

[is_explicitly_convertible](#)

Definition at line 344 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.514 `std::is_lvalue_reference< typename >` Struct Template Reference

[is_lvalue_reference](#)

Inherits `false_type`.

5.515 `std::is_nothrow_constructible< _Tp, _Args >` Struct Template Reference 2577

5.514.1 Detailed Description

`template<typename> struct std::is_lvalue_reference< typename >`

[is_lvalue_reference](#)

Definition at line 69 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.515 `std::is_nothrow_constructible< _Tp, _Args >` Struct Template Reference

[is_nothrow_constructible](#)

Inherits `integral_constant< bool, __is_nt_constructible_helper< is_constructible< _Tp, _Args...>::value, _Tp, _Args...>::__value >`.

5.515.1 Detailed Description

`template<typename _Tp, typename... _Args> struct std::is_nothrow_constructible< _Tp, _Args >`

[is_nothrow_constructible](#)

Definition at line 253 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.516 `std::is_placeholder< _Tp >` Struct Template Reference

Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].

Inherits `integral_constant< int, 0 >`.

5.516.1 Detailed Description

template<typename _Tp> struct std::is_placeholder< _Tp >

Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].

Definition at line 831 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.517 `std::is_placeholder< _Placeholder< _Num > >` Struct Template Reference

Inherits `integral_constant< int, _Num >`.

5.517.1 Detailed Description

template<int _Num> struct std::is_placeholder< _Placeholder< _Num > >

Partial specialization of [is_placeholder](#) that provides the placeholder number for the placeholder objects defined by `libstdc++`.

Definition at line 888 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

5.518 `std::is_pod< _Tp >` Struct Template Reference

[is_pod](#)

Inherits `integral_constant< bool, __is_pod(_Tp)>`.

5.518.1 Detailed Description

template<typename _Tp> struct std::is_pod< _Tp >

[is_pod](#)

Definition at line 191 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.519 `std::is_reference< _Tp >` Struct Template Reference

[is_reference](#)

Inherits `integral_constant< bool,(is_lvalue_reference< _Tp >::value||is_rvalue_reference< _Tp >::value)>`.

5.519.1 Detailed Description

```
template<typename _Tp> struct std::is_reference< _Tp >
```

[is_reference](#)

Definition at line 89 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.520 `std::is_rvalue_reference< typename >` Struct Template Reference

[is_rvalue_reference](#)

Inherits `false_type`.

5.520.1 Detailed Description

```
template<typename> struct std::is_rvalue_reference< typename >
```

[is_rvalue_reference](#)

Definition at line 78 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.521 `std::is_signed< _Tp >` Struct Template Reference

[is_signed](#)

Inherits `integral_constant< bool, __is_signed_helper< _Tp >::value >`.

5.521.1 Detailed Description

```
template<typename _Tp> struct std::is_signed< _Tp >
```

[is_signed](#)

Definition at line 163 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.522 `std::is_standard_layout< _Tp >` Struct Template Reference

[is_standard_layout](#)

Inherits `integral_constant< bool, __is_standard_layout(_Tp)>`.

5.522.1 Detailed Description

```
template<typename _Tp> struct std::is_standard_layout< _Tp >
```

[is_standard_layout](#)

Definition at line 184 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.523 `std::is_trivial< _Tp >` Struct Template Reference

[is_trivial](#)

Inherits `integral_constant< bool, __is_trivial(_Tp)>`.

5.523.1 Detailed Description

```
template<typename _Tp> struct std::is_trivial< _Tp >
```

[is_trivial](#)

Definition at line 178 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.524 `std::is_unsigned< _Tp >` Struct Template Reference

[is_unsigned](#)

Inherits `integral_constant< bool,(is_arithmetic< _Tp >::value &&!is_signed< _Tp >::value)>`.

5.524.1 Detailed Description

```
template<typename _Tp> struct std::is_unsigned< _Tp >
```

[is_unsigned](#)

Definition at line 169 of file `type_traits`.

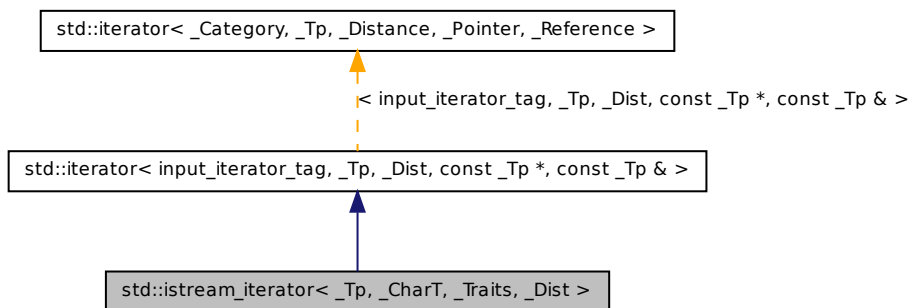
The documentation for this struct was generated from the following file:

- [type_traits](#)

5.525 `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >` Class Template Reference

Provides input iterator semantics for streams.

Inheritance diagram for `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Dist` **difference_type**
- typedef `basic_istream< _CharT, _Traits >` **istream_type**
- typedef `input_iterator_tag` **iterator_category**
- typedef `const _Tp *` **pointer**
- typedef `const _Tp &` **reference**
- typedef `_Traits` **traits_type**
- typedef `_Tp` **value_type**

Public Member Functions

- `istream_iterator()`
- `istream_iterator(istream_type &__s)`
- `istream_iterator(const istream_iterator &__obj)`
- `bool _M_equal(const istream_iterator &__x) const`
- `const _Tp & operator*() const`
- `istream_iterator & operator++()`
- `istream_iterator operator++(int)`
- `const _Tp * operator->() const`

5.525.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >
```

Provides input iterator semantics for streams.

Definition at line 47 of file stream_iterator.h.

5.525.2 Member Typedef Documentation

5.525.2.1 `typedef _Dist std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 124 of file stl_iterator_base_types.h.

5.525.2.2 `typedef input_iterator_tag std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 120 of file stl_iterator_base_types.h.

5.525.2.3 `typedef const _Tp * std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 126 of file stl_iterator_base_types.h.

5.525.2.4 `typedef const _Tp & std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 128 of file stl_iterator_base_types.h.

5.525.2.5 `typedef _Tp std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file `std_iterator_base_types.h`.

5.525.3 Constructor & Destructor Documentation

5.525.3.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator () [inline]`

Construct end of input stream iterator.

Definition at line 62 of file `stream_iterator.h`.

5.525.3.2 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (istream_type & __s) [inline]`

Construct start of input stream iterator.

Definition at line 66 of file `stream_iterator.h`.

The documentation for this class was generated from the following file:

- [stream_iterator.h](#)

5.526 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

Provides input iterator semantics for streambufs.

Inheritance diagram for `std::istreambuf_iterator< _CharT, _Traits >`:



Public Types

- `typedef _Traits::off_type difference_type`
- `typedef input_iterator_tag iterator_category`
- `typedef _CharT * pointer`

5.526 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference 2585

- typedef _CharT & [reference](#)
- typedef _CharT [value_type](#)
- typedef _CharT [char_type](#)
- typedef _Traits [traits_type](#)
- typedef _Traits::int_type [int_type](#)
- typedef [basic_streambuf](#)< _CharT, _Traits > [streambuf_type](#)
- typedef [basic_istream](#)< _CharT, _Traits > [istream_type](#)

Public Member Functions

- [istreambuf_iterator](#) () throw ()
- [istreambuf_iterator](#) ([istream_type](#) &__s) throw ()
- [istreambuf_iterator](#) ([streambuf_type](#) *__s) throw ()
- bool [equal](#) (const [istreambuf_iterator](#) &__b) const
- [char_type](#) [operator*](#) () const
- [istreambuf_iterator](#) [operator++](#) (int)
- [istreambuf_iterator](#) & [operator++](#) ()

Friends

- template<bool _IsMove, typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__
type __[copy_move_a2](#) ([istreambuf_iterator](#)< _CharT2 >, [istreambuf_iterator](#)<
_CharT2 >, _CharT2 *)
- template<typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, [ostreambuf_
iterator](#)< _CharT2 > >::__type [copy](#) ([istreambuf_iterator](#)< _CharT2 >,
[istreambuf_iterator](#)< _CharT2 >, [ostreambuf_iterator](#)< _CharT2 >)
- template<typename _CharT2 >
__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, [istreambuf_
iterator](#)< _CharT2 > >::__type [find](#) ([istreambuf_iterator](#)< _CharT2 >,
[istreambuf_iterator](#)< _CharT2 >, const _CharT2 &)

5.526.1 Detailed Description

**template<typename _CharT, typename _Traits> class std::istreambuf_iterator<
_CharT, _Traits >**

Provides input iterator semantics for streambufs.

Definition at line 50 of file streambuf_iterator.h.

5.526.2 Member Typedef Documentation

5.526.2.1 `template<typename _CharT, typename _Traits> typedef _CharT
std::istreambuf_iterator< _CharT, _Traits>::char_type`

Public typedefs.

Definition at line 58 of file streambuf_iterator.h.

5.526.2.2 `typedef _Traits::off_type std::iterator< input_iterator_tag, _CharT
, _Traits::off_type, _CharT*, _CharT&>::difference_type
[inherited]`

Distance between iterators is represented as this type.

Definition at line 124 of file stl_iterator_base_types.h.

5.526.2.3 `template<typename _CharT, typename _Traits> typedef
_Traits::int_type std::istreambuf_iterator< _CharT, _Traits
>::int_type`

Public typedefs.

Definition at line 60 of file streambuf_iterator.h.

5.526.2.4 `template<typename _CharT, typename _Traits> typedef
basic_istream< _CharT, _Traits> std::istreambuf_iterator< _CharT,
_Traits>::istream_type`

Public typedefs.

Definition at line 62 of file streambuf_iterator.h.

5.526.2.5 `typedef input_iterator_tag std::iterator< input_iterator_tag, _CharT
, _Traits::off_type, _CharT*, _CharT&>::iterator_category
[inherited]`

One of the [tag types](#).

Definition at line 120 of file stl_iterator_base_types.h.

5.526.2.6 `typedef _CharT* std::iterator< input_iterator_tag, _CharT,
_Traits::off_type, _CharT*, _CharT&>::pointer [inherited]`

This type represents a pointer-to-value_type.

5.526 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

Definition at line 126 of file stl_iterator_base_types.h.

5.526.2.7 `typedef _CharT & std::iterator< input_iterator_tag , _CharT
, _Traits::off_type , _CharT * , _CharT & >::reference
[inherited]`

This type represents a reference-to-value_type.

Definition at line 128 of file stl_iterator_base_types.h.

5.526.2.8 `template<typename _CharT , typename _Traits > typedef
basic_streambuf< _CharT, _Traits> std::istreambuf_iterator<
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 61 of file streambuf_iterator.h.

5.526.2.9 `template<typename _CharT , typename _Traits > typedef _Traits
std::istreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 59 of file streambuf_iterator.h.

5.526.2.10 `typedef _CharT std::iterator< input_iterator_tag , _CharT
, _Traits::off_type , _CharT * , _CharT & >::value_type
[inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file stl_iterator_base_types.h.

5.526.3 Constructor & Destructor Documentation

5.526.3.1 `template<typename _CharT , typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
) throw () [inline]`

Construct end of input stream iterator.

Definition at line 96 of file streambuf_iterator.h.

5.526.3.2 `template<typename _CharT, typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
istream_type & __s) throw () [inline]`

Construct start of input stream iterator.

Definition at line 100 of file streambuf_iterator.h.

5.526.3.3 `template<typename _CharT, typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
streambuf_type * __s) throw () [inline]`

Construct start of streambuf iterator.

Definition at line 104 of file streambuf_iterator.h.

5.526.4 Member Function Documentation

5.526.4.1 `template<typename _CharT, typename _Traits > bool
std::istreambuf_iterator< _CharT, _Traits >::equal (const
istreambuf_iterator< _CharT, _Traits > & __b) const [inline]`

Return true both iterators are end or both are not end.

Definition at line 160 of file streambuf_iterator.h.

5.526.4.2 `template<typename _CharT, typename _Traits > char_type
std::istreambuf_iterator< _CharT, _Traits >::operator* () const
[inline]`

Return the current character pointed to by iterator. This returns `/// streambuf.sgetc()`. It cannot be assigned. NB: The result of `/// operator*()` on an end of stream is undefined.

Definition at line 111 of file streambuf_iterator.h.

5.526.4.3 `template<typename _CharT, typename _Traits >
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits
>::operator++ (int) [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 140 of file streambuf_iterator.h.

References `std::basic_streambuf< _CharT, _Traits >::sbumpc()`.

5.527.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t,
typename _Pointer = _Tp*, typename _Reference = _Tp&> struct std::iterator<
_Category, _Tp, _Distance, _Pointer, _Reference >
```

Common iterator class. This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 117 of file `stl_iterator_base_types.h`.

5.527.2 Member Typedef Documentation

5.527.2.1 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_type`

Distance between iterators is represented as this type.

Reimplemented in [std::reverse_iterator< _Iterator >](#).

Definition at line 124 of file `stl_iterator_base_types.h`.

5.527.2.2 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category`

One of the [tag types](#).

Definition at line 120 of file `stl_iterator_base_types.h`.

5.527.2.3 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer`

This type represents a pointer-to-value_type.

Reimplemented in [std::reverse_iterator< _Iterator >](#).

Definition at line 126 of file `stl_iterator_base_types.h`.

5.527.2.4 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Reference std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>::reference`

This type represents a reference-to-value_type.

Reimplemented in [std::reverse_iterator<_Iterator>](#).

Definition at line 128 of file `stl_iterator_base_types.h`.

5.527.2.5 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Tp std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>::value_type`

The type "pointed to" by the iterator.

Definition at line 122 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.528 `std::iterator_traits<_Tp * >` Struct Template Reference

Partial specialization for pointer types.

Public Types

- typedef `ptrdiff_t` **difference_type**
- typedef [random_access_iterator_tag](#) **iterator_category**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `_Tp` **value_type**

5.528.1 Detailed Description

`template<typename _Tp> struct std::iterator_traits<_Tp * >`

Partial specialization for pointer types.

Definition at line 174 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.529 `std::iterator_traits< const _Tp * >` Struct Template Reference

Partial specialization for const pointer types.

Public Types

- typedef ptrdiff_t **difference_type**
- typedef [random_access_iterator_tag](#) **iterator_category**
- typedef const _Tp * **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

5.529.1 Detailed Description

`template<typename _Tp> struct std::iterator_traits< const _Tp * >`

Partial specialization for const pointer types.

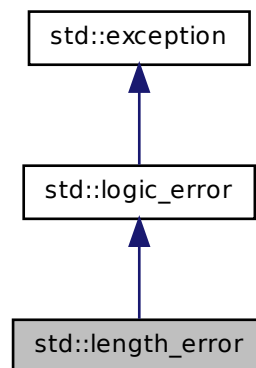
Definition at line 185 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.530 `std::length_error` Class Reference

Inheritance diagram for `std::length_error`:



Public Member Functions

- **`length_error`** (const `string` &__arg)
- virtual const char * `what` () const throw ()

5.530.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a `basic_string` instance).

Definition at line 88 of file `stdexcept`.

5.530.2 Member Function Documentation

5.530.2.1 virtual const char* `std::logic_error::what` () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

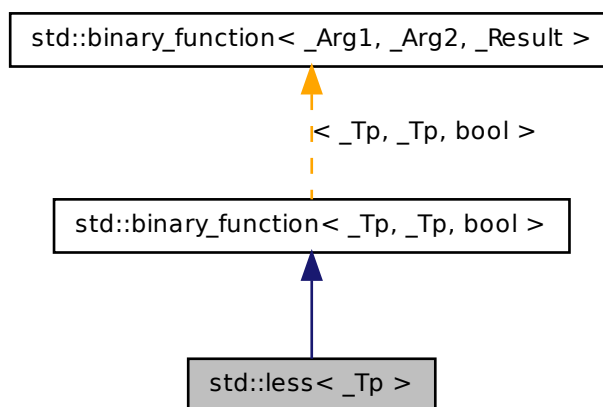
The documentation for this class was generated from the following file:

- [stdexcept](#)

5.531 std::less< _Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::less< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.531.1 Detailed Description

`template<typename _Tp> struct std::less< _Tp >`

One of the [comparison functors](#).

Definition at line 226 of file stl_function.h.

5.531.2 Member Typedef Documentation

5.531.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.531.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.531.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

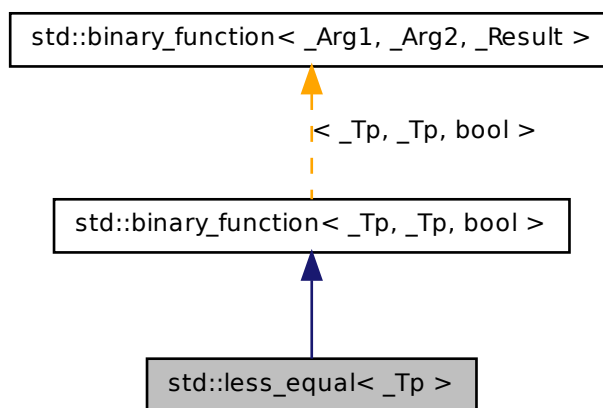
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.532 std::less_equal< _Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for `std::less_equal<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.532.1 Detailed Description

`template<typename _Tp> struct std::less_equal<_Tp>`

One of the [comparison functors](#).

Definition at line 244 of file `stl_function.h`.

5.532.2 Member Typedef Documentation

5.532.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.532.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.532.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.533 std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference

A model of a linear congruential random number generator.

Public Types

- `typedef _UIntType result_type`

Public Member Functions

- `linear_congruential_engine (result_type __s=default_seed)`
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, linear_-
congruential_engine>::value> ::type>
linear_congruential_engine (_Sseq &__q)`

- void [discard](#) (unsigned long long __z)
- [result_type](#) max () const
- [result_type](#) min () const
- [result_type](#) operator() ()
- void [seed](#) ([result_type](#) __s=default_seed)
- template<typename _Sseq >
[std::enable_if](#)< std::is_class< _Sseq >::value >::type [seed](#) (_Sseq &__q)

Static Public Attributes

- static const [result_type](#) **default_seed**
- static const [result_type](#) **increment**
- static const [result_type](#) **modulus**
- static const [result_type](#) **multiplier**

Friends

- template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits >
[std::basic_ostream](#)< _CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::linear_congruential_engine](#)< _UIntType1, __a1, __c1, __m1 > &)
- bool [operator==](#) (const [linear_congruential_engine](#) &__lhs, const [linear_congruential_engine](#) &__rhs)
- template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &, [std::linear_congruential_engine](#)< _UIntType1, __a1, __c1, __m1 > &)

5.533.1 Detailed Description

template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> class [std::linear_congruential_engine](#)< _UIntType, __a, __c, __m >

A model of a linear congruential random number generator. A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is [std::numeric_limits<_UIntType>::max\(\)](#) plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 160 of file random.h.

5.533.2 Member Typedef Documentation

5.533.2.1 `template<typename _UIntType, _UIntType __a,
_UIntType __c, _UIntType __m> typedef _UIntType
std::linear_congruential_engine< _UIntType, __a, __c, __m
>::result_type`

The type of the generated random value.

Definition at line 169 of file random.h.

5.533.3 Constructor & Destructor Documentation

5.533.3.1 `template<typename _UIntType, _UIntType __a, _UIntType __c,
_UIntType __m> std::linear_congruential_engine< _UIntType,
__a, __c, __m >::linear_congruential_engine (result_type __s =
default_seed) [inline, explicit]`

Constructs a linear_congruential_engine random number generator engine with seed __s. The default seed value is 1.

Parameters

__s The initial seed value.

Definition at line 187 of file random.h.

References std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

5.533.3.2 `template<typename _UIntType, _UIntType __a, _UIntType
__c, _UIntType __m> template<typename _Sseq, typename =
typename std::enable_if<!std::is_same<_Sseq, linear_congruential_
engine>::value> ::type> std::linear_congruential_engine<
_UIntType, __a, __c, __m >::linear_congruential_engine (_Sseq &
__q) [inline, explicit]`

Constructs a linear_congruential_engine random number generator engine seeded from the seed sequence __q.

Parameters

__q the seed sequence.

Definition at line 200 of file random.h.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m>::seed()`.

5.533.4 Member Function Documentation

5.533.4.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine<_UIntType, __a, __c, __m>::discard (unsigned long long __z) [inline]`

Discard a sequence of random numbers.

Todo

Look for a faster way to do discard.

Definition at line 250 of file random.h.

5.533.4.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::max () const [inline]`

Gets the largest possible value in the output range.

Todo

This should be constexpr.

Definition at line 241 of file random.h.

5.533.4.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::min () const [inline]`

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be > 0 , otherwise 0 is allowed.

Todo

This should be constexpr.

Definition at line 232 of file random.h.

5.533.4.4 `template<typename _UIntType, _UIntType __a, _UIntType __c,
 _UIntType __m> result_type std::linear_congruential_engine<
 _UIntType, __a, __c, __m >::operator() () [inline]`

Gets the next random number in the sequence.

Definition at line 260 of file random.h.

5.533.4.5 `template<typename _UIntType , _UIntType __a, _UIntType
 __c, _UIntType __m> template<typename _Sseq >
 std::enable_if< std::is_class< _Sseq >::value >::type
 std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
 _Sseq & __q)`

Reseeds the linear_congruential_engine random number generator engine sequence using values from the seed sequence __q.

Parameters

__q the seed sequence.

Seeds the LCR engine with a value generated by __q.

Definition at line 145 of file random.tcc.

References std::__lg(), and std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

5.533.4.6 `template<typename _UIntType , _UIntType __a, _UIntType __c,
 _UIntType __m> void std::linear_congruential_engine< _UIntType,
 __a, __c, __m >::seed (result_type __s = default_seed)`

Reseeds the linear_congruential_engine random number generator engine sequence to the seed __s.

Parameters

__s The new seed.

Seeds the LCR with integral value __s, adjusted so that the ring identity is never a member of the convergence set.

Definition at line 129 of file random.tcc.

Referenced by std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine(), and std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

5.533.5 Friends And Related Function Documentation

5.533.5.1 `template<typename _UIntType, _UIntType __a, _UIntType __c,
_UIntType __m> template<typename _UIntType1, _UIntType1
__a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT
, typename _Traits > std::basic_ostream<_CharT, _Traits>&
operator<< (std::basic_ostream<_CharT, _Traits> &, const
std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1 > &
) [friend]`

Writes the textual representation of the state $x(i)$ of x to `__os`.

Parameters

`__os` The output stream.

`__lcr` A % [linear_congruential_engine](#) random number generator.

Returns

`__os`.

5.533.5.2 `template<typename _UIntType, _UIntType __a,
_UIntType __c, _UIntType __m> bool operator==(const
linear_congruential_engine<_UIntType, __a, __c, __m> & __lhs,
const linear_congruential_engine<_UIntType, __a, __c, __m> &
__rhs) [friend]`

Compares two linear congruential random number generator objects of the same type for equality.

Parameters

`__lhs` A linear congruential random number generator object.

`__rhs` Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 278 of file random.h.

5.533.5.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits > &, std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &) [friend]`

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

Parameters

`__is` The input stream.

`__lcr` A % [linear_congruential_engine](#) random number generator.

Returns

`__is`.

5.533.6 Member Data Documentation

5.533.6.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::increment [static]`

An increment.

Definition at line 174 of file random.h.

5.533.6.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::modulus [static]`

The modulus.

Definition at line 176 of file random.h.

5.533.6.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> const _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::multiplier [static]`

The multiplier.

Definition at line 172 of file random.h.

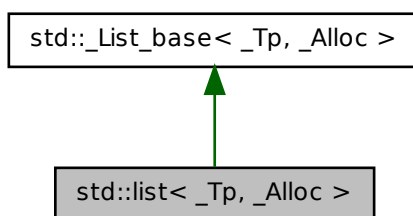
The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.534 `std::list< _Tp, _Alloc >` Class Template Reference

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Inheritance diagram for `std::list< _Tp, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_List_const_iterator< _Tp >` **const_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `_Tp_alloc_type::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_List_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- [list](#) ()
- [list](#) (const allocator_type &__a)
- [list](#) (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
 [list](#) (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- [list](#) (const [list](#) &__x)
- [list](#) (size_type __n)
- [list](#) ([list](#) &&__x)
- [list](#) ([initializer_list](#)< value_type > __l, const allocator_type &__a=allocator_type())
- void [assign](#) (size_type __n, const value_type &__val)
- template<typename _InputIterator >
 void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) ([initializer_list](#)< value_type > __l)
- reference [back](#) ()
- const_reference [back](#) () const
- iterator [begin](#) ()
- const_iterator [begin](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [cend](#) () const
- void [clear](#) ()
- const_reverse_iterator [crbegin](#) () const
- const_reverse_iterator [crend](#) () const
- template<typename... _Args>
 iterator [emplace](#) (iterator __position, _Args &&...__args)
- template<typename... _Args>
 void [emplace_back](#) (_Args &&...__args)
- template<typename... _Args>
 void [emplace_front](#) (_Args &&...__args)
- bool [empty](#) () const
- iterator [end](#) ()
- const_iterator [end](#) () const
- iterator [erase](#) (iterator __position)
- iterator [erase](#) (iterator __first, iterator __last)
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const
- iterator [insert](#) (iterator __position, const value_type &__x)
- iterator [insert](#) (iterator __position, value_type &&__x)

- void [insert](#) (iterator __p, [initializer_list](#)< value_type > __l)
- void [insert](#) (iterator __position, size_type __n, const value_type &__x)
- template<typename _InputIterator >
void [insert](#) (iterator __position, _InputIterator __first, _InputIterator __last)
- size_type [max_size](#) () const
- void [merge](#) (list &&__x)
- void [merge](#) (list &__x)
- template<typename _StrictWeakOrdering >
void [merge](#) (list &&, _StrictWeakOrdering)
- template<typename _StrictWeakOrdering >
void [merge](#) (list &__x, _StrictWeakOrdering __comp)
- list & operator= (const list &__x)
- list & operator= (list &&__x)
- list & operator= ([initializer_list](#)< value_type > __l)
- void [pop_back](#) ()
- void [pop_front](#) ()
- void [push_back](#) (const value_type &__x)
- void [push_back](#) (value_type &&__x)
- void [push_front](#) (value_type &&__x)
- void [push_front](#) (const value_type &__x)
- [reverse_iterator](#) rbegin ()
- [const_reverse_iterator](#) rbegin () const
- void [remove](#) (const _Tp &__value)
- template<typename _Predicate >
void [remove_if](#) (_Predicate)
- [reverse_iterator](#) rend ()
- [const_reverse_iterator](#) rend () const
- void [resize](#) (size_type __new_size)
- void [resize](#) (size_type __new_size, const value_type &__x)
- void [reverse](#) ()
- size_type [size](#) () const
- template<typename _StrictWeakOrdering >
void [sort](#) (_StrictWeakOrdering)
- void [sort](#) ()
- void [splice](#) (iterator __position, list &__x, iterator __i)
- void [splice](#) (iterator __position, list &&__x, iterator __i)
- void [splice](#) (iterator __position, list &__x)
- void [splice](#) (iterator __position, list &__x, iterator __first, iterator __last)
- void [splice](#) (iterator __position, list &&__x, iterator __first, iterator __last)
- void [splice](#) (iterator __position, list &&__x)
- void [swap](#) (list &__x)
- template<typename _BinaryPredicate >
void [unique](#) (_BinaryPredicate)
- void [unique](#) ()

Protected Types

- typedef [_List_node](#)< _Tp > **_Node**
- typedef _Alloc::template rebind< [_List_node](#)< _Tp > >::other **_Node_alloc_type**

Protected Member Functions

- template<typename _Integer >
void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- template<typename _InputIterator >
void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_check_equal_allocators** ([list](#) &__x)
- void **_M_clear** ()
- template<typename... _Args>
[_Node](#) * **_M_create_node** (_Args &&...__args)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- void **_M_erase** ([iterator](#) __position)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__x)
- [_List_node](#)< _Tp > * **_M_get_node** ()
- [_Node_alloc_type](#) & **_M_get_Node_allocator** ()
- const [_Node_alloc_type](#) & **_M_get_Node_allocator** () const
- [_Tp_alloc_type](#) **_M_get_Tp_allocator** () const
- void **_M_init** ()
- template<typename _InputIterator >
void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer >
void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename... _Args>
void **_M_insert** ([iterator](#) __position, _Args &&...__args)
- void **_M_put_node** ([_List_node](#)< _Tp > *__p)
- void **_M_transfer** ([iterator](#) __position, [iterator](#) __first, [iterator](#) __last)

Protected Attributes

- [_List_impl](#) **_M_impl**

5.534.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class
std::list<_Tp, _Alloc >
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike [std::vector](#) and [std::deque](#), random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, [std::list](#) provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

```
A <---> B <---> C <---> D
```

is actually circular; a link exists between `A` and `D`. The list class holds (as its only data member) a private `list::iterator` pointing to `D`, not to `A`! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

Definition at line 417 of file `stl_list.h`.

5.534.2 Constructor & Destructor Documentation

```
5.534.2.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list ( ) [inline]
```

Default constructor creates no elements.

Definition at line 500 of file `stl_list.h`.

5.534.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (const allocator_type & __a)
[inline, explicit]`

Creates a list with no elements.

Parameters

a An allocator object.

Definition at line 508 of file stl_list.h.

5.534.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (size_type __n) [inline,
explicit]`

Creates a list with default constructed elements.

Parameters

n The number of elements to initially create.

This constructor fills the list with *n* default constructed elements.

Definition at line 520 of file stl_list.h.

5.534.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (size_type __n, const value_type &
__value, const allocator_type & __a = allocator_type())
[inline]`

Creates a list with copies of an exemplar element.

Parameters

n The number of elements to initially create.

value An element to copy.

a An allocator object.

This constructor fills the list with *n* copies of *value*.

Definition at line 532 of file stl_list.h.

5.534.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (const list<_Tp, _Alloc> & __x)
[inline]`

List copy constructor.

Parameters

x A list of identical element and allocator types.

The newly-created list uses a copy of the allocation object used by *x*.

Definition at line 559 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

5.534.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (list<_Tp, _Alloc> && __x)
[inline]`

List move constructor.

Parameters

x A list of identical element and allocator types.

The newly-created list contains the exact contents of *x*. The contents of *x* are a valid, but unspecified list.

Definition at line 571 of file `stl_list.h`.

5.534.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc>::list (initializer_list<value_type> & __l,
const allocator_type & __a = allocator_type()) [inline]`

Builds a list from an [initializer_list](#).

Parameters

l An [initializer_list](#) of `value_type`.

a An allocator object.

Create a list consisting of copies of the elements in the [initializer_list](#) *l*. This is linear in `l.size()`.

Definition at line 582 of file `stl_list.h`.

5.534.2.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 template<typename _InputIterator > std::list< _Tp, _Alloc >::list (
 _InputIterator __first, _InputIterator __last, const allocator_type &
 __a = allocator_type()) [inline]`

Builds a list from a range.

Parameters

first An input iterator.

last An input iterator.

a An allocator object.

Create a list consisting of copies of the elements from *[first,last)*. This is linear in N (where N is distance(*first,last*)).

Definition at line 599 of file stl_list.h.

5.534.3 Member Function Documentation

5.534.3.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 template<typename... _Args> _Node* std::list< _Tp, _Alloc
 >::_M_create_node (_Args &&... __args) [inline,
 protected]`

Parameters

x An instance of user data.

Allocates space for a new node and constructs a copy of *x* in it.

Definition at line 477 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::emplace(), and std::list< _Tp, _Alloc >::insert().

5.534.3.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
 void std::list< _Tp, _Alloc >::assign (size_type __n, const
 value_type & __val) [inline]`

Assigns a given value to a list.

Parameters

n Number of elements to be assigned.

val Value to be assigned.

This function fills a list with n copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 670 of file `stl_list.h`.

```
5.534.3.3  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            template<typename _InputIterator > void std::list<_Tp, _Alloc
            >::assign ( _InputIterator __first, _InputIterator __last )
            [inline]
```

Assigns a range to a list.

Parameters

first An input iterator.

last An input iterator.

This function fills a list with copies of the elements in the range $[first, last)$.

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 687 of file `stl_list.h`.

```
5.534.3.4  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            void std::list<_Tp, _Alloc >::assign ( initializer_list< value_type >
            __l ) [inline]
```

Assigns an [initializer_list](#) to a list.

Parameters

l An [initializer_list](#) of `value_type`.

Replace the contents of the list with copies of the elements in the [initializer_list](#) *l*. This is linear in `l.size()`.

Definition at line 703 of file `stl_list.h`.

References `std::list<_Tp, _Alloc >::assign()`.

Referenced by `std::list<_Tp, _Alloc >::assign()`.

```
5.534.3.5  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
            reference std::list<_Tp, _Alloc >::back ( ) [inline]
```

Returns a read/write reference to the data at the last element of the list.

Definition at line 903 of file stl_list.h.

References std::end().

5.534.3.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list<_Tp, _Alloc>::back () const [inline]`

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 915 of file stl_list.h.

References std::end().

5.534.3.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list<_Tp, _Alloc>::begin () [inline]`

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 718 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::list(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::operator=(), std::operator==(), std::list< _Tp, _Alloc >::remove_if(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::splice(), and std::list< _Tp, _Alloc >::unique().

5.534.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 727 of file stl_list.h.

5.534.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list<_Tp, _Alloc>::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 791 of file stl_list.h.

5.534.3.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 800 of file `stl_list.h`.

5.534.3.11 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::clear () [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1182 of file `stl_list.h`.

5.534.3.12 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 809 of file `stl_list.h`.

References `std::end()`.

5.534.3.13 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 818 of file `stl_list.h`.

References `std::begin()`.

5.534.3.14 `template<typename _Tp , typename _Alloc > template<typename...
_Args> list< _Tp, _Alloc >::iterator list::emplace (iterator
__position, _Args &&... __args)`

Constructs object in list before specified iterator.

Parameters

position A const_iterator into the list.

args Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 87 of file list.tcc.

References std::list< _Tp, _Alloc >::_M_create_node().

5.534.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::list< _Tp, _Alloc >::empty () const [inline]`

Returns true if the list is empty. (Thus [begin\(\)](#) would equal [end\(\)](#).)

Definition at line 828 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::sort(), and std::list< _Tp, _Alloc >::splice().

5.534.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 736 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::list(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::operator==(), std::list< _Tp, _Alloc >::remove_if(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::splice(), and std::list< _Tp, _Alloc >::unique().

5.534.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 745 of file stl_list.h.

5.534.3.18 `template<typename _Tp, typename _Alloc> list< _Tp, _Alloc>::iterator list::erase (iterator __position)`

Remove element at given position.

Parameters

position Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 108 of file `list.tcc`.

Referenced by `std::list< _Tp, _Alloc>::operator=()`, and `std::list< _Tp, _Alloc>::resize()`.

5.534.3.19 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc>::erase (iterator __first, iterator __last) [inline]`

Remove a range of elements.

Parameters

first Iterator pointing to the first element to be erased.

last Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1148 of file `stl_list.h`.

5.534.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::front () [inline]`

Returns a read/write reference to the data at the first element of the list.

Definition at line 887 of file stl_list.h.

References std::begin().

5.534.3.21 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 895 of file stl_list.h.

References std::begin().

5.534.3.22 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::list< _Tp, _Alloc >::get_allocator () const
[inline]`

Get a copy of the memory allocation object.

Reimplemented from [std::List_base< _Tp, _Alloc >](#).

Definition at line 709 of file stl_list.h.

5.534.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (iterator __position,
value_type && __x) [inline]`

Inserts given rvalue into list before specified iterator.

Parameters

position An iterator into the list.

x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1049 of file stl_list.h.

5.534.3.24 `template<typename _Tp, typename _Alloc> list<_Tp, _Alloc>::iterator list::insert (iterator __position, const value_type & __x)`

Inserts given value into list before specified iterator.

Parameters

position An iterator into the list.

x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 98 of file list.tcc.

References `std::list<_Tp, _Alloc>::_M_create_node()`.

Referenced by `std::list<_Tp, _Alloc>::operator=()`, and `std::list<_Tp, _Alloc>::resize()`.

5.534.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::insert (iterator __p, initializer_list<value_type> __l) [inline]`

Inserts the contents of an [initializer_list](#) into list before specified iterator.

Parameters

p An iterator into the list.

l An [initializer_list](#) of value_type.

This function will insert copies of the data in the [initializer_list](#) *l* into the list before the location specified by *p*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1066 of file stl_list.h.

References `std::list<_Tp, _Alloc>::insert()`.

Referenced by `std::list<_Tp, _Alloc>::insert()`.

5.534.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::insert (iterator __position, size_type
__n, const value_type & __x) [inline]`

Inserts a number of copies of given data into the list.

Parameters

position An iterator into the list.

n Number of elements to be inserted.

x Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1083 of file stl_list.h.

5.534.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::list< _Tp,
_Alloc >::insert (iterator __position, _InputIterator __first,
_InputIterator __last) [inline]`

Inserts a range into the list.

Parameters

position An iterator into the list.

first An input iterator.

last An input iterator.

This function will insert copies of the data in the range [*first,last*) into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1104 of file stl_list.h.

5.534.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::max_size () const [inline]`

Returns the [size\(\)](#) of the largest possible list.

Definition at line 838 of file stl_list.h.

5.534.3.29 `template<typename _Tp , typename _Alloc > void list::merge (list< _Tp, _Alloc > && __x)`

Merge sorted lists.

Parameters

x Sorted list to merge.

Assumes that both *x* and this list are sorted according to operator<(). Merges elements of *x* into this list in sorted order, leaving *x* empty when complete. Elements in this list precede elements in *x* that are equal.

Definition at line 287 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), std::begin(), std::list< _Tp, _Alloc >::end(), and std::end().

Referenced by std::list< _Tp, _Alloc >::sort().

5.534.3.30 `template<typename _Tp , typename _Alloc > template<typename _StrictWeakOrdering > void list::merge (list< _Tp, _Alloc > && __x, _StrictWeakOrdering __comp)`

Merge sorted lists according to comparison function.

Parameters

x Sorted list to merge.

StrictWeakOrdering Comparison function defining sort order.

Assumes that both *x* and this list are sorted according to StrictWeakOrdering. Merges elements of *x* into this list in sorted order, leaving *x* empty when complete. Elements in this list precede elements in *x* that are equivalent according to StrictWeakOrdering().

Definition at line 321 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), std::begin(), std::list< _Tp, _Alloc >::end(), and std::end().

5.534.3.31 `template<typename _Tp , typename _Alloc > list< _Tp, _Alloc > & list::operator= (const list< _Tp, _Alloc > & __x)`

List assignment operator.

No explicit dtor needed as the _Base dtor takes care of things. The _Base dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

x A list of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 184 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::end(), std::list< _Tp, _Alloc >::erase(), and std::list< _Tp, _Alloc >::insert().

5.534.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list& std::list< _Tp, _Alloc >::operator= (initializer_list<
value_type > __l) [inline]`

List initializer list assignment operator.

Parameters

l An [initializer_list](#) of value_type.

Replace the contents of the list with copies of the elements in the [initializer_list](#) *l*. This is linear in l.size().

Definition at line 652 of file stl_list.h.

5.534.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list& std::list< _Tp, _Alloc >::operator= (list< _Tp, _Alloc > &&
__x) [inline]`

List move assignment operator.

Parameters

x A list of identical element and allocator types.

The contents of *x* are moved into this list (without copying). *x* is a valid, but unspecified list

Definition at line 635 of file stl_list.h.

5.534.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_back () [inline]`

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1001 of file `stl_list.h`.

5.534.3.35 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::pop_front () [inline]`

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 961 of file `stl_list.h`.

References `std::begin()`.

5.534.3.36 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::push_back (const value_type & __x
) [inline]`

Add data to the end of the list.

Parameters

`x` Data to be added.

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 975 of file `stl_list.h`.

References `std::end()`.

5.534.3.37 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list<_Tp, _Alloc>::push_front (const value_type & __x
) [inline]`

Add data to the front of the list.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 934 of file stl_list.h.

References std::begin().

5.534.3.38 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
reverse_iterator std::list< _Tp, _Alloc >::rbegin () [inline]`

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 754 of file stl_list.h.

References std::end().

5.534.3.39 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
const_reverse_iterator std::list< _Tp, _Alloc >::rbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 763 of file stl_list.h.

References std::end().

5.534.3.40 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
void std::list< _Tp, _Alloc >::remove (const _Tp & __value)`

Remove all elements equal to value.

Parameters

value The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.534.3.41 `template<typename _Tp, typename _Alloc> template<typename
_Predicate> void list::remove_if (_Predicate __pred)`

Remove all elements satisfying a predicate.

Parameters

Predicate Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 391 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, and `std::list< _Tp, _Alloc >::end()`.

5.534.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 781 of file stl_list.h.

References `std::begin()`.

5.534.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 772 of file stl_list.h.

References `std::begin()`.

5.534.3.44 `template<typename _Tp, typename _Alloc> void list::resize (
size_type __new_size, const value_type & __x)`

Resizes the list to the specified number of elements.

Parameters

new_size Number of elements the list should contain.

x Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

Definition at line 153 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::end(), std::list< _Tp, _Alloc >::erase(), and std::list< _Tp, _Alloc >::insert().

5.534.3.45 **template<typename _Tp, typename _Alloc > void list::resize (size_type __new_size)**

Resizes the list to the specified number of elements.

Parameters

new_size Number of elements the list should contain.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

Definition at line 138 of file list.tcc.

References std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::end(), and std::list< _Tp, _Alloc >::erase().

5.534.3.46 **template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::reverse () [inline]**

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1402 of file stl_list.h.

5.534.3.47 **template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list< _Tp, _Alloc >::size () const [inline]**

Returns the number of elements in the list.

Definition at line 833 of file stl_list.h.

References std::begin(), __gnu_cxx::distance(), and std::end().

5.534.3.48 `template<typename _Tp, typename _Alloc > void list::sort ()`

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 353 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

5.534.3.49 `template<typename _Tp, typename _Alloc > template<typename _StrictWeakOrdering > void list::sort (_StrictWeakOrdering __comp)`

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 430 of file list.tcc.

References `std::list< _Tp, _Alloc >::begin()`, `std::list< _Tp, _Alloc >::empty()`, `std::list< _Tp, _Alloc >::merge()`, `std::list< _Tp, _Alloc >::splice()`, and `std::list< _Tp, _Alloc >::swap()`.

5.534.3.50 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice (iterator __position, list< _Tp, _Alloc > && __x, iterator __first, iterator __last) [inline]`

Insert range from another list.

Parameters

position Iterator referencing the element to insert before.

x Source list.

first Iterator referencing the start of range in *x*.

last Iterator referencing the end of range in *x*.

Removes elements in the range `[first,last)` and inserts them before *position* in constant time.

Undefined if *position* is in `[first,last)`.

Definition at line 1268 of file `stl_list.h`.

5.534.3.51 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (iterator __position, list<
_Tp, _Alloc > && __x, iterator __i) [inline]`

Insert element from another list.

Parameters

position Iterator referencing the element to insert before.

x Source list.

i Iterator referencing the element to move.

Removes the element in list *x* referenced by *i* and inserts it into the current list before *position*.

Definition at line 1232 of file stl_list.h.

5.534.3.52 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (iterator __position, list<
_Tp, _Alloc > && __x) [inline]`

Insert contents of another list.

Parameters

position Iterator referencing the element to insert before.

x Source list.

The elements of *x* are inserted in constant time in front of the element referenced by *position*. *x* becomes an empty list.

Requires this != *x*.

Definition at line 1202 of file stl_list.h.

References std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::empty(), and std::list< _Tp, _Alloc >::end().

Referenced by std::list< _Tp, _Alloc >::sort().

5.534.3.53 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (list< _Tp, _Alloc > & __x)
[inline]`

Swaps data with another list.

Parameters

x A list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global [std::swap\(\)](#) function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 1165 of file `stl_list.h`.

Referenced by `std::list<_Tp, _Alloc >::sort()`, and `std::swap()`.

5.534.3.54 `template<typename _Tp , typename _Alloc > void list::unique ()`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 266 of file `list.tcc`.

References `std::list<_Tp, _Alloc >::begin()`, and `std::list<_Tp, _Alloc >::end()`.

5.534.3.55 `template<typename _Tp , typename _Alloc > template<typename _BinaryPredicate > void list::unique (_BinaryPredicate __binary_pred)`

Remove consecutive elements satisfying a predicate.

Parameters

BinaryPredicate Binary predicate function or object.

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 409 of file `list.tcc`.

References `std::list<_Tp, _Alloc >::begin()`, and `std::list<_Tp, _Alloc >::end()`.

The documentation for this class was generated from the following files:

- [stl_list.h](#)
- [list.tcc](#)

5.535 std::locale Class Reference

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

Classes

- class [facet](#)

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

- class [id](#)

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.

Public Types

- typedef int [category](#)

Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const [locale](#) &__other) throw ()
- [locale](#) (const [locale](#) &__base, const char *__s, [category](#) __cat)
- [locale](#) (const [locale](#) &__base, const [locale](#) &__add, [category](#) __cat)
- [locale](#) (const char *__s)
- template<typename _Facet >
 [locale](#) (const [locale](#) &__other, _Facet *__f)
- [~locale](#) () throw ()
- template<typename _Facet >
 [locale combine](#) (const [locale](#) &__other) const
- [string name](#) () const
- bool [operator!=](#) (const [locale](#) &__other) const throw ()
- template<typename _Char, typename _Traits, typename _Alloc >
 bool [operator\(\)](#) (const [basic_string](#)< _Char, _Traits, _Alloc > &__s1, const [basic_string](#)< _Char, _Traits, _Alloc > &__s2) const

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator() (const basic_string< _CharT, _Traits, _Alloc > &__s1, const`
`basic_string< _CharT, _Traits, _Alloc > &__s2) const`
- `const locale & operator= (const locale &__other) throw ()`
- `bool operator== (const locale &__other) const throw ()`

Static Public Member Functions

- `static const locale & classic ()`
- `static locale global (const locale &)`

Static Public Attributes

- `static const category none`
- `static const category ctype`
- `static const category numeric`
- `static const category collate`
- `static const category time`
- `static const category monetary`
- `static const category messages`
- `static const category all`

Friends

- `struct __use_cache`
- `class _Impl`
- `class facet`
- `template<typename _Facet >`
`bool has_facet (const locale &) throw ()`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &)`

5.535.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing. Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file `locale_classes.h`.

5.535.2 Member Typedef Documentation

5.535.2.1 typedef int std::locale::category

Definition of [locale::category](#).

Definition at line 67 of file locale_classes.h.

5.535.3 Constructor & Destructor Documentation

5.535.3.1 std::locale::locale () throw ()

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by combine().

5.535.3.2 std::locale::locale (const locale & __other) throw ()

Copy constructor.

Constructs a copy of *other*.

Parameters

other The locale to copy.

5.535.3.3 std::locale::locale (const char * __s) [explicit]

Named locale constructor.

Constructs a copy of the named C library locale.

Parameters

s Name of the locale to construct.

Exceptions

[std::runtime_error](#) if *s* is null or an undefined locale.

5.535.3.4 `std::locale::locale (const locale & __base, const char * __s, category __cat)`

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

Parameters

base The locale to copy.

s Name of the locale to use facets from.

cat Set of categories defining the facets to use from *s*.

Exceptions

[*std::runtime_error*](#) if *s* is null or an undefined locale.

5.535.3.5 `std::locale::locale (const locale & __base, const locale & __add, category __cat)`

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

Parameters

base The locale to copy.

add The locale to use facets from.

cat Set of categories defining the facets to use from *add*.

5.535.3.6 `template<typename _Facet > std::locale::locale (const locale & __other, _Facet * __f)`

Construct locale with another facet.

Constructs a copy of the locale *other*. The facet is added to , replacing an existing facet of type *Facet* if there is one. If *__f* is null, this locale is a copy of *other*.

Parameters

other The locale to copy.

f The facet to add in.

Definition at line 43 of file locale_classes.tcc.

5.535.3.7 std::locale::~~locale () throw ()

Locale destructor.

5.535.4 Member Function Documentation

5.535.4.1 static const locale& std::locale::classic () [static]

Return reference to the C locale.

5.535.4.2 template<typename _Facet > locale std::locale::combine (const locale & __other) const

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type Facet from the locale *other* into the new locale.

Parameters

Facet The facet type to copy from other

other The locale to copy from.

Returns

Newly constructed locale.

Exceptions

[*std::runtime_error*](#) if other has no facet of type Facet.

Definition at line 61 of file locale_classes.tcc.

References locale().

5.535.4.3 static locale std::locale::global (const locale &) [static]

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call std::setlocale(LC_ALL, loc.name()).

Parameters

locale The new locale to make global.

Returns

Copy of the old global locale.

5.535.4.4 string std::locale::name () const

Return locale name.

Returns

Locale name or "*" if unnamed.

5.535.4.5 bool std::locale::operator!=(const locale & __other) const throw () [inline]

Locale inequality.

Parameters

other The locale to compare against.

Returns

! (*this == other)

Definition at line 234 of file locale_classes.h.

5.535.4.6 template<typename _Char , typename _Traits , typename _Alloc > bool std::locale::operator() (const basic_string< _Char, _Traits, _Alloc > & __s1, const basic_string< _Char, _Traits, _Alloc > & __s2) const

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector *v* of strings could be sorted according to locale *loc* by doing:

```
std::sort(v.begin(), v.end(), loc);
```

Parameters

s1 First string to compare.

s2 Second string to compare.

Returns

True if collate<Char> facet compares *s1* < *s2*, else false.

5.535.4.7 `const locale& std::locale::operator= (const locale & __other) throw ()`

Assignment operator.

Set this locale to be a copy of *other*.

Parameters

other The locale to copy.

Returns

A reference to this locale.

5.535.4.8 `bool std::locale::operator== (const locale & __other) const throw ()`

Locale equality.

Parameters

other The locale to compare against.

Returns

True if *other* and this refer to the same locale instance, are copies, or have the same name. False otherwise.

5.535.5 Friends And Related Function Documentation

5.535.5.1 `template<typename _Facet > bool has_facet (const locale &)
throw () [friend]`

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Parameters

Facet The facet type to test the presence of.

locale The locale to test.

Returns

true if locale contains a facet of type Facet, else false.

5.535.5.2 template<typename _Facet > const _Facet& use_facet (const locale &) [friend]

Return a facet.

use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws [std::bad_cast](#) if the locale doesn't contain a facet of type Facet.

Parameters

Facet The facet type to access.

locale The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

[std::bad_cast](#) if locale doesn't contain a facet of type Facet.

5.535.6 Member Data Documentation**5.535.6.1 const category std::locale::all [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

Definition at line 105 of file locale_classes.h.

5.535.6.2 const category std::locale::collate [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 101 of file `locale_classes.h`.

5.535.6.3 const category std::locale::ctype [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 99 of file `locale_classes.h`.

5.535.6.4 const category std::locale::messages [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

5.535.6.5 const category std::locale::monetary [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 103 of file `locale_classes.h`.

5.535.6.6 const category std::locale::none [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 98 of file `locale_classes.h`.

5.535.6.7 `const category std::locale::numeric` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 100 of file `locale_classes.h`.

5.535.6.8 `const category std::locale::time` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 102 of file `locale_classes.h`.

The documentation for this class was generated from the following files:

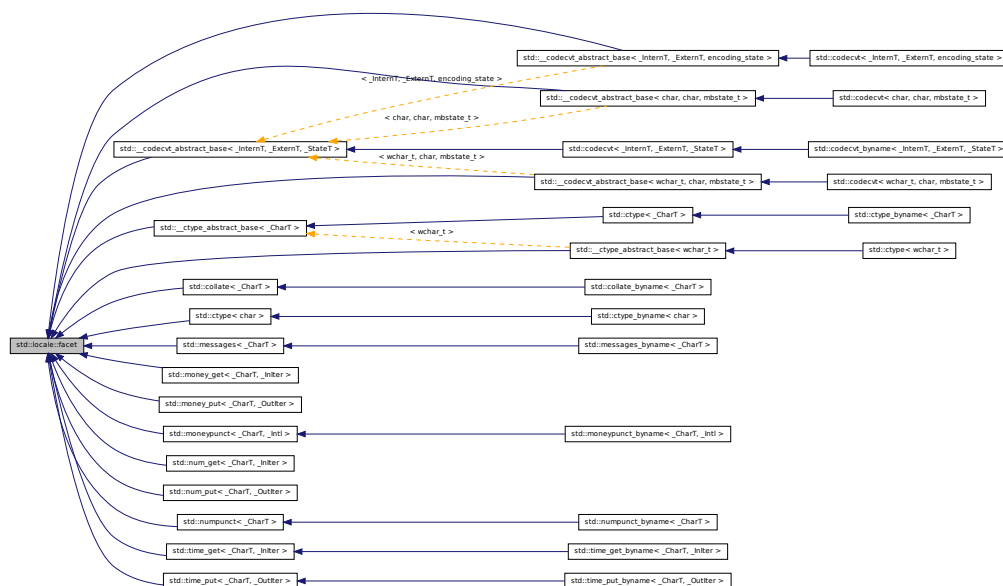
- [locale_classes.h](#)
- [locale_classes.tcc](#)

5.536 `std::locale::facet` Class Reference

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Inheritance diagram for std::locale::facet:



Protected Member Functions

- `facet` (size_t __refs=0) throw ()
- virtual `~facet` ()

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Friends

- class `locale`
- class `locale::_Impl`

5.536.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management. Facets may not be copied or assigned.

Definition at line 337 of file `locale_classes.h`.

5.536.2 Constructor & Destructor Documentation

5.536.2.1 `std::locale::facet::facet (size_t __refs = 0) throw () [inline, explicit, protected]`

Facet constructor.

This is the constructor provided by the standard. If `refs` is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

Parameters

refs The initial value for reference count.

Definition at line 369 of file `locale_classes.h`.

5.536.2.2 `virtual std::locale::facet::~~facet () [protected, virtual]`

Facet destructor.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

5.537 std::locale::id Class Reference

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

Public Member Functions

- [id](#) ()
- `size_t _M_id () const throw ()`

Friends

- `template<typename _Facet >`
`bool has_facet (const locale &) throw ()`
- `class locale`
- `class locale::_Impl`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &)`

5.537.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.

Definition at line 435 of file locale_classes.h.

5.537.2 Constructor & Destructor Documentation

5.537.2.1 `std::locale::id::id () [inline]`

Constructor.

Definition at line 466 of file locale_classes.h.

5.537.3 Friends And Related Function Documentation

5.537.3.1 `template<typename _Facet > bool has_facet (const locale &) throw () [friend]`

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Parameters

Facet The facet type to test the presence of.

locale The locale to test.

Returns

true if locale contains a facet of type Facet, else false.

5.537.3.2 `template<typename _Facet > const _Facet& use_facet (const locale &) [friend]`

Return a facet.

use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws [std::bad_cast](#) if the locale doesn't contain a facet of type Facet.

Parameters

Facet The facet type to access.

locale The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

[std::bad_cast](#) if locale doesn't contain a facet of type Facet.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

5.538 `std::lock_guard< _Mutex >` Class Template Reference

Scoped lock idiom.

Public Types

- `typedef _Mutex mutex_type`

Public Member Functions

- **lock_guard** (mutex_type &__m)
- **lock_guard** (mutex_type &__m, [adopt_lock_t](#))
- **lock_guard** (const [lock_guard](#) &)
- [lock_guard](#) & **operator=** (const [lock_guard](#) &)

5.538.1 Detailed Description

`template<typename _Mutex> class std::lock_guard< _Mutex >`

Scoped lock idiom.

Definition at line 392 of file mutex.

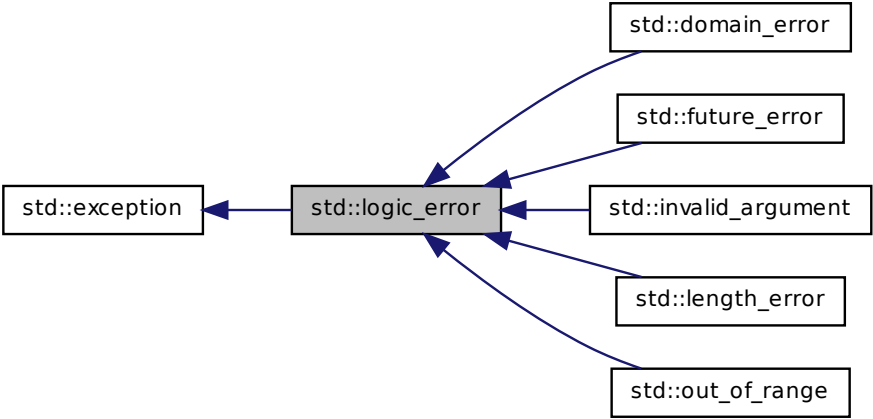
The documentation for this class was generated from the following file:

- [mutex](#)

5.539 std::logic_error Class Reference

One of two subclasses of exception.

Inheritance diagram for std::logic_error:



Public Member Functions

- [logic_error](#) (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.539.1 Detailed Description

One of two subclasses of exception. Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

Definition at line 53 of file `stdexcept`.

5.539.2 Constructor & Destructor Documentation

5.539.2.1 `std::logic_error::logic_error (const string & __arg) [explicit]`

Takes a character string describing the error.

5.539.3 Member Function Documentation

5.539.3.1 `virtual const char* std::logic_error::what () const throw () [virtual]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

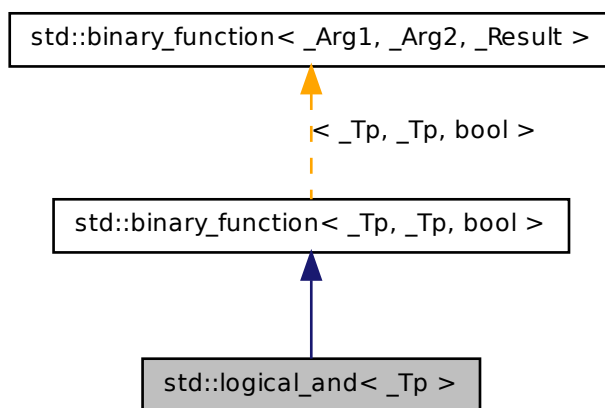
The documentation for this class was generated from the following file:

- [stdexcept](#)

5.540 `std::logical_and< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for std::logical_and< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.540.1 Detailed Description

`template<typename _Tp> struct std::logical_and< _Tp >`

One of the [Boolean operations functors](#).

Definition at line 263 of file `stl_function.h`.

5.540.2 Member Typedef Documentation

5.540.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.540.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.540.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

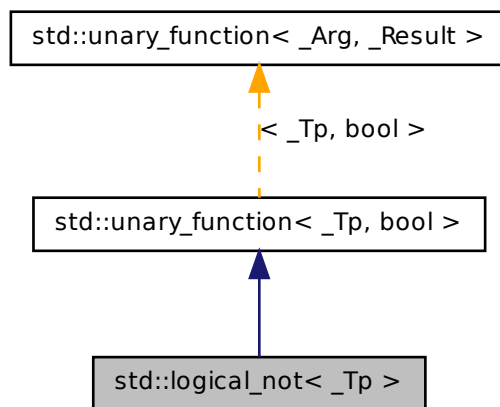
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.541 `std::logical_not< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for std::logical_not< _Tp >:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `bool` [result_type](#)

Public Member Functions

- `bool` **operator()** (`const _Tp &__x`) `const`

5.541.1 Detailed Description

`template<typename _Tp> struct std::logical_not< _Tp >`

One of the [Boolean operations functors](#).

Definition at line 281 of file `stl_function.h`.

5.541.2 Member Typedef Documentation

5.541.2.1 `typedef _Tp std::unary_function< _Tp , bool >::argument_type` `[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.541.2.2 `typedef bool std::unary_function< _Tp , bool >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

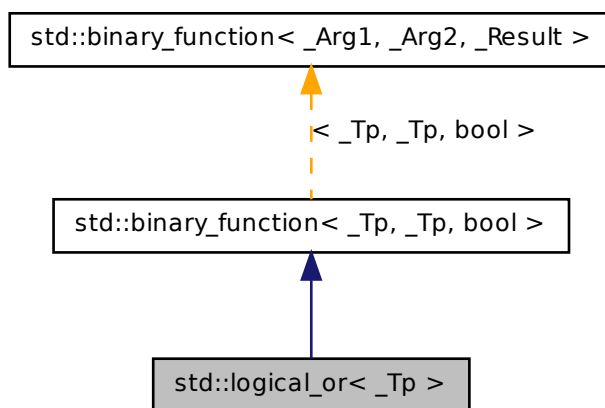
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.542 `std::logical_or< _Tp >` Struct Template Reference

One of the [Boolean operations functors](#).

Inheritance diagram for std::logical_or< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.542.1 Detailed Description

`template<typename _Tp> struct std::logical_or< _Tp >`

One of the [Boolean operations functors](#).

Definition at line 272 of file `stl_function.h`.

5.542.2 Member Typedef Documentation

5.542.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.542.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.542.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.543 `std::lognormal_distribution< _RealType >` Class Template Reference

A [lognormal_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- **lognormal_distribution** (_RealType __m=_RealType(0), _RealType __s=_RealType(1))
- **lognormal_distribution** (const [param_type](#) &__p)
- _RealType **m** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- void **param** (const [param_type](#) &__param)
- [param_type](#) **param** () const
- void **reset** ()
- _RealType **s** () const

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_ostream](#)< _CharT, _Traits > & **operator<<** ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::lognormal_distribution](#)< _RealType1 > &)
- template<typename _RealType1 >
bool **operator==** (const [std::lognormal_distribution](#)< _RealType1 > &__d1, const [std::lognormal_distribution](#)< _RealType1 > &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & **operator>>** ([std::basic_istream](#)< _CharT, _Traits > &, [std::lognormal_distribution](#)< _RealType1 > &)

5.543.1 Detailed Description

template<typename _RealType = double> class std::lognormal_distribution< _RealType >

A [lognormal_distribution](#) random number distribution. The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2177 of file random.h.

5.543.2 Member Typedef Documentation

5.543.2.1 `template<typename _RealType = double> typedef _RealType
std::lognormal_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2184 of file random.h.

5.543.3 Member Function Documentation

5.543.3.1 `template<typename _RealType = double> result_type
std::lognormal_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2268 of file random.h.

5.543.3.2 `template<typename _RealType = double> result_type
std::lognormal_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2261 of file random.h.

5.543.3.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::lognormal_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2276 of file random.h.

References `std::lognormal_distribution< _RealType >::operator()()`, and
`std::lognormal_distribution< _RealType >::param()`.

Referenced by `std::lognormal_distribution< _RealType >::operator()()`.

5.543.3.4 `template<typename _RealType = double> void
std::lognormal_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

5.543 std::lognormal_distribution< _RealType > Class Template Reference 2653

Parameters

`__param` The new parameter set of the distribution.

Definition at line 2254 of file random.h.

```
5.543.3.5 template<typename _RealType = double> param_type  
std::lognormal_distribution< _RealType >::param ( ) const  
[inline]
```

Returns the parameter set of the distribution.

Definition at line 2246 of file random.h.

Referenced by std::lognormal_distribution< _RealType >::operator()().

```
5.543.3.6 template<typename _RealType = double> void  
std::lognormal_distribution< _RealType >::reset ( ) [inline]
```

Resets the distribution state.

Definition at line 2228 of file random.h.

References std::normal_distribution< _RealType >::reset().

5.543.4 Friends And Related Function Documentation

```
5.543.4.1 template<typename _RealType = double> template<typename  
_RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream<_CharT, _Traits>& operator<<  
( std::basic_ostream< _CharT, _Traits > & , const  
std::lognormal_distribution< _RealType1 > & ) [friend]
```

Inserts a lognormal_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A lognormal_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.543.4.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::lognormal_distribution<
_RealType1 > & __d1, const std::lognormal_distribution<
_RealType1 > & __d2) [friend]`

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2292 of file random.h.

5.543.4.3 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream< _CharT, _Traits > & ,
std::lognormal_distribution< _RealType1 > &) [friend]`

Extracts a lognormal_distribution random number distribution __x from the input stream __is.

Parameters

__is An input stream.

__x A lognormal_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

5.544 std::lognormal_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef [lognormal_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m=_RealType(0), _RealType __s=_RealType(1))
- _RealType **m** () const
- _RealType **s** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.544.1 Detailed Description

template<typename _RealType = double> struct std::lognormal_distribution<_RealType >::param_type

Parameter type.

Definition at line 2186 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.545 std::make_signed< _Tp > Struct Template Reference

[make_signed](#)

Public Types

- typedef __make_signed_selector< _Tp >::__type **type**

5.545.1 Detailed Description

template<typename _Tp> struct std::make_signed< _Tp >

[make_signed](#)

Definition at line 644 of file type_traits.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.546 std::make_unsigned< _Tp > Struct Template Reference

[make_unsigned](#)

Public Types

- typedef `__make_unsigned_selector< _Tp >::__type` **type**

5.546.1 Detailed Description

`template<typename _Tp> struct std::make_unsigned< _Tp >`

[make_unsigned](#)

Definition at line 567 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.547 `std::map< _Key, _Tp, _Compare, _Alloc >` Class Template Reference

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Rep_type::const_iterator` **const_iterator**
- typedef `_Pair_alloc_type::const_pointer` **const_pointer**
- typedef `_Pair_alloc_type::const_reference` **const_reference**
- typedef `_Rep_type::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Rep_type::difference_type` **difference_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Pair_alloc_type::pointer` **pointer**
- typedef `_Pair_alloc_type::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse_iterator**
- typedef `_Rep_type::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- [map](#) ()
- [map](#) (const _Compare &__comp, const allocator_type &__a=allocator_type())
- [map](#) (map &&__x)
- [map](#) (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())
- [map](#) (const map &__x)
- template<typename _InputIterator >
[map](#) (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator >
[map](#) (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- mapped_type & [at](#) (const key_type &__k)
- const mapped_type & [at](#) (const key_type &__k) const
- iterator [begin](#) ()
- const_iterator [begin](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [cend](#) () const
- void [clear](#) ()
- size_type [count](#) (const key_type &__x) const
- const_reverse_iterator [crbegin](#) () const
- const_reverse_iterator [crend](#) () const
- bool [empty](#) () const
- iterator [end](#) ()
- const_iterator [end](#) () const
- std::pair< const_iterator, const_iterator > [equal_range](#) (const key_type &__x) const
- std::pair< iterator, iterator > [equal_range](#) (const key_type &__x)
- iterator [erase](#) (iterator __position)
- size_type [erase](#) (const key_type &__x)
- iterator [erase](#) (iterator __first, iterator __last)
- iterator [find](#) (const key_type &__x)
- const_iterator [find](#) (const key_type &__x) const
- allocator_type [get_allocator](#) () const
- template<typename _InputIterator >
void [insert](#) (_InputIterator __first, _InputIterator __last)
- iterator [insert](#) (iterator __position, const value_type &__x)
- std::pair< iterator, bool > [insert](#) (const value_type &__x)
- void [insert](#) (std::initializer_list< value_type > __list)
- key_compare [key_comp](#) () const
- const_iterator [lower_bound](#) (const key_type &__x) const
- iterator [lower_bound](#) (const key_type &__x)

- `size_type max_size () const`
- `map & operator= (map &&__x)`
- `map & operator= (const map &__x)`
- `map & operator= (initializer_list< value_type > __l)`
- `mapped_type & operator[] (const key_type &__k)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `size_type size () const`
- `void swap (map &__x)`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `value_compare value_comp () const`

Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool operator< (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)`
- `template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool operator== (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)`

5.547.1 Detailed Description

`template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> class std::map< _Key, _Tp, _Compare, _Alloc >`

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 86 of file `stl_map.h`.

5.547.2 Constructor & Destructor Documentation

5.547.2.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map () [inline]`

Default constructor creates no elements.

Definition at line 150 of file stl_map.h.

5.547.2.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map (const _Compare & __comp, const allocator_type & __a = allocator_type()) [inline, explicit]`

Creates a map with no elements.

Parameters

comp A comparison object.

a An allocator object.

Definition at line 159 of file stl_map.h.

5.547.2.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map< _Key, _Tp, _Compare, _Alloc >::map (const map< _Key, _Tp, _Compare, _Alloc > & __x) [inline]`

Map copy constructor.

Parameters

x A map of identical element and allocator types.

The newly-created map uses a copy of the allocation object used by *x*.

Definition at line 170 of file stl_map.h.

5.547.2.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map (map<_Key, _Tp, _Compare, _Alloc> && __x) [inline]`

Map move constructor.

Parameters

x A map of identical element and allocator types.

The newly-created map contains the exact contents of *x*. The contents of *x* are a valid, but unspecified map.

Definition at line 181 of file `stl_map.h`.

5.547.2.5 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map (initializer_list<value_type> & __l, const _Compare & __c = _Compare(), const allocator_type & __a = allocator_type()) [inline]`

Builds a map from an [initializer_list](#).

Parameters

l An [initializer_list](#).

comp A comparison object.

a An allocator object.

Create a map consisting of copies of the elements in the [initializer_list](#) *l*. This is linear in *N* if the range is already sorted, and *NlogN* otherwise (where *N* is *l.size()*).

Definition at line 195 of file `stl_map.h`.

5.547.2.6 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::map<_Key, _Tp, _Compare, _Alloc>::map (_InputIterator __first, _InputIterator __last) [inline]`

Builds a map from a range.

Parameters

first An input iterator.

5.547 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2661

last An input iterator.

Create a map consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 212 of file stl_map.h.

```
5.547.2.7 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> template<typename _InputIterator > std::map<
_Key, _Tp, _Compare, _Alloc >::map ( _InputIterator __first,
_InputIterator __last, const _Compare & __comp, const
allocator_type & __a = allocator_type() ) [inline]
```

Builds a map from a range.

Parameters

first An input iterator.

last An input iterator.

comp A comparison functor.

a An allocator object.

Create a map consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 228 of file stl_map.h.

5.547.3 Member Function Documentation

```
5.547.3.1 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> mapped_type& std::map< _Key, _Tp, _Compare,
_Alloc >::at ( const key_type & __k ) [inline]
```

Access to map data.

Parameters

k The key for which data should be retrieved.

Returns

A reference to the data whose key is equivalent to *k*, if such a data is present in the map.

Exceptions

std::out_of_range If no such data is present.

Definition at line 465 of file stl_map.h.

References `std::end()`.

5.547.3.2 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc
>::begin () [inline]`

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 307 of file stl_map.h.

5.547.3.3 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare,
_Alloc>::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 316 of file stl_map.h.

5.547.3.4 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare,
_Alloc>::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 380 of file stl_map.h.

5.547.3.5 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare,
_Alloc>::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

5.547 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2663

Definition at line 389 of file stl_map.h.

5.547.3.6 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> void std::map< _Key, _Tp, _Compare, _Alloc
>::clear () [inline]`

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 663 of file stl_map.h.

5.547.3.7 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc
>::count (const key_type & __x) const [inline]`

Finds the number of elements with given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 723 of file stl_map.h.

5.547.3.8 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp,
_Compare, _Alloc >::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 398 of file stl_map.h.

5.547.3.9 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp,
_Compare, _Alloc >::crend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 407 of file `stl_map.h`.

5.547.3.10 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> bool std::map< _Key, _Tp, _Compare, _Alloc
>::empty () const [inline]`

Returns true if the map is empty. (Thus `begin()` would equal `end()`.)

Definition at line 416 of file `stl_map.h`.

5.547.3.11 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 334 of file `stl_map.h`.

5.547.3.12 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc
>::end () [inline]`

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 325 of file `stl_map.h`.

5.547 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2665

5.547.3.13 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> std::pair<iterator, iterator> std::map< _Key,
_Tp, _Compare, _Alloc >::equal_range (const key_type & __x)
[inline]`

Finds a subsequence matching given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 792 of file stl_map.h.

5.547.3.14 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> std::pair<const_iterator, const_iterator>
std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (const
key_type & __x) const [inline]`

Finds a subsequence matching given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 811 of file `stl_map.h`.

```
5.547.3.15  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc
              >::erase ( iterator __position ) [inline]
```

Erases an element from a map.

Parameters

position An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 573 of file `stl_map.h`.

```
5.547.3.16  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> size_type std::map<_Key, _Tp, _Compare, _Alloc
              >::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

x Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 603 of file `stl_map.h`.

5.547 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2667

5.547.3.17 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc
>::erase (iterator __first, iterator __last) [inline]`

Erases a [first,last) range of elements from a map.

Parameters

first Iterator pointing to the start of the range to be erased.

last Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 622 of file stl_map.h.

5.547.3.18 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::find (const key_type & __x) const [inline]`

Tries to locate an element in a map.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 711 of file stl_map.h.

5.547.3.19 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::find (const key_type & __x) [inline]`

Tries to locate an element in a map.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 696 of file `stl_map.h`.

5.547.3.20 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::map<_Key, _Tp, _Compare, _Alloc>::get_allocator () const [inline]`

Get a copy of the memory allocation object.

Definition at line 297 of file `stl_map.h`.

5.547.3.21 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc>::insert (const value_type & __x) [inline]`

Attempts to insert a `std::pair` into the map.

Parameters

x Pair to be inserted (see `std::make_pair` for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

5.547 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2669

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 500 of file stl_map.h.

```
5.547.3.22 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> void std::map< _Key, _Tp, _Compare, _Alloc  
>::insert ( std::initializer_list< value_type > __list ) [inline]
```

Attempts to insert a list of std::pairs into the map.

Parameters

list A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 512 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::insert().

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::insert().

```
5.547.3.23 template<typename _Key, typename _Tp, typename _Compare =  
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
_Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc  
>::insert ( iterator __position, const value_type & __x )  
[inline]
```

Attempts to insert a [std::pair](#) into the map.

Parameters

position An iterator that serves as a hint as to where the pair should be inserted.

x Pair to be inserted (see [std::make_pair](#) for easy creation of pairs).

Returns

An iterator that points to the element with key of *x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument [insert\(\)](#) does. Note that the first parameter

is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 540 of file `stl_map.h`.

```
5.547.3.24  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> template<typename _InputIterator > void
              std::map<_Key, _Tp, _Compare, _Alloc >::insert ( _InputIterator
              __first, _InputIterator __last ) [inline]
```

Template function that attempts to insert a range of elements.

Parameters

first Iterator pointing to the start of the range to be inserted.

last Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 553 of file `stl_map.h`.

```
5.547.3.25  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> key_compare std::map<_Key, _Tp, _Compare,
              _Alloc >::key_comp ( ) const [inline]
```

Returns the key comparison object out of which the map was constructed.

Definition at line 672 of file `stl_map.h`.

```
5.547.3.26  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc
              >::lower_bound ( const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

5.547 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2671

Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 738 of file `stl_map.h`.

```
5.547.3.27  template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::lower_bound ( const key_type & __x ) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or `end()` if no such element exists.

Definition at line 753 of file `stl_map.h`.

```
5.547.3.28  template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc
>::max_size ( ) const  [inline]
```

Returns the maximum size of the map.

Definition at line 426 of file `stl_map.h`.

```
5.547.3.29  template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> map& std::map< _Key, _Tp, _Compare, _Alloc
>::operator= ( map< _Key, _Tp, _Compare, _Alloc > && __x )
[inline]
```

Map move assignment operator.

Parameters

x A map of identical element and allocator types.

The contents of *x* are moved into this map (without copying). *x* is a valid, but unspecified map.

Definition at line 266 of file `stl_map.h`.

```
5.547.3.30  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> map& std::map<_Key, _Tp, _Compare, _Alloc
              >::operator= ( const map<_Key, _Tp, _Compare, _Alloc> & __x
              ) [inline]
```

Map assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

x A map of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 251 of file `stl_map.h`.

```
5.547.3.31  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> map& std::map<_Key, _Tp, _Compare, _Alloc
              >::operator= ( initializer_list<value_type> & __l ) [inline]
```

Map list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a map with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 287 of file `stl_map.h`.

5.547 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2673

5.547.3.32 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> mapped_type& std::map< _Key, _Tp, _Compare,
_Alloc >::operator[] (const key_type & __k) [inline]`

Subscript ([]) access to map data.

Parameters

k The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([]) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 443 of file stl_map.h.

References std::end().

5.547.3.33 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> reverse_iterator std::map< _Key, _Tp, _Compare,
_Alloc >::rbegin () [inline]`

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 343 of file stl_map.h.

5.547.3.34 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp,
_Compare, _Alloc >::rbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 352 of file stl_map.h.

5.547.3.35 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 370 of file `stl_map.h`.

5.547.3.36 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc>::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 361 of file `stl_map.h`.

5.547.3.37 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::map<_Key, _Tp, _Compare, _Alloc>::size () const [inline]`

Returns the size of the map.

Definition at line 421 of file `stl_map.h`.

5.547.3.38 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::map<_Key, _Tp, _Compare, _Alloc>::swap (map<_Key, _Tp, _Compare, _Alloc> & __x) [inline]`

Swaps data with another map.

Parameters

x A map of the same element and allocator types.

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

5.547 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference 2675

Definition at line 653 of file stl_map.h.

Referenced by std::swap().

```
5.547.3.39  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare,  
             _Alloc >::upper_bound ( const key_type & __x ) const  
             [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or [end\(\)](#).

Definition at line 773 of file stl_map.h.

```
5.547.3.40  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc  
             >::upper_bound ( const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 763 of file stl_map.h.

```
5.547.3.41  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> value_compare std::map< _Key, _Tp, _Compare,  
             _Alloc >::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 680 of file `stl_map.h`.

The documentation for this class was generated from the following file:

- [stl_map.h](#)

5.548 `std::mask_array< _Tp >` Class Template Reference

Reference to selected subset of an array.

Public Types

- `typedef _Tp value_type`

Public Member Functions

- `mask_array` (const `mask_array` &)
- `template<class _Dom >`
void **operator**%= (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator`%= (const `valarray`< `_Tp` > &) const
- void `operator`&= (const `valarray`< `_Tp` > &) const
- `template<class _Dom >`
void **operator**&= (const `_Expr`< `_Dom`, `_Tp` > &) const
- `template<class _Dom >`
void **operator***= (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator`*= (const `valarray`< `_Tp` > &) const
- `template<class _Dom >`
void **operator**+= (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator`+= (const `valarray`< `_Tp` > &) const
- void `operator`-= (const `valarray`< `_Tp` > &) const
- `template<class _Dom >`
void **operator**-= (const `_Expr`< `_Dom`, `_Tp` > &) const
- `template<class _Dom >`
void **operator**/= (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator`/= (const `valarray`< `_Tp` > &) const
- void `operator`<<= (const `valarray`< `_Tp` > &) const
- `template<class _Dom >`
void **operator**<<= (const `_Expr`< `_Dom`, `_Tp` > &) const
- `template<class _Dom >`
void **operator**= (const `_Expr`< `_Dom`, `_Tp` > &) const

- `mask_array` & `operator=` (const `mask_array` &)
- void `operator=` (const `valarray`< _Tp > &) const
- template<class _Ex >
void `operator=` (const _Expr< _Ex, _Tp > &__e) const
- void `operator=` (const _Tp &) const
- void `operator>>=` (const `valarray`< _Tp > &) const
- template<class _Dom >
void `operator>>=` (const _Expr< _Dom, _Tp > &) const
- void `operator^=` (const `valarray`< _Tp > &) const
- template<class _Dom >
void `operator^=` (const _Expr< _Dom, _Tp > &) const
- void `operator|=` (const `valarray`< _Tp > &) const
- template<class _Dom >
void `operator|=` (const _Expr< _Dom, _Tp > &) const

Friends

- class `valarray`< _Tp >

5.548.1 Detailed Description

`template<class _Tp> class std::mask_array< _Tp >`

Reference to selected subset of an array. A `mask_array` is a reference to the actual elements of an array specified by a bitmask in the form of an array of bool. The way to get a `mask_array` is to call `operator[]`(`valarray`<bool>) on a `valarray`. The returned `mask_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if a `mask_array` is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to `array[1]` and `array[3]` in the underlying array.

Parameters

Tp Element type.

Definition at line 61 of file `mask_array.h`.

5.548.2 Member Function Documentation

5.548.2.1 `template<class _Tp> void std::mask_array< _Tp >::operator%= (const valarray< _Tp > &) const`

Modulo slice elements by corresponding elements of v.

5.548.2.2 `template<class _Tp> void std::mask_array< _Tp >::operator&= (const valarray< _Tp > &) const`

Logical and slice elements with corresponding elements of *v*.

5.548.2.3 `template<class _Tp> void std::mask_array< _Tp >::operator*= (const valarray< _Tp > &) const`

Multiply slice elements by corresponding elements of *v*.

5.548.2.4 `template<class _Tp> void std::mask_array< _Tp >::operator+= (const valarray< _Tp > &) const`

Add corresponding elements of *v* to slice elements.

5.548.2.5 `template<class _Tp> void std::mask_array< _Tp >::operator-= (const valarray< _Tp > &) const`

Subtract corresponding elements of *v* from slice elements.

5.548.2.6 `template<class _Tp> void std::mask_array< _Tp >::operator/= (const valarray< _Tp > &) const`

Divide slice elements by corresponding elements of *v*.

5.548.2.7 `template<class _Tp> void std::mask_array< _Tp >::operator<<= (const valarray< _Tp > &) const`

Left shift slice elements by corresponding elements of *v*.

5.548.2.8 `template<class _Tp> void std::mask_array< _Tp >::operator>>= (const valarray< _Tp > &) const`

Right shift slice elements by corresponding elements of *v*.

5.548.2.9 `template<class _Tp> void std::mask_array< _Tp >::operator^= (const valarray< _Tp > &) const`

Logical xor slice elements with corresponding elements of *v*.

5.549 std::match_results< _Bi_iter, _Allocator > Class Template Reference 2679

5.548.2.10 `template<class _Tp> void std::mask_array< _Tp >::operator|= (const valarray< _Tp > &) const`

Logical or slice elements with corresponding elements of v.

The documentation for this class was generated from the following file:

- [mask_array.h](#)

5.549 std::match_results< _Bi_iter, _Allocator > Class Template Reference

The results of a match or search operation.

Inheritance diagram for std::match_results< _Bi_iter, _Allocator >:



Private Types

- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**

Private Member Functions

- `_Tp_alloc_type::pointer` **M_allocate** (size_t __n)
- `pointer` **M_allocate_and_copy** (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- void **M_assign_aux** (_InputIterator __first, _InputIterator __last, `std::input_iterator_tag`)
- void **M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, `std::forward_iterator_tag`)
- void **M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- size_type **M_check_len** (size_type __n, const char *__s) const
- void **M_deallocate** (typename _Tp_alloc_type::pointer __p, size_t __n)
- void **M_default_append** (size_type __n)

- void **_M_default_initialize** (size_type __n)
- void **_M_erase_at_end** (pointer __pos)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__value)
- void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- const_Tp_alloc_type & **_M_get_Tp_allocator** () const
- _Tp_alloc_type & **_M_get_Tp_allocator** ()
- void **_M_initialize_dispatch** (_Integer __n, _Integer __value, __true_type)
- void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_insert_aux** (iterator __position, _Args &&...__args)
- void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __val, __true_type)
- void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- void **_M_range_check** (size_type __n) const
- void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- void **_M_range_insert** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- void **_M_range_insert** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const value_type &__val)
- void **assign** (initializer_list< value_type > __l)
- **reference at** (size_type __n)
- **const_reference at** (size_type __n) const
- **reference back** ()
- **const_reference back** () const
- iterator **begin** ()
- size_type **capacity** () const
- void **clear** ()
- **const_reverse_iterator crbegin** () const
- **const_reverse_iterator crend** () const
- **std::sub_match< _Bi_iter > * data** ()
- const **std::sub_match< _Bi_iter > * data** () const
- iterator **emplace** (iterator __position, _Args &&...__args)
- void **emplace_back** (_Args &&...__args)
- iterator **end** ()
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __first, iterator __last)

- `reference front ()`
- `const_reference front () const`
- `void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `iterator insert (iterator __position, value_type &&__x)`
- `void insert (iterator __position, size_type __n, const value_type &__x)`
- `iterator insert (iterator __position, const value_type &__x)`
- `void insert (iterator __position, initializer_list< value_type > __l)`
- `reference operator[] (size_type __n)`
- `const_reference operator[] (size_type __n) const`
- `void pop_back ()`
- `void push_back (const value_type &__x)`
- `void push_back (value_type &&__x)`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rbegin ()`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `void reserve (size_type __n)`
- `void resize (size_type __new_size, const value_type &__x)`
- `void resize (size_type __new_size)`
- `void shrink_to_fit ()`
- `void swap (vector &__x)`

Private Attributes

- `_Vector_impl _M_impl`

Friends

- `class __regex::_SpecializedResults<_Bi_iter, _Allocator>`

10.? Public Types

- `typedef sub_match<_Bi_iter> value_type`
- `typedef const value_type & const_reference`
- `typedef const_reference reference`
- `typedef _Base_type::const_iterator const_iterator`
- `typedef const_iterator iterator`
- `typedef std::iterator_traits<_Bi_iter>::difference_type difference_type`
- `typedef _Allocator::size_type size_type`
- `typedef _Allocator allocator_type`
- `typedef std::iterator_traits<_Bi_iter>::value_type char_type`
- `typedef std::basic_string< char_type > string_type`

10.1 Construction, Copying, and Destruction

- `match_results` (`const _Allocator &__a=_Allocator()`)
- `match_results` (`const match_results &__rhs`)
- `match_results & operator=` (`const match_results __rhs`)
- `~match_results` ()

10.2 Size

- `size_type size` () `const`
- `size_type max_size` () `const`
- `bool empty` () `const`

10.3 Element Access

- `difference_type length` (`size_type __sub=0`) `const`
- `difference_type position` (`size_type __sub=0`) `const`
- `string_type str` (`size_type __sub=0`) `const`
- `const_reference operator[]` (`size_type __sub`) `const`
- `const_reference prefix` () `const`
- `const_reference suffix` () `const`
- `const_iterator begin` () `const`
- `const_iterator cbegin` () `const`
- `const_iterator end` () `const`
- `const_iterator cend` () `const`

10.4 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- `template<typename _Out_iter >`
`_Out_iter format` (`_Out_iter __out, const string_type &__fmt, regex_constants::match_flag_type __flags=regex_constants::format_default`) `const`
- `string_type format` (`const string_type &__fmt, regex_constants::match_flag_type __flags=regex_constants::format_default`) `const`

10.5 Allocator

- `allocator_type get_allocator` () `const`

10.6 Swap

- void `swap` (`match_results` &__that)

5.549.1 Detailed Description

```
template<typename _Bi_iter, typename _Allocator = allocator<sub_match<_Bi_iter>>>> class std::match_results< _Bi_iter, _Allocator >
```

The results of a match or search operation. A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template `match_results`.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The `sub_match` object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the `sub_match` member `matched` is always true. The `sub_match` object stored at index `n` denotes what matched the marked sub-expression `n` within the matched expression. If the sub-expression `n` participated in a regular expression match then the `sub_match` member `matched` evaluates to true, and members `first` and `second` denote the range of characters [`first`, `second`) which formed that match. Otherwise `matched` is false, and members `first` and `second` point to the end of the sequence that was searched.

Definition at line 1446 of file `regex.h`.

5.549.2 Constructor & Destructor Documentation

```
5.549.2.1 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter>>>> std::match_results<
_Bi_iter, _Allocator >::match_results ( const _Allocator & __a =
_Allocator() ) [inline, explicit]
```

Constructs a default `match_results` container.

Postcondition

`size()` returns 0 and `str()` returns an empty string.

Definition at line 1495 of file `regex.h`.

Referenced by `std::match_results< _FwdIterT, _Alloc >::operator=()`.

5.549.2.2 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter,
_Allocator>::match_results (const match_results<_Bi_iter,
_Allocator> & __rhs) [inline]`

Copy constructs a match_results.

Definition at line 1502 of file regex.h.

5.549.2.3 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter,
_Allocator>::~~match_results () [inline]`

Destroys a match_results object.

Definition at line 1519 of file regex.h.

5.549.3 Member Function Documentation

5.549.3.1 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter>>> const_iterator
std::match_results<_Bi_iter, _Allocator>::begin () const
[inline]`

Gets an iterator to the start of the sub_match collection.

Reimplemented from [std::vector< std::sub_match<_Bi_iter>, _Allocator >](#).

Definition at line 1658 of file regex.h.

5.549.3.2 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter>>> const_iterator
std::match_results<_Bi_iter, _Allocator>::cbegin () const
[inline]`

Gets an iterator to the start of the sub_match collection.

Reimplemented from [std::vector< std::sub_match<_Bi_iter>, _Allocator >](#).

Definition at line 1665 of file regex.h.

5.549 std::match_results< _Bi_iter, _Allocator > Class Template Reference 2685

5.549.3.3 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_iterator
std::match_results< _Bi_iter, _Allocator >::cend () const
[inline]`

Gets an iterator to one-past-the-end of the collection.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1683 of file regex.h.

5.549.3.4 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> bool std::match_results<
_Bi_iter, _Allocator >::empty () const [inline]`

Indicates if the match_results contains no results.

Return values

true The match_results object is empty.

false The match_results object is not empty.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1555 of file regex.h.

Referenced by `std::match_results< _FwdIterT, _Alloc >::cend()`, `std::match_results< _FwdIterT, _Alloc >::end()`, `std::match_results< _FwdIterT, _Alloc >::prefix()`, and `std::match_results< _FwdIterT, _Alloc >::suffix()`.

5.549.3.5 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_iterator
std::match_results< _Bi_iter, _Allocator >::end () const
[inline]`

Gets an iterator to one-past-the-end of the collection.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1672 of file regex.h.

```

5.549.3.6 template<typename _Bi_iter, typename _Allocator
= allocator<sub_match<_Bi_iter> >> string_type
std::match_results< _Bi_iter, _Allocator >::format ( const
string_type & __fmt, regex_constants::match_flag_type __flags =
regex_constants::format_default ) const

```

Todo

Implement this function.

```

5.549.3.7 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> template<typename
_Out_iter > _Out_iter std::match_results< _Bi_iter,
_Alocator >::format ( _Out_iter __out, const string_type
& __fmt, regex_constants::match_flag_type __flags =
regex_constants::format_default ) const [inline]

```

Todo

Implement this function.

Definition at line 1707 of file regex.h.

```

5.549.3.8 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> allocator_type
std::match_results< _Bi_iter, _Allocator >::get_allocator ( ) const
[inline]

```

Gets a copy of the allocator.

Reimplemented from [std::_Vector_base< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1731 of file regex.h.

```

5.549.3.9 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> difference_type
std::match_results< _Bi_iter, _Allocator >::length ( size_type __sub
= 0 ) const [inline]

```

Gets the length of the indicated submatch.

Parameters

sub indicates the submatch.

5.549 std::match_results< _Bi_iter, _Allocator > Class Template Reference 2687

This function returns the length of the indicated submatch, or the length of the entire match if `sub` is zero (the default).

Definition at line 1573 of file `regex.h`.

```
5.549.3.10  template<typename _Bi_iter, typename _Allocator =  
              allocator<sub_match<_Bi_iter> >> size_type std::match_results<  
              _Bi_iter, _Allocator >::max_size( ) const  [inline]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count()` + 1 if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Reimplemented from [std::vector< std::sub_match< _Bi_iter >, _Allocator >](#).

Definition at line 1546 of file `regex.h`.

```
5.549.3.11  template<typename _Bi_iter, typename _Allocator =  
              allocator<sub_match<_Bi_iter> >> match_results&  
              std::match_results< _Bi_iter, _Allocator >::operator=( const  
              match_results< _Bi_iter, _Allocator > __rhs )  [inline]
```

Assigns `rhs` to `*this`.

Definition at line 1510 of file `regex.h`.

```
5.549.3.12  template<typename _Bi_iter, typename _Allocator =  
              allocator<sub_match<_Bi_iter> >> const_reference  
              std::match_results< _Bi_iter, _Allocator >::operator[] ( size_type  
              __sub ) const  [inline]
```

Gets a `sub_match` reference for the match or submatch.

Parameters

sub indicates the submatch.

This function gets a reference to the indicated submatch, or the entire match if `sub` is zero.

If `sub >= size()` then this function returns a `sub_match` with a special value indicating no submatch.

Definition at line 1617 of file `regex.h`.

5.549.3.13 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> difference_type
std::match_results< _Bi_iter, _Allocator >::position (size_type
__sub = 0) const [inline]`

Gets the offset of the beginning of the indicated submatch.

Parameters

sub indicates the submatch.

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of *sub* is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

Returns -1 if *sub* is out of range.

Definition at line 1589 of file `regex.h`.

5.549.3.14 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_reference
std::match_results< _Bi_iter, _Allocator >::prefix () const
[inline]`

Gets a `sub_match` representing the match prefix.

This function gets a reference to a `sub_match` object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1632 of file `regex.h`.

Referenced by `std::match_results< _FwdIterT, _Alloc >::position()`.

5.549.3.15 `template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> size_type std::match_results<
_Bi_iter, _Allocator >::size () const [inline]`

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Reimplemented from `std::vector< std::sub_match< _Bi_iter >, _Allocator >`.

5.549 std::match_results< _Bi_iter, _Allocator > Class Template Reference 2689

Definition at line 1539 of file regex.h.

Referenced by std::match_results< _FwdIterT, _Alloc >::operator[](), and std::match_results< _FwdIterT, _Alloc >::position().

```
5.549.3.16 template<typename _Bi_iter, typename _Allocator
= allocator<sub_match<_Bi_iter> >> string_type
std::match_results< _Bi_iter, _Allocator >::str ( size_type __sub =
0 ) const [inline]
```

Gets the match or submatch converted to a string type.

Parameters

sub indicates the submatch.

This function gets the submatch (or match, if *sub* is zero) extracted from the target range and converted to the associated string type.

Definition at line 1603 of file regex.h.

```
5.549.3.17 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> const_reference
std::match_results< _Bi_iter, _Allocator >::suffix ( ) const
[inline]
```

Gets a sub_match representing the match suffix.

This function gets a reference to a sub_match object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1647 of file regex.h.

```
5.549.3.18 template<typename _Bi_iter, typename _Allocator =
allocator<sub_match<_Bi_iter> >> void std::match_results<
_Bi_iter, _Allocator >::swap ( match_results< _Bi_iter, _Allocator
> & __that ) [inline]
```

Swaps the contents of two [match_results](#).

Definition at line 1745 of file regex.h.

Referenced by std::swap().

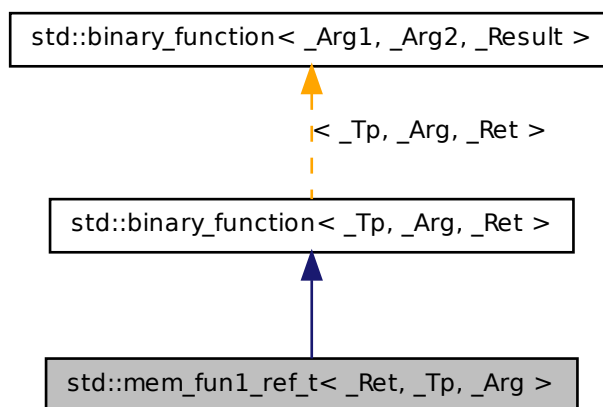
The documentation for this class was generated from the following file:

- [regex.h](#)

5.550 `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Ret` [result_type](#)
- typedef `_Arg` [second_argument_type](#)

Public Member Functions

- `mem_fun1_ref_t` (`_Ret` (`_Tp::*__pf`) (`_Arg`))
- `_Ret operator()` (`_Tp &__r, _Arg __x`) const

5.550.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`

One of the [adaptors for member /// pointers](#).

Definition at line 632 of file `stl_function.h`.

5.550.2 Member Typedef Documentation

5.550.2.1 `typedef _Tp std::binary_function<_Tp, _Arg, _Ret>::first_argument_type` [`inherited`]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.550.2.2 `typedef _Ret std::binary_function<_Tp, _Arg, _Ret>::result_type` [`inherited`]

type of the return type

Definition at line 118 of file `stl_function.h`.

5.550.2.3 `typedef _Arg std::binary_function<_Tp, _Arg, _Ret>::second_argument_type` [`inherited`]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

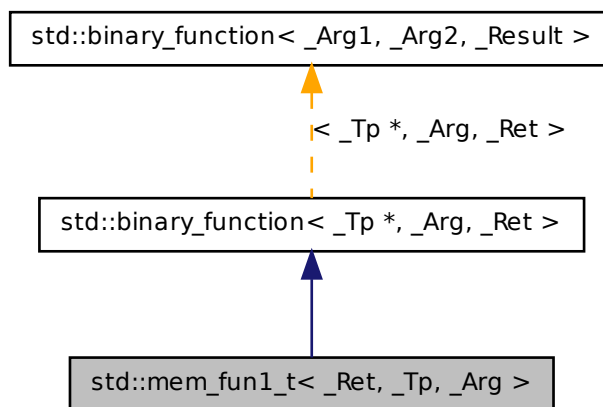
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.551 `std::mem_fun1_t<_Ret, _Tp, _Arg>` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::mem_fun1_t< _Ret, _Tp, _Arg >`:



Public Types

- typedef `_Tp *` [first_argument_type](#)
- typedef `_Ret` [result_type](#)
- typedef `_Arg` [second_argument_type](#)

Public Member Functions

- `mem_fun1_t` (`_Ret` (`_Tp::*__pf`) (`_Arg`))
- `_Ret operator()` (`_Tp *__p`, `_Arg __x`) `const`

5.551.1 Detailed Description

`template<typename _Ret, typename _Tp, typename _Arg> class std::mem_fun1_t< _Ret, _Tp, _Arg >`

One of the [adaptors for member /// pointers](#).

Definition at line 596 of file `stl_function.h`.

5.551.2 Member Typedef Documentation

5.551.2.1 `typedef _Tp * std::binary_function< _Tp *, _Arg , _Ret
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.551.2.2 `typedef _Ret std::binary_function< _Tp *, _Arg , _Ret
>::result_type [inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.551.2.3 `typedef _Arg std::binary_function< _Tp *, _Arg , _Ret
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

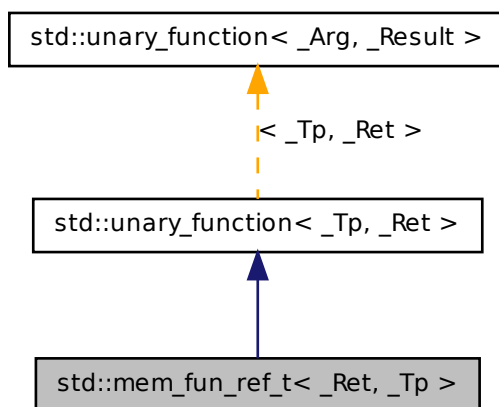
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.552 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::mem_fun_ref_t< _Ret, _Tp >`:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `mem_fun_ref_t` (`_Ret`(`_Tp`::*`__pf`)())
- `_Ret operator()` (`_Tp` &`__r`) const

5.552.1 Detailed Description

```
template<typename _Ret, typename _Tp> class std::mem_fun_ref_t< _Ret, _Tp
>
```

One of the [adaptors for member /// pointers](#).

Definition at line 560 of file `stl_function.h`.

5.552.2 Member Typedef Documentation

5.552.2.1 `typedef _Tp std::unary_function< _Tp, _Ret >::argument_type` [`inherited`]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.552.2.2 `typedef _Ret std::unary_function< _Tp, _Ret >::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

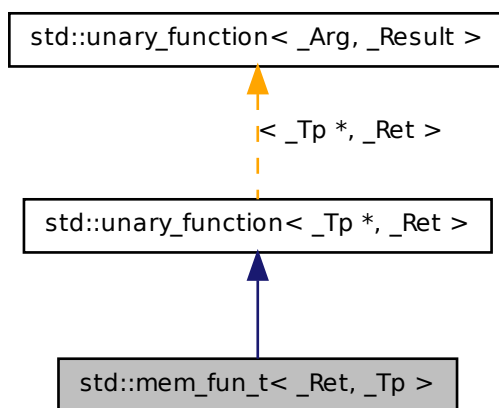
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.553 `std::mem_fun_t< _Ret, _Tp >` Class Template Reference

One of the [adaptors for member /// pointers](#).

Inheritance diagram for `std::mem_fun_t< _Ret, _Tp >`:



Public Types

- typedef `_Tp *` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `mem_fun_t` (`_Ret(_Tp::__pf)()`)
- `_Ret operator()` (`_Tp *__p`) `const`

5.553.1 Detailed Description

`template<typename _Ret, typename _Tp> class std::mem_fun_t< _Ret, _Tp >`

One of the [adaptors for member /// pointers](#).

Definition at line 524 of file `stl_function.h`.

5.553.2 Member Typedef Documentation

5.553.2.1 `typedef _Tp * std::unary_function<_Tp *, _Ret>::argument_type` [`inherited`]

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.553.2.2 `typedef _Ret std::unary_function<_Tp *, _Ret>::result_type` [`inherited`]

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

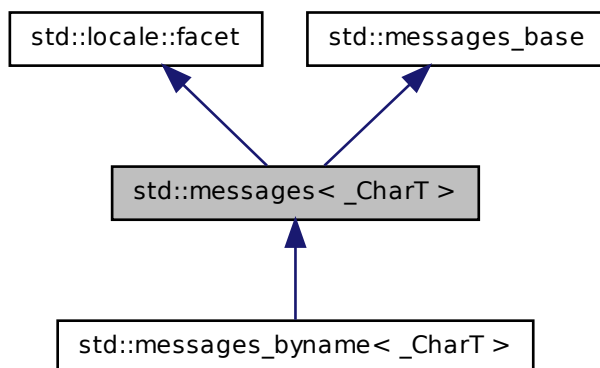
- [stl_function.h](#)

5.554 `std::messages<_CharT>` Class Template Reference

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

Inheritance diagram for `std::messages<_CharT>`:



Public Types

- typedef int **catalog**
- typedef `_CharT` **char_type**
- typedef `basic_string<_CharT>` **string_type**

Public Member Functions

- **messages** (size_t __refs=0)
- **messages** (__c_locale __cloc, const char *__s, size_t __refs=0)
- void **close** (catalog __c) const
- **string_type** **get** (catalog __c, int __set, int __msgid, const **string_type** &__s) const
- catalog **open** (const `basic_string<char>` &__s, const **locale** &__loc) const
- catalog **open** (const `basic_string<char>` &, const **locale** &, const char *) const

Static Public Attributes

- static **locale::id** **id**

Protected Member Functions

- virtual `~messages()`
- `string_type _M_convert_from_char` (char *) const
- char * `_M_convert_to_char` (const `string_type` &__msg) const
- virtual void `do_close` (catalog) const
- template<>
`string do_get` (catalog, int, int, const `string` &) const
- template<>
`wstring do_get` (catalog, int, int, const `wstring` &) const
- virtual `string_type do_get` (catalog, int, int, const `string_type` &__default) const
- virtual catalog `do_open` (const `basic_string`<char> &, const `locale` &) const

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (__c_locale &__cloc) throw ()
- static void `_S_create_c_locale` (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void `_S_destroy_c_locale` (__c_locale &__cloc)
- static `__c_locale _S_get_c_locale` ()
- static `_GLIBCXX_CONST` const char * `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (__c_locale __cloc, const char *__s)

Protected Attributes

- `__c_locale _M_c_locale_messages`
- const char * `_M_name_messages`

Friends

- class `locale::_Impl`

5.554.1 Detailed Description

`template<typename _CharT> class std::messages<_CharT>`

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined. This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The

second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar_t instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

Definition at line 1690 of file locale_facets_nonio.h.

5.554.2 Member Typedef Documentation

5.554.2.1 `template<typename _CharT > typedef _CharT std::messages<_CharT >::char_type`

Public typedefs.

Reimplemented in [std::messages_byname<_CharT >](#).

Definition at line 1696 of file locale_facets_nonio.h.

5.554.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::messages<_CharT >::string_type`

Public typedefs.

Reimplemented in [std::messages_byname<_CharT >](#).

Definition at line 1697 of file locale_facets_nonio.h.

5.554.3 Constructor & Destructor Documentation

5.554.3.1 `template<typename _CharT > std::messages<_CharT >::messages (size_t __refs = 0) [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 43 of file messages_members.h.

5.554.3.2 `template<typename _CharT > std::messages<_CharT >::messages
(__c_locale __cloc, const char * __s, size_t __refs = 0)
[explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

cloc The C locale.

s The name of a locale.

refs Refcount to pass to the base class.

Definition at line 49 of file messages_members.h.

5.554.3.3 `template<typename _CharT > std::messages<_CharT
>::~messages() [protected, virtual]`

Destructor.

Definition at line 78 of file messages_members.h.

5.554.4 Member Data Documentation

5.554.4.1 `template<typename _CharT > locale::id std::messages<_CharT
>::id [static]`

Numpunct facet id.

Definition at line 1708 of file locale_facets_nonio.h.

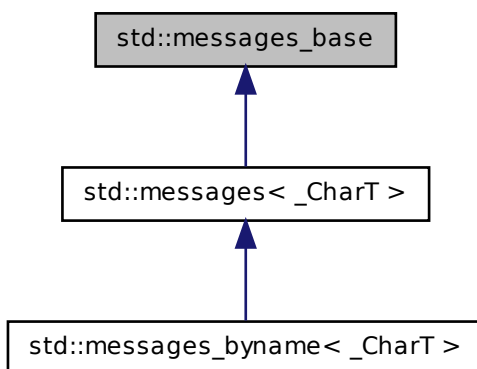
The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [messages_members.h](#)

5.555 std::messages_base Struct Reference

Messages facet base class providing catalog typedef.

Inheritance diagram for `std::messages_base`:



Public Types

- typedef int **catalog**

5.555.1 Detailed Description

Messages facet base class providing catalog typedef.

Definition at line 1663 of file `locale_facets_nonio.h`.

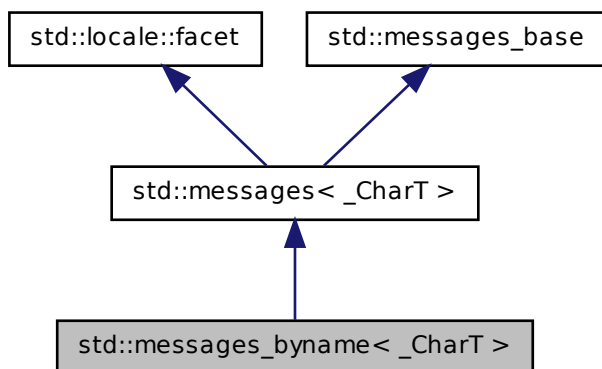
The documentation for this struct was generated from the following file:

- [locale_facets_nonio.h](#)

5.556 `std::messages_byname<_CharT>` Class Template Reference

class [messages_byname](#) [22.2.7.2].

Inheritance diagram for std::messages_byname< _CharT >:



Public Types

- typedef int **catalog**
- typedef `_CharT` **char_type**
- typedef `basic_string< _CharT >` **string_type**

Public Member Functions

- **messages_byname** (const char *__s, size_t __refs=0)
- void **close** (catalog __c) const
- **string_type** **get** (catalog __c, int __set, int __msgid, const **string_type** &__s) const
- catalog **open** (const **basic_string**< char > &, const **locale** &, const char *) const
- catalog **open** (const **basic_string**< char > &__s, const **locale** &__loc) const

Static Public Attributes

- static **locale::id** **id**

Protected Member Functions

- [string_type](#) **_M_convert_from_char** (char *) const
- char * **_M_convert_to_char** (const [string_type](#) &__msg) const
- virtual void **do_close** (catalog) const
- template<>
[wstring](#) **do_get** (catalog, int, int, const [wstring](#) &) const
- template<>
[string](#) **do_get** (catalog, int, int, const [string](#) &) const
- virtual [string_type](#) **do_get** (catalog, int, int, const [string_type](#) &__default) const
- virtual catalog **do_open** (const [basic_string](#)< char > &, const [locale](#) &) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_messages**
- const char * **_M_name_messages**

Friends

- class **locale::_Impl**

5.556.1 Detailed Description

template<typename _CharT> class std::messages_byname< _CharT >

class [messages_byname](#) [22.2.7.2].

Definition at line 1907 of file locale_facets_nonio.h.

5.556.2 Member Typedef Documentation

5.556.2.1 `template<typename _CharT > typedef _CharT
std::messages_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::messages< _CharT >](#).

Definition at line 1910 of file locale_facets_nonio.h.

5.556.2.2 `template<typename _CharT > typedef basic_string<_CharT>
std::messages_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::messages< _CharT >](#).

Definition at line 1911 of file locale_facets_nonio.h.

5.556.3 Member Data Documentation

5.556.3.1 `template<typename _CharT > locale::id std::messages< _CharT
>::id [static, inherited]`

Numpunct facet id.

Definition at line 1708 of file locale_facets_nonio.h.

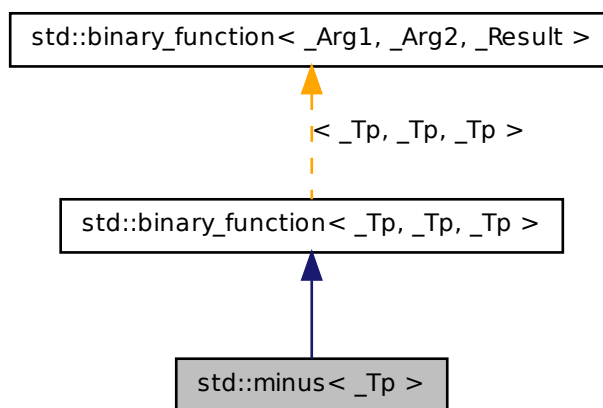
The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [messages_members.h](#)

5.557 std::minus< _Tp > Struct Template Reference

One of the [math functors](#).

Inheritance diagram for `std::minus<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

5.557.1 Detailed Description

`template<typename _Tp> struct std::minus<_Tp>`

One of the [math functors](#).

Definition at line 144 of file `stl_function.h`.

5.557.2 Member Typedef Documentation

5.557.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type` [inherited]

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.557.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type` [inherited]

type of the return type

Definition at line 118 of file stl_function.h.

5.557.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type` [inherited]

the type of the second argument

Definition at line 117 of file stl_function.h.

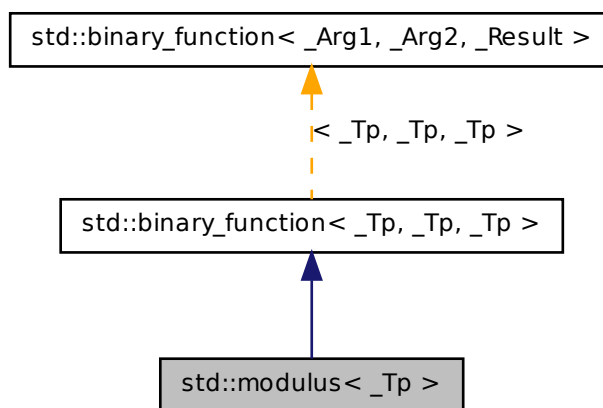
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.558 std::modulus< _Tp > Struct Template Reference

One of the [math functors](#).

Inheritance diagram for `std::modulus<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

5.558.1 Detailed Description

`template<typename _Tp> struct std::modulus<_Tp>`

One of the [math functors](#).

Definition at line 171 of file `stl_function.h`.

5.558.2 Member Typedef Documentation

5.558.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file stl_function.h.

5.558.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type
[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.558.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

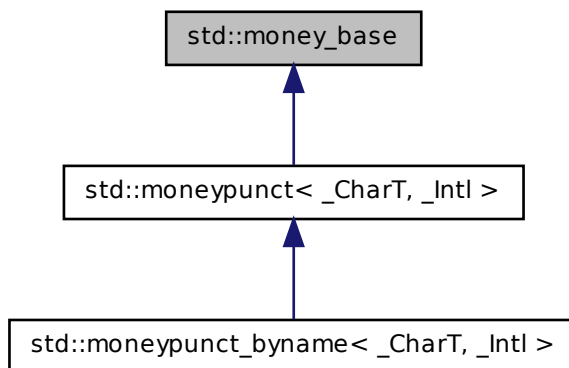
- [stl_function.h](#)

5.559 std::money_base Class Reference

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

Inheritance diagram for `std::money_base`:



Public Types

- enum { `_S_minus`, `_S_zero`, `_S_end` }
- enum **part** {
 `none`, `space`, `symbol`, `sign`,
 `value` }

Static Public Member Functions

- static `_GLIBCXX_CONST` pattern **`_S_construct_pattern`** (`char __precedes`, `char __space`, `char __posn`) throw ()

Static Public Attributes

- static const `char *` **`_S_atoms`**
- static const pattern **`_S_default_pattern`**

5.559.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

See also

[moneypunct::pos_format\(\)](#) and [moneypunct::neg_format\(\)](#) for details of how these fields are interpreted.

Definition at line 835 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

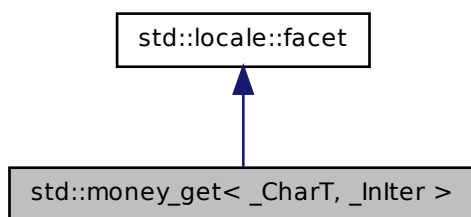
- [locale_facets_nonio.h](#)

5.560 `std::money_get< _CharT, _InIter >` Class Template Reference

Primary class template [money_get](#).

This facet encapsulates the code to parse and return a monetary amount from a string.

Inheritance diagram for `std::money_get< _CharT, _InIter >`:

**Public Types**

- typedef `_CharT` [char_type](#)
- typedef `_InIter` [iter_type](#)
- typedef [basic_string< _CharT >](#) [string_type](#)

Public Member Functions

- [money_get](#) (size_t __refs=0)
- [iter_type get](#) (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const
- [iter_type get](#) (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~money_get](#) ()
- template<bool _Intl>
[iter_type _M_extract](#) (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__digits) const
- virtual [iter_type do_get](#) (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const
- virtual [iter_type do_get](#) (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.560.1 Detailed Description

template<typename _CharT, typename _InIter> class std::money_get< _CharT, _InIter >

Primary class template [money_get](#).

This facet encapsulates the code to parse and return a monetary amount from a string. The [money_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [money_get](#) facet.

Definition at line 1365 of file locale_facets_nonio.h.

5.560.2 Member Typedef Documentation

**5.560.2.1 template<typename _CharT , typename _InIter > typedef _CharT
std::money_get< _CharT, _InIter >::char_type**

Public typedefs.

Definition at line 1371 of file locale_facets_nonio.h.

**5.560.2.2 template<typename _CharT , typename _InIter > typedef _InIter
std::money_get< _CharT, _InIter >::iter_type**

Public typedefs.

Definition at line 1372 of file locale_facets_nonio.h.

**5.560.2.3 template<typename _CharT , typename _InIter > typedef
basic_string<_CharT> std::money_get< _CharT, _InIter
>::string_type**

Public typedefs.

Definition at line 1373 of file locale_facets_nonio.h.

5.560.3 Constructor & Destructor Documentation

5.560.3.1 `template<typename _CharT, typename _InIter> std::money_get<_CharT, _InIter>::money_get (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 1387 of file locale_facets_nonio.h.

5.560.3.2 `template<typename _CharT, typename _InIter> virtual std::money_get<_CharT, _InIter>::~~money_get () [inline, protected, virtual]`

Destructor.

Definition at line 1455 of file locale_facets_nonio.h.

5.560.4 Member Function Documentation

5.560.4.1 `template<typename _CharT, typename _InIter> _InIter std::money_get<_CharT, _InIter>::do_get (iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, long double & __units) const [protected, virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for details.

Definition at line 363 of file locale_facets_nonio.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`.

Referenced by `std::money_get<_CharT, _InIter>::get()`.

```

5.560.4.2 template<typename _CharT, typename _InIter > _InIter
std::money_get< _CharT, _InIter >::do_get ( iter_type __s,
iter_type __end, bool __intl, ios_base & __io, ios_base::iostate &
__err, string_type & __digits ) const [protected, virtual]

```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for details.

Definition at line 376 of file locale_facets_nonio.tcc.

References `std::ios_base::M_getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::resize()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

```

5.560.4.3 template<typename _CharT, typename _InIter > iter_type
std::money_get< _CharT, _InIter >::get ( iter_type __s, iter_type
__end, bool __intl, ios_base & __io, ios_base::iostate & __err,
string_type & __digits ) const [inline]

```

Read and parse a monetary value.

This function reads characters from *s*, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *digits*. For example, the string \$10.01 in a US locale would store 1001 in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of [do_get\(\)](#).

Parameters

- s* Start of characters to parse.
- end* End of characters to parse.
- intl* Parameter to use `_facet<moneypunct<CharT,intl>>`.
- io* Source of facets and io state.
- err* Error field to set if parsing fails.
- digits* Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1448 of file locale_facets_nonio.h.

References `std::money_get<_CharT, _InIter>::do_get()`.

5.560.4.4 `template<typename _CharT, typename _InIter> iter_type
std::money_get<_CharT, _InIter>::get (iter_type __s, iter_type
__end, bool __intl, ios_base & __io, ios_base::iostate & __err, long
double & __units) const [inline]`

Read and parse a monetary value.

This function reads characters from *s*, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *units* as an integral value `moneypunct::frac_digits()` * the actual amount. For example, the string \$10.01 in a US locale would store 1001 in *units*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`. *units* is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

Parameters

- s* Start of characters to parse.
- end* End of characters to parse.
- intl* Parameter to use `_facet<moneypunct<CharT,intl>>`.
- io* Source of facets and io state.
- err* Error field to set if parsing fails.
- units* Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1417 of file locale_facets_nonio.h.

References `std::money_get<_CharT, _InIter>::do_get()`.

5.560.5 Member Data Documentation

5.560.5.1 `template<typename _CharT, typename _InIter> locale::id
std::money_get<_CharT, _InIter>::id [static]`

Numpunct facet id.

Definition at line 1377 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

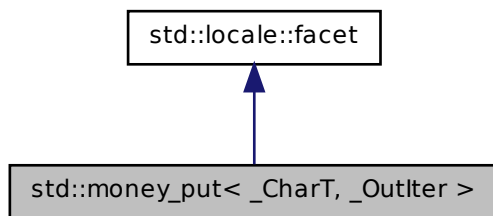
- [locale_facets_nonio.h](#)
- [locale_facets_nonio.tcc](#)

5.561 `std::money_put< _CharT, _OutIter >` Class Template Reference

Primary class template [money_put](#).

This facet encapsulates the code to format and output a monetary amount.

Inheritance diagram for `std::money_put< _CharT, _OutIter >`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `_OutIter` [iter_type](#)
- typedef [basic_string< _CharT >](#) [string_type](#)

Public Member Functions

- [money_put](#) (`size_t __refs=0`)
- [iter_type put](#) ([iter_type](#) __s, `bool __intl`, [ios_base](#) &__io, [char_type](#) __fill, `const string_type &__digits`) `const`
- [iter_type put](#) ([iter_type](#) __s, `bool __intl`, [ios_base](#) &__io, [char_type](#) __fill, `long double __units`) `const`

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~money_put](#) ()
- template<bool _Intl>
[iter_type](#) [_M_insert](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const [string_type](#) &__digits) const
- virtual [iter_type](#) [do_put](#) ([iter_type](#) __s, bool __intl, [ios_base](#) &__io, [char_type](#) __fill, const [string_type](#) &__digits) const
- virtual [iter_type](#) [do_put](#) ([iter_type](#) __s, bool __intl, [ios_base](#) &__io, [char_type](#) __fill, long double __units) const

Static Protected Member Functions

- static [__c_locale](#) [_S_clone_c_locale](#) ([__c_locale](#) &__cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) &__cloc, const char *__s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) &__cloc)
- static [__c_locale](#) [_S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static [__c_locale](#) [_S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.561.1 Detailed Description

template<typename _CharT, typename _OutIter> class std::money_put< _CharT, _OutIter >

Primary class template [money_put](#).

This facet encapsulates the code to format and output a monetary amount. The [money_put](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [money_put](#) facet.

Definition at line 1516 of file [locale_facets_nonio.h](#).

5.561.2 Member Typedef Documentation

5.561.2.1 `template<typename _CharT, typename _OutIter > typedef _CharT
std::money_put< _CharT, _OutIter >::char_type`

Public typedefs.

Definition at line 1521 of file locale_facets_nonio.h.

5.561.2.2 `template<typename _CharT, typename _OutIter > typedef _OutIter
std::money_put< _CharT, _OutIter >::iter_type`

Public typedefs.

Definition at line 1522 of file locale_facets_nonio.h.

5.561.2.3 `template<typename _CharT, typename _OutIter > typedef
basic_string<_CharT> std::money_put< _CharT, _OutIter
>::string_type`

Public typedefs.

Definition at line 1523 of file locale_facets_nonio.h.

5.561.3 Constructor & Destructor Documentation

5.561.3.1 `template<typename _CharT, typename _OutIter >
std::money_put< _CharT, _OutIter >::money_put (size_t __refs =
0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 1537 of file locale_facets_nonio.h.

5.561.3.2 `template<typename _CharT, typename _OutIter > virtual
std::money_put< _CharT, _OutIter >::~~money_put ()
[inline, protected, virtual]`

Destructor.

Definition at line 1587 of file locale_facets_nonio.h.

5.561.4 Member Function Documentation

5.561.4.1 `template<typename _CharT, typename _OutIter > _OutIter
std::money_put<_CharT, _OutIter >::do_put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, long double __units)
const [protected, virtual]`

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *s*. For example, the value 1001 in a US locale would write \$10.01 to *s*.

This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#).

Parameters

s The stream to write to.

intl Parameter to use_facet<moneypunct<CharT,intl> >.

io Source of facets and io state.

fill char_type to use for padding.

units Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 568 of file locale_facets_nonio.tcc.

References `std::ios_base::getloc()`, and `std::__ctype_abstract_base<_CharT >::widen()`.

Referenced by `std::money_put<_CharT, _OutIter >::put()`.

5.561.4.2 `template<typename _CharT, typename _OutIter > _OutIter
std::money_put<_CharT, _OutIter >::do_put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, const string_type &
__digits) const [protected, virtual]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the string `1001` in a US locale would write `$10.01` to *s*.

This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#).

Parameters

s The stream to write to.

intl Parameter to use `_facet<moneypunct<CharT,intl>>`.

io Source of facets and io state.

fill `char_type` to use for padding.

units Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 606 of file `locale_facets_nonio.tcc`.

5.561.4.3 `template<typename _CharT, typename _OutIter> iter_type
std::money_put<_CharT, _OutIter>::put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, const string_type &
__digits) const [inline]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the string `1001` in a US locale would write `$10.01` to *s*.

This function works by returning the result of [do_put\(\)](#).

Parameters

s The stream to write to.

intl Parameter to use `_facet<moneypunct<CharT,intl>>`.

io Source of facets and io state.

fill `char_type` to use for padding.

units Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 1580 of file locale_facets_nonio.h.

References `std::money_put<_CharT, _OutIter >::do_put()`.

5.561.4.4 `template<typename _CharT, typename _OutIter > iter_type
std::money_put<_CharT, _OutIter >::put (iter_type __s, bool
__intl, ios_base & __io, char_type __fill, long double __units)
const [inline]`

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to *s*. For example, the value 1001 in a US locale would write \$10.01 to *s*.

This function works by returning the result of `do_put()`.

Parameters

s The stream to write to.

intl Parameter to use `_facet<moneypunct<CharT,intl> >`.

io Source of facets and io state.

fill `char_type` to use for padding.

units Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 1557 of file locale_facets_nonio.h.

References `std::money_put<_CharT, _OutIter >::do_put()`.

5.561.5 Member Data Documentation

5.561.5.1 `template<typename _CharT, typename _OutIter > locale::id
std::money_put<_CharT, _OutIter >::id [static]`

Numpunct facet id.

Definition at line 1527 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

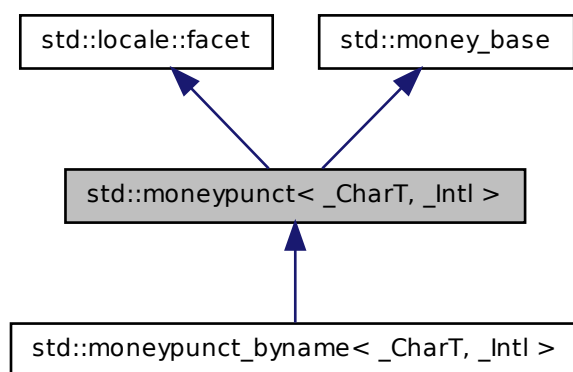
- [locale_facets_nonio.h](#)
- [locale_facets_nonio.tcc](#)

5.562 std::moneypunct< _CharT, _Intl > Class Template Reference

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Inheritance diagram for std::moneypunct< _CharT, _Intl >:



Public Types

- enum { **_S_minus**, **_S_zero**, **_S_end** }
- typedef __moneypunct_cache< _CharT, _Intl > **__cache_type**
- enum **part** {
none, **space**, **symbol**, **sign**,
value }
- typedef _CharT [char_type](#)
- typedef [basic_string](#)< _CharT > [string_type](#)

Public Member Functions

- [moneypunct](#) (size_t __refs=0)
 - [moneypunct](#) (__cache_type *__cache, size_t __refs=0)
 - [moneypunct](#) (__c_locale __cloc, const char *__s, size_t __refs=0)
 - [string_type curr_symbol](#) () const
 - [char_type decimal_point](#) () const
 - int [frac_digits](#) () const
 - [string grouping](#) () const
 - [string_type negative_sign](#) () const
 - [string_type positive_sign](#) () const
 - [char_type thousands_sep](#) () const
-
- pattern [pos_format](#) () const
 - pattern [neg_format](#) () const

Static Public Member Functions

- static _GLIBCXX_CONST pattern [_S_construct_pattern](#) (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * [_S_atoms](#)
- static const pattern [_S_default_pattern](#)
- static [locale::id](#) [id](#)
- static const bool [intl](#)

Protected Member Functions

- virtual [~moneypunct](#) ()
- template<>
void [_M_initialize_moneypunct](#) (__c_locale, const char *)
- void [_M_initialize_moneypunct](#) (__c_locale __cloc=0, const char *__name=0)
- template<>
void [_M_initialize_moneypunct](#) (__c_locale, const char *)
- template<>
void [_M_initialize_moneypunct](#) (__c_locale, const char *)
- template<>
void [_M_initialize_moneypunct](#) (__c_locale, const char *)
- virtual [string_type do_curr_symbol](#) () const

- virtual [char_type do_decimal_point](#) () const
- virtual int [do_frac_digits](#) () const
- virtual [string do_grouping](#) () const
- virtual pattern [do_neg_format](#) () const
- virtual [string_type do_negative_sign](#) () const
- virtual pattern [do_pos_format](#) () const
- virtual [string_type do_positive_sign](#) () const
- virtual [char_type do_thousands_sep](#) () const

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Friends

- class `locale::_Impl`

5.562.1 Detailed Description

`template<typename _CharT, bool _Intl> class std::moneypunct< _CharT, _Intl >`

Primary class template `moneypunct`.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 929 of file `locale_facets_nonio.h`.

5.562.2 Member Typedef Documentation

5.562.2.1 `template<typename _CharT, bool _Intl> typedef _CharT std::moneypunct< _CharT, _Intl >::char_type`

Public typedefs.

Reimplemented in `std::moneypunct_byname< _CharT, _Intl >`.

Definition at line 935 of file `locale_facets_nonio.h`.

5.562.2.2 `template<typename _CharT, bool _Intl> typedef
basic_string<_CharT> std::moneypunct<_CharT, _Intl
>::string_type`

Public typedefs.

Reimplemented in [std::moneypunct_byname<_CharT, _Intl>](#).

Definition at line 936 of file locale_facets_nonio.h.

5.562.3 Constructor & Destructor Documentation

5.562.3.1 `template<typename _CharT, bool _Intl> std::moneypunct<_CharT,
_Intl>::moneypunct (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 958 of file locale_facets_nonio.h.

5.562.3.2 `template<typename _CharT, bool _Intl> std::moneypunct<_CharT,
_Intl>::moneypunct (__cache_type * __cache, size_t __refs = 0)
[inline, explicit]`

Constructor performs initialization.

This is an internal constructor.

Parameters

cache Cache for optimization.

refs Passed to the base facet class.

Definition at line 971 of file locale_facets_nonio.h.

5.562.3.3 `template<typename _CharT, bool _Intl> std::moneypunct<_CharT,
_Intl>::moneypunct (__c_locale __cloc, const char * __s, size_t
__refs = 0) [inline, explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

- cloc* The C locale.
- s* The name of a locale.
- refs* Passed to the base facet class.

Definition at line 986 of file locale_facets_nonio.h.

5.562.3.4 `template<typename _CharT, bool _Intl> virtual std::moneypunct<_CharT, _Intl >::~~moneypunct () [protected, virtual]`

Destructor.

5.562.4 Member Function Documentation

5.562.4.1 `template<typename _CharT, bool _Intl> string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const [inline]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning returning `moneypunct<char_type>::do_curr_symbol()`.

Returns

string_type representing a currency symbol.

Definition at line 1056 of file locale_facets_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_curr_symbol()`.

5.562.4.2 `template<typename _CharT, bool _Intl> char_type std::moneypunct< _CharT, _Intl >::decimal_point () const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `moneypunct<char_type>::do_decimal_point()`.

Returns

char_type representing a decimal point.

Definition at line 1000 of file locale_facets_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_decimal_point()`.

5.562.4.3 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct< _CharT, _Intl >::do_curr_symbol () const
[inline, protected, virtual]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

[curr_symbol\(\)](#) for details.

Returns

string_type representing a currency symbol.

Definition at line 1202 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::curr_symbol()`.

5.562.4.4 `template<typename _CharT, bool _Intl> virtual char_type
std::moneypunct< _CharT, _Intl >::do_decimal_point () const
[inline, protected, virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1164 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::decimal_point()`.

5.562.4.5 `template<typename _CharT, bool _Intl> virtual int
std::moneypunct< _CharT, _Intl >::do_frac_digits () const
[inline, protected, virtual]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

[frac_digits\(\)](#) for details.

Returns

Number of digits in amount fraction.

Definition at line 1242 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::frac_digits().

5.562.4.6 `template<typename _CharT, bool _Intl> virtual string
std::moneypunct< _CharT, _Intl >::do_grouping () const
[inline, protected, virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

[grouping\(\)](#) for details.

Returns

String representing grouping specification.

Definition at line 1189 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::grouping().

5.562.4.7 `template<typename _CharT, bool _Intl> virtual pattern
std::moneypunct< _CharT, _Intl >::do_neg_format () const
[inline, protected, virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

[neg_format\(\)](#) for details.

Returns

Pattern for money values.

Definition at line 1270 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::neg_format().

5.562.4.8 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct< _CharT, _Intl >::do_negative_sign () const
[inline, protected, virtual]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

[negative_sign\(\)](#) for details.

Returns

string_type representing a negative sign.

Definition at line 1228 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::negative_sign()`.

5.562.4.9 `template<typename _CharT, bool _Intl> virtual pattern
std::moneypunct< _CharT, _Intl >::do_pos_format () const
[inline, protected, virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

[pos_format\(\)](#) for details.

Returns

Pattern for money values.

Definition at line 1256 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::pos_format()`.

5.562.4.10 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct< _CharT, _Intl >::do_positive_sign () const
[inline, protected, virtual]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

[positive_sign\(\)](#) for details.

Returns

string_type representing a positive sign.

Definition at line 1215 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::positive_sign().

5.562.4.11 `template<typename _CharT, bool _Intl> virtual char_type
std::moneypunct< _CharT, _Intl >::do_thousands_sep () const
[inline, protected, virtual]`

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1176 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::thousands_sep().

5.562.4.12 `template<typename _CharT, bool _Intl> int std::moneypunct<
_CharT, _Intl >::frac_digits () const [inline]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning [moneypunct<char_type>::do_frac_digits\(\)](#).

The fractional part of a money amount is optional. But if it is present, there must be [frac_digits\(\)](#) digits.

Returns

Number of digits in amount fraction.

Definition at line 1106 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_frac_digits().

5.562.4.13 `template<typename _CharT, bool _Intl> string std::moneypunct<_CharT, _Intl>::grouping () const [inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `32` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

Definition at line 1043 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_grouping()`.

5.562.4.14 `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format () const [inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none},

`curr_symbol()` == '\$' `positive_sign()` == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

Returns

Pattern for money values.

Definition at line 1146 of file locale_facets_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_neg_format()`.

5.562.4.15 `template<typename _CharT, bool _Intl> string_type
std::moneypunct< _CharT, _Intl >::negative_sign () const
[inline]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

Definition at line 1090 of file locale_facets_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_negative_sign()`.

5.562.4.16 `template<typename _CharT, bool _Intl> pattern std::moneypunct<
_CharT, _Intl >::pos_format () const [inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be

present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

Returns

Pattern for money values.

Definition at line 1142 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_pos_format()`.

5.562.4.17 `template<typename _CharT, bool _Intl> string_type
std::moneypunct<_CharT, _Intl>::positive_sign () const
[inline]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

Definition at line 1073 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_positive_sign()`.

5.562.4.18 `template<typename _CharT, bool _Intl> char_type
std::moneypunct<_CharT, _Intl>::thousands_sep () const
[inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

5.563 std::moneypunct_byname< _CharT, _Intl > Class Template Reference

Definition at line 1013 of file locale_facets_nonio.h.

References `std::moneypunct< _CharT, _Intl >::do_thousands_sep()`.

5.562.5 Member Data Documentation

5.562.5.1 `template<typename _CharT, bool _Intl> locale::id
std::moneypunct< _CharT, _Intl >::id [static]`

Numpunct facet id.

Definition at line 948 of file locale_facets_nonio.h.

5.562.5.2 `template<typename _CharT, bool _Intl> const bool
std::moneypunct< _CharT, _Intl >::intl [static]`

This value is provided by the standard, but no reason for its /// existence.

Reimplemented in [std::moneypunct_byname< _CharT, _Intl >](#).

Definition at line 946 of file locale_facets_nonio.h.

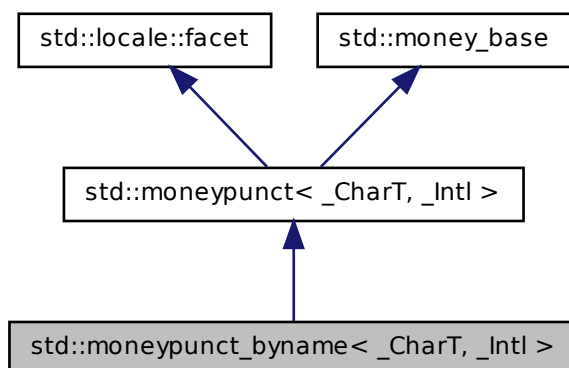
The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

5.563 std::moneypunct_byname< _CharT, _Intl > Class Template Reference

class [moneypunct_byname](#) [22.2.6.4].

Inheritance diagram for `std::moneypunct_byname< _CharT, _Intl >`:



Public Types

- enum { **_S_minus**, **_S_zero**, **_S_end** }
- typedef `__moneypunct_cache< _CharT, _Intl >` **__cache_type**
- typedef `_CharT` **char_type**
- enum **part** {
 none, **space**, **symbol**, **sign**,
 value }
- typedef `basic_string< _CharT >` **string_type**

Public Member Functions

- **moneypunct_byname** (const char *__s, size_t __refs=0)
- **string_type curr_symbol** () const
- **char_type decimal_point** () const
- int **frac_digits** () const
- **string grouping** () const
- **string_type negative_sign** () const
- **string_type positive_sign** () const
- **char_type thousands_sep** () const

5.563 std::moneypunct_byname<_CharT, _Intl> Class Template Reference 2737

- pattern [pos_format](#) () const
- pattern [neg_format](#) () const

Static Public Member Functions

- static `_GLIBCXX_CONST` pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**
- static [locale::id](#) **id**
- static const bool [intl](#)

Protected Member Functions

- template<>
void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale __cloc=0, const char *__name=0)
- template<>
void **_M_initialize_moneypunct** (__c_locale, const char *)
- template<>
void **_M_initialize_moneypunct** (__c_locale, const char *)
- template<>
void **_M_initialize_moneypunct** (__c_locale, const char *)
- virtual [string_type](#) **do_curr_symbol** () const
- virtual [char_type](#) **do_decimal_point** () const
- virtual int **do_frac_digits** () const
- virtual [string](#) **do_grouping** () const
- virtual pattern **do_neg_format** () const
- virtual [string_type](#) **do_negative_sign** () const
- virtual pattern **do_pos_format** () const
- virtual [string_type](#) **do_positive_sign** () const
- virtual [char_type](#) **do_thousands_sep** () const

Static Protected Member Functions

- static `__c_locale` **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)

- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Friends

- class **locale::_Impl**

5.563.1 Detailed Description

template<typename _CharT, bool _Intl> class std::moneypunct_byname< _CharT, _Intl >

class [moneypunct_byname](#) [22.2.6.4].

Definition at line 1319 of file locale_facets_nonio.h.

5.563.2 Member Typedef Documentation

5.563.2.1 template<typename _CharT , bool _Intl> typedef _CharT std::moneypunct_byname< _CharT, _Intl >::char_type

Public typedefs.

Reimplemented from [std::moneypunct< _CharT, _Intl >](#).

Definition at line 1322 of file locale_facets_nonio.h.

5.563.2.2 template<typename _CharT , bool _Intl> typedef basic_string<_CharT> std::moneypunct_byname< _CharT, _Intl >::string_type

Public typedefs.

Reimplemented from [std::moneypunct< _CharT, _Intl >](#).

Definition at line 1323 of file locale_facets_nonio.h.

5.563.3 Member Function Documentation

5.563.3.1 `template<typename _CharT, bool _Intl> string_type
std::moneypunct<_CharT, _Intl>::curr_symbol () const
[inline, inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning [returning moneypunct<char_type>::do_curr_symbol\(\)](#).

Returns

string_type representing a currency symbol.

Definition at line 1056 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_curr_symbol()`.

5.563.3.2 `template<typename _CharT, bool _Intl> char_type
std::moneypunct<_CharT, _Intl>::decimal_point () const
[inline, inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning [returning moneypunct<char_type>::do_decimal_point\(\)](#).

Returns

char_type representing a decimal point.

Definition at line 1000 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_decimal_point()`.

5.563.3.3 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct<_CharT, _Intl>::do_curr_symbol () const
[inline, protected, virtual, inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

[curr_symbol\(\)](#) for details.

Returns

string_type representing a currency symbol.

Definition at line 1202 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::curr_symbol()`.

5.563.3.4 `template<typename _CharT, bool _Intl> virtual char_type
std::moneypunct<_CharT, _Intl >::do_decimal_point () const
[inline, protected, virtual, inherited]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1164 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::decimal_point()`.

5.563.3.5 `template<typename _CharT, bool _Intl> virtual int
std::moneypunct<_CharT, _Intl >::do_frac_digits () const
[inline, protected, virtual, inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

[frac_digits\(\)](#) for details.

Returns

Number of digits in amount fraction.

Definition at line 1242 of file locale_facets_nonio.h.

Referenced by `std::moneypunct<_CharT, _Intl >::frac_digits()`.

5.563 std::moneypunct_byname<_CharT, _Intl> Class Template Reference 2741

5.563.3.6 `template<typename _CharT, bool _Intl> virtual string
std::moneypunct<_CharT, _Intl>::do_grouping () const
[inline, protected, virtual, inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

[grouping\(\)](#) for details.

Returns

String representing grouping specification.

Definition at line 1189 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl>::grouping().

5.563.3.7 `template<typename _CharT, bool _Intl> virtual pattern
std::moneypunct<_CharT, _Intl>::do_neg_format () const
[inline, protected, virtual, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

[neg_format\(\)](#) for details.

Returns

Pattern for money values.

Definition at line 1270 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl>::neg_format().

5.563.3.8 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct<_CharT, _Intl>::do_negative_sign () const
[inline, protected, virtual, inherited]`

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

[negative_sign\(\)](#) for details.

Returns

string_type representing a negative sign.

Definition at line 1228 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl >::negative_sign()`.

5.563.3.9 `template<typename _CharT, bool _Intl> virtual pattern
std::moneypunct<_CharT, _Intl >::do_pos_format () const
[inline, protected, virtual, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

[pos_format\(\)](#) for details.

Returns

Pattern for money values.

Definition at line 1256 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct<_CharT, _Intl >::pos_format()`.

5.563.3.10 `template<typename _CharT, bool _Intl> virtual string_type
std::moneypunct<_CharT, _Intl >::do_positive_sign () const
[inline, protected, virtual, inherited]`

Return positive sign string.

This function returns a *string_type* to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

[positive_sign\(\)](#) for details.

Returns

string_type representing a positive sign.

5.563 std::moneypunct_byname<_CharT, _Intl> Class Template Reference2743

Definition at line 1215 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl>::positive_sign().

5.563.3.11 `template<typename _CharT, bool _Intl> virtual char_type
std::moneypunct<_CharT, _Intl>::do_thousands_sep() const
[inline, protected, virtual, inherited]`

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1176 of file locale_facets_nonio.h.

Referenced by std::moneypunct<_CharT, _Intl>::thousands_sep().

5.563.3.12 `template<typename _CharT, bool _Intl> int std::moneypunct<
_CharT, _Intl>::frac_digits() const [inline, inherited]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning [moneypunct<char_type>::do_frac_digits\(\)](#).

The fractional part of a money amount is optional. But if it is present, there must be [frac_digits\(\)](#) digits.

Returns

Number of digits in amount fraction.

Definition at line 1106 of file locale_facets_nonio.h.

References std::moneypunct<_CharT, _Intl>::do_frac_digits().

5.563.3.13 `template<typename _CharT, bool _Intl> string std::moneypunct<
_CharT, _Intl>::grouping() const [inline, inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the [grouping\(\)](#) returns `32` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling [moneypunct<char_type>::do_grouping\(\)](#).

Returns

string representing grouping specification.

Definition at line 1043 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::do_grouping()`.

5.563.3.14 `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format() const [inline, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning [moneypunct<char_type>::do_pos_format\(\)](#) or [moneypunct<char_type>::do_neg_format\(\)](#).

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of [curr_symbol\(\)](#) may be present. The sign field indicates that the value of [positive_sign\(\)](#) or [negative_sign\(\)](#) must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and [pos_format\(\)](#) pattern {symbol,sign,value,none}, [curr_symbol\(\)](#) == '\$' [positive_sign\(\)](#) == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

Returns

Pattern for money values.

Definition at line 1146 of file `locale_facets_nonio.h`.

5.563 std::moneypunct_byname<_CharT, _Intl> Class Template Reference 2745

References std::moneypunct<_CharT, _Intl>::do_neg_format().

5.563.3.15 `template<typename _CharT, bool _Intl> string_type
std::moneypunct<_CharT, _Intl>::negative_sign () const
[inline, inherited]`

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. It does so by returning returning [moneypunct<char_type>::do_negative_sign\(\)](#).

If the return value contains more than one character, the first character appears in the position indicated by [neg_format\(\)](#) and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

Definition at line 1090 of file locale_facets_nonio.h.

References std::moneypunct<_CharT, _Intl>::do_negative_sign().

5.563.3.16 `template<typename _CharT, bool _Intl> pattern std::moneypunct<
_CharT, _Intl>::pos_format () const [inline, inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning [moneypunct<char_type>::do_pos_format\(\)](#) or [moneypunct<char_type>::do_neg_format\(\)](#).

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of [curr_symbol\(\)](#) may be present. The sign field indicates that the value of [positive_sign\(\)](#) or [negative_sign\(\)](#) must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and [pos_format\(\)](#) pattern {symbol,sign,value,none}, [curr_symbol\(\)](#) == '\$' [positive_sign\(\)](#) == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

Returns

Pattern for money values.

Definition at line 1142 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_pos_format()`.

5.563.3.17 `template<typename _CharT, bool _Intl> string_type
std::moneypunct<_CharT, _Intl >::positive_sign () const
[inline, inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

Definition at line 1073 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_positive_sign()`.

5.563.3.18 `template<typename _CharT, bool _Intl> char_type
std::moneypunct<_CharT, _Intl >::thousands_sep () const
[inline, inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1013 of file locale_facets_nonio.h.

References `std::moneypunct<_CharT, _Intl >::do_thousands_sep()`.

5.563.4 Member Data Documentation

5.563.4.1 `template<typename _CharT, bool _Intl> locale::id`
`std::moneypunct< _CharT, _Intl >::id` [`static`, `inherited`]

Numpunct facet id.

Definition at line 948 of file `locale_facets_nonio.h`.

5.563.4.2 `template<typename _CharT, bool _Intl> const bool`
`std::moneypunct_byname< _CharT, _Intl >::intl` [`static`]

This value is provided by the standard, but no reason for its /// existence.

Reimplemented from `std::moneypunct< _CharT, _Intl >`.

Definition at line 1325 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

5.564 `std::move_iterator< _Iterator >` Class Template Reference

Public Types

- `typedef __traits_type::difference_type` **difference_type**
- `typedef __traits_type::iterator_category` **iterator_category**
- `typedef _Iterator` **iterator_type**
- `typedef _Iterator` **pointer**
- `typedef value_type &&` **reference**
- `typedef __traits_type::value_type` **value_type**

Public Member Functions

- **move_iterator** (iterator_type __i)
- `template<typename _Iter >`
move_iterator (const [move_iterator](#)< _Iter > &__i)
- iterator_type **base** () const
- reference **operator*** () const
- [move_iterator](#) **operator+** (difference_type __n) const
- [move_iterator](#) **operator++** (int)

- [move_iterator](#) & **operator++** ()
- [move_iterator](#) & **operator+=** (difference_type __n)
- [move_iterator](#) **operator-** (difference_type __n) const
- [move_iterator](#) & **operator--** ()
- [move_iterator](#) **operator--** (int)
- [move_iterator](#) & **operator-=** (difference_type __n)
- pointer **operator->** () const
- reference **operator[]** (difference_type __n) const

Protected Types

- typedef iterator_traits< _Iterator > **__traits_type**

Protected Attributes

- _Iterator **_M_current**

5.564.1 Detailed Description

template<typename _Iterator> class std::move_iterator< _Iterator >

Class template [move_iterator](#) is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 916 of file stl_iterator.h.

The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

5.565 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Public Types

- `typedef _Alloc allocator_type`
- `typedef _Rep_type::const_iterator const_iterator`
- `typedef _Pair_alloc_type::const_pointer const_pointer`
- `typedef _Pair_alloc_type::const_reference const_reference`
- `typedef _Rep_type::const_reverse_iterator const_reverse_iterator`
- `typedef _Rep_type::difference_type difference_type`
- `typedef _Rep_type::iterator iterator`
- `typedef _Compare key_compare`
- `typedef _Key key_type`
- `typedef _Tp mapped_type`
- `typedef _Pair_alloc_type::pointer pointer`
- `typedef _Pair_alloc_type::reference reference`
- `typedef _Rep_type::reverse_iterator reverse_iterator`
- `typedef _Rep_type::size_type size_type`
- `typedef std::pair< const _Key, _Tp > value_type`

Public Member Functions

- `multimap()`
- `multimap(const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `multimap(multimap &&__x)`
- `multimap(initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())`
- `multimap(const multimap &__x)`
- `template<typename _InputIterator >
multimap(_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator >
multimap(_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `iterator begin()`
- `const_iterator begin() const`
- `const_iterator cbegin() const`
- `const_iterator cend() const`
- `void clear()`
- `size_type count(const key_type &__x) const`
- `const_reverse_iterator crbegin() const`
- `const_reverse_iterator crend() const`
- `bool empty() const`
- `iterator end()`
- `const_iterator end() const`

- `std::pair< const_iterator, const_iterator > equal_range` (const key_type &__x) const
- `std::pair< iterator, iterator > equal_range` (const key_type &__x)
- iterator `erase` (iterator __first, iterator __last)
- size_type `erase` (const key_type &__x)
- iterator `erase` (iterator __position)
- iterator `find` (const key_type &__x)
- const_iterator `find` (const key_type &__x) const
- allocator_type `get_allocator` () const
- iterator `insert` (const value_type &__x)
- iterator `insert` (iterator __position, const value_type &__x)
- template<typename _InputIterator >
void `insert` (_InputIterator __first, _InputIterator __last)
- void `insert` (initializer_list< value_type > __l)
- key_compare `key_comp` () const
- const_iterator `lower_bound` (const key_type &__x) const
- iterator `lower_bound` (const key_type &__x)
- size_type `max_size` () const
- `multimap` & `operator=` (const `multimap` &__x)
- `multimap` & `operator=` (`multimap` &&__x)
- `multimap` & `operator=` (initializer_list< value_type > __l)
- `reverse_iterator` `rbegin` ()
- `const_reverse_iterator` `rbegin` () const
- `reverse_iterator` `rend` ()
- `const_reverse_iterator` `rend` () const
- size_type `size` () const
- void `swap` (`multimap` &__x)
- const_iterator `upper_bound` (const key_type &__x) const
- iterator `upper_bound` (const key_type &__x)
- value_compare `value_comp` () const

Friends

- template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool **operator**< (const `multimap`< _K1, _T1, _C1, _A1 > &, const `multimap`< _K1, _T1, _C1, _A1 > &)
- template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool **operator**== (const `multimap`< _K1, _T1, _C1, _A1 > &, const `multimap`< _K1, _T1, _C1, _A1 > &)

5.565.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> class
std::multimap< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key,T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 86 of file `stl_multimap.h`.

5.565.2 Constructor & Destructor Documentation

5.565.2.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap() [inline]`

Default constructor creates no elements.

Definition at line 148 of file `stl_multimap.h`.

5.565.2.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap(const _Compare & __comp, const allocator_type & __a = allocator_type()) [inline, explicit]`

Creates a multimap with no elements.

Parameters

comp A comparison object.

a An allocator object.

Definition at line 157 of file `stl_multimap.h`.

```

5.565.2.3  template<typename _Key, typename _Tp, typename _Compare =
            std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
            _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc
            >::multimap ( const multimap<_Key, _Tp, _Compare, _Alloc > &
            __x ) [inline]

```

Multimap copy constructor.

Parameters

x A multimap of identical element and allocator types.

The newly-created multimap uses a copy of the allocation object used by *x*.

Definition at line 168 of file `stl_multimap.h`.

```

5.565.2.4  template<typename _Key, typename _Tp, typename _Compare =
            std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
            _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc
            >::multimap ( multimap<_Key, _Tp, _Compare, _Alloc > && __x
            ) [inline]

```

Multimap move constructor.

Parameters

x A multimap of identical element and allocator types.

The newly-created multimap contains the exact contents of *x*. The contents of *x* are a valid, but unspecified multimap.

Definition at line 179 of file `stl_multimap.h`.

```

5.565.2.5  template<typename _Key, typename _Tp, typename _Compare =
            std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
            _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc
            >::multimap ( initializer_list< value_type > __l, const _Compare
            & __comp = _Compare(), const allocator_type & __a =
            allocator_type() ) [inline]

```

Builds a multimap from an [initializer_list](#).

Parameters

l An [initializer_list](#).

comp A comparison functor.

5.565 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

a An allocator object.

Create a multimap consisting of copies of the elements from the [initializer_list](#). This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 192 of file `stl_multimap.h`.

```
5.565.2.6 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> template<typename _InputIterator >
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
_InputIterator __first, _InputIterator __last ) [inline]
```

Builds a multimap from a range.

Parameters

first An input iterator.

last An input iterator.

Create a multimap consisting of copies of the elements from `[first,last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(first,last)`).

Definition at line 209 of file `stl_multimap.h`.

```
5.565.2.7 template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> template<typename _InputIterator >
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
_InputIterator __first, _InputIterator __last, const _Compare &
__comp, const allocator_type & __a = allocator_type() )
[inline]
```

Builds a multimap from a range.

Parameters

first An input iterator.

last An input iterator.

comp A comparison functor.

a An allocator object.

Create a multimap consisting of copies of the elements from `[first,last)`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `distance(first,last)`).

Definition at line 225 of file `stl_multimap.h`.

5.565.3 Member Function Documentation

5.565.3.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::begin () [inline]`

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 304 of file `stl_multimap.h`.

5.565.3.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 313 of file `stl_multimap.h`.

5.565.3.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 377 of file `stl_multimap.h`.

5.565.3.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 386 of file `stl_multimap.h`.

5.565 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

5.565.3.5 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc
>::clear () [inline]`

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 601 of file stl_multimap.h.

5.565.3.6 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare,
_Alloc >::count (const key_type & __x) const [inline]`

Finds the number of elements with given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

Definition at line 658 of file stl_multimap.h.

5.565.3.7 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::crbegin () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 395 of file stl_multimap.h.

5.565.3.8 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::crend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 404 of file stl_multimap.h.

5.565.3.9 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> bool std::multimap<_Key, _Tp, _Compare, _Alloc
>::empty () const [inline]`

Returns true if the multimap is empty.

Definition at line 411 of file stl_multimap.h.

5.565.3.10 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::multimap<_Key, _Tp,
_Compare, _Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 331 of file stl_multimap.h.

5.565.3.11 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare,
_Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 322 of file stl_multimap.h.

5.565.3.12 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> std::pair<iterator, iterator> std::multimap<_Key,
_Tp, _Compare, _Alloc >::equal_range (const key_type & __x)
[inline]`

Finds a subsequence matching given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

5.565 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 725 of file stl_multimap.h.

5.565.3.13 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> std::pair<const_iterator, const_iterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
const key_type & __x) const [inline]`

Finds a subsequence matching given key.

Parameters

x Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 742 of file stl_multimap.h.

5.565.3.14 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::erase (iterator __position) [inline]`

Erases an element from a multimap.

Parameters

position An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 509 of file `stl_multimap.h`.

```
5.565.3.15  template<typename _Key, typename _Tp, typename _Compare =
               std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
               _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare,
               _Alloc>::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

x Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 539 of file `stl_multimap.h`.

```
5.565.3.16  template<typename _Key, typename _Tp, typename _Compare =
               std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
               _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare,
               _Alloc>::erase ( iterator __first, iterator __last ) [inline]
```

Erases a `[first,last)` range of elements from a multimap.

Parameters

first Iterator pointing to the start of the range to be erased.

last Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.565 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

Definition at line 558 of file stl_multimap.h.

```
5.565.3.17  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >> const_iterator std::multimap< _Key, _Tp,
              _Compare, _Alloc >::find ( const key_type & __x ) const
              [inline]
```

Tries to locate an element in a multimap.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or [end\(\)](#) if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ([end\(\)](#)) iterator.

Definition at line 649 of file stl_multimap.h.

```
5.565.3.18  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare,
              _Alloc >::find ( const key_type & __x ) [inline]
```

Tries to locate an element in a multimap.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or [end\(\)](#) if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ([end\(\)](#)) iterator.

Definition at line 634 of file stl_multimap.h.

5.565.3.19 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> allocator_type std::multimap< _Key, _Tp,
_Compare, _Alloc >::get_allocator () const [inline]`

Get a copy of the memory allocation object.

Definition at line 294 of file `stl_multimap.h`.

5.565.3.20 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::insert (const value_type & __x) [inline]`

Inserts a [std::pair](#) into the multimap.

Parameters

x Pair to be inserted (see [std::make_pair](#) for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a [std::map](#) the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 438 of file `stl_multimap.h`.

5.565.3.21 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,
_Alloc >::insert (iterator __position, const value_type & __x)
[inline]`

Inserts a [std::pair](#) into the multimap.

Parameters

position An iterator that serves as a hint as to where the pair should be inserted.

x Pair to be inserted (see [std::make_pair](#) for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

5.565 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

This function inserts a (key, value) pair into the multimap. Contrary to a [std::map](#) the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 462 of file stl_multimap.h.

```
5.565.3.22  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> template<typename _InputIterator > void  
             std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (  
             _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

Parameters

first Iterator pointing to the start of the range to be inserted.

last Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 476 of file stl_multimap.h.

```
5.565.3.23  template<typename _Key, typename _Tp, typename _Compare =  
             std::less<_Key>, typename _Alloc = std::allocator<std::pair<const  
             _Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc  
             >::insert ( initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of std::pairs into the multimap.

Parameters

list A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 488 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc >::insert().

Referenced by std::multimap< _Key, _Tp, _Compare, _Alloc >::insert().

5.565.3.24 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> key_compare std::multimap<_Key, _Tp,
_Compare, _Alloc>::key_comp () const [inline]`

Returns the key comparison object out of which the multimap was constructed.

Definition at line 610 of file `stl_multimap.h`.

5.565.3.25 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_iterator std::multimap<_Key, _Tp,
_Compare, _Alloc>::lower_bound (const key_type & __x) const
[inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or `end()`.

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to `end()`.

Definition at line 688 of file `stl_multimap.h`.

5.565.3.26 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare,
_Alloc>::lower_bound (const key_type & __x) [inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or `end()`.

5.565 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 673 of file stl_multimap.h.

```
5.565.3.27  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare,
              _Alloc >::max_size ( ) const  [inline]
```

Returns the maximum size of the multimap.

Definition at line 421 of file stl_multimap.h.

```
5.565.3.28  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> multimap& std::multimap< _Key, _Tp, _Compare,
              _Alloc >::operator= ( multimap< _Key, _Tp, _Compare, _Alloc >
              && __x ) [inline]
```

Multimap move assignment operator.

Parameters

x A multimap of identical element and allocator types.

The contents of *x* are moved into this multimap (without copying). *x* is a valid, but unspecified multimap.

Definition at line 263 of file stl_multimap.h.

```
5.565.3.29  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> multimap& std::multimap< _Key, _Tp, _Compare,
              _Alloc >::operator= ( const multimap< _Key, _Tp, _Compare,
              _Alloc > & __x ) [inline]
```

Multimap assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

x A multimap of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 248 of file `stl_multimap.h`.

```
5.565.3.30  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >>> multimap& std::multimap<_Key, _Tp, _Compare,
              _Alloc >::operator= ( initializer_list< value_type > __l )
              [inline]
```

Multimap list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a multimap with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 284 of file `stl_multimap.h`.

```
5.565.3.31  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >>> reverse_iterator std::multimap<_Key, _Tp,
              _Compare, _Alloc >::rbegin ( ) [inline]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 340 of file `stl_multimap.h`.

```
5.565.3.32  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp> >>> const_reverse_iterator std::multimap<_Key, _Tp,
              _Compare, _Alloc >::rbegin ( ) const [inline]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 349 of file `stl_multimap.h`.

5.565 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

5.565.3.33 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> reverse_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 358 of file stl_multimap.h.

5.565.3.34 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp,
_Compare, _Alloc >::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 367 of file stl_multimap.h.

5.565.3.35 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> size_type std::multimap< _Key, _Tp, _Compare,
_Alloc >::size () const [inline]`

Returns the size of the multimap.

Definition at line 416 of file stl_multimap.h.

5.565.3.36 `template<typename _Key, typename _Tp, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>> void std::multimap< _Key, _Tp, _Compare, _Alloc
>::swap (multimap< _Key, _Tp, _Compare, _Alloc > & __x)
[inline]`

Swaps data with another multimap.

Parameters

x A multimap of the same element and allocator types.

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 591 of file stl_multimap.h.

Referenced by std::swap().

```
5.565.3.37  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp,
              _Compare, _Alloc >::upper_bound ( const key_type & __x ) const
              [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or [end\(\)](#).

Definition at line 708 of file stl_multimap.h.

```
5.565.3.38  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare,
              _Alloc >::upper_bound ( const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 698 of file stl_multimap.h.

```
5.565.3.39  template<typename _Key, typename _Tp, typename _Compare =
              std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
              _Key, _Tp>>> value_compare std::multimap< _Key, _Tp,
              _Compare, _Alloc >::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

Definition at line 618 of file stl_multimap.h.

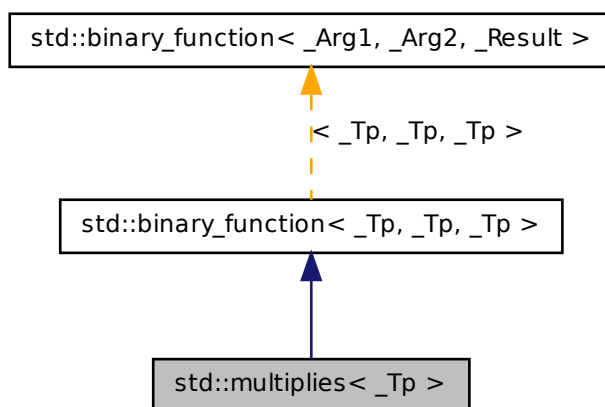
The documentation for this class was generated from the following file:

- [stl_multimap.h](#)

5.566 std::multiplies< _Tp > Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::multiplies< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

5.566.1 Detailed Description

`template<typename _Tp> struct std::multiplies< _Tp >`

One of the [math functors](#).

Definition at line 153 of file `stl_function.h`.

5.566.2 Member Typedef Documentation

5.566.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.566.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.566.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.567 `std::multiset< _Key, _Compare, _Alloc >` Class Template Reference

A standard container made up of elements, which can be retrieved in logarithmic time.

Public Types

- `typedef _Alloc allocator_type`

- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Key_alloc_type::const_pointer **const_pointer**
- typedef _Key_alloc_type::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::const_iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Key_alloc_type::pointer **pointer**
- typedef _Key_alloc_type::reference **reference**
- typedef _Rep_type::const_reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **multiset** ()
- **multiset** (const _Compare &__comp, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- **multiset** (const **multiset** &__x)
- template<typename _InputIterator >
multiset (_InputIterator __first, _InputIterator __last)
- **multiset** (**multiset** &&__x)
- **multiset** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- iterator **begin** () const
- iterator **cbegin** () const
- iterator **rend** () const
- void **clear** ()
- size_type **count** (const key_type &__x) const
- reverse_iterator **crbegin** () const
- reverse_iterator **crend** () const
- bool **empty** () const
- iterator **end** () const
- iterator **erase** (iterator __first, iterator __last)
- size_type **erase** (const key_type &__x)
- iterator **erase** (iterator __position)
- allocator_type **get_allocator** () const

- iterator [insert](#) (const value_type &__x)
 - iterator [insert](#) (iterator __position, const value_type &__x)
 - template<typename _InputIterator >
void [insert](#) (_InputIterator __first, _InputIterator __last)
 - void [insert](#) (initializer_list< value_type > __l)
 - key_compare [key_comp](#) () const
 - size_type [max_size](#) () const
 - multiset & [operator=](#) (const multiset &__x)
 - multiset & [operator=](#) (multiset &&__x)
 - multiset & [operator=](#) (initializer_list< value_type > __l)
 - reverse_iterator [rbegin](#) () const
 - reverse_iterator [rend](#) () const
 - size_type [size](#) () const
 - void [swap](#) (multiset &__x)
 - value_compare [value_comp](#) () const
-
- iterator [find](#) (const key_type &__x)
 - const_iterator [find](#) (const key_type &__x) const
-
- iterator [lower_bound](#) (const key_type &__x)
 - const_iterator [lower_bound](#) (const key_type &__x) const
-
- iterator [upper_bound](#) (const key_type &__x)
 - const_iterator [upper_bound](#) (const key_type &__x) const
-
- [std::pair](#)< iterator, iterator > [equal_range](#) (const key_type &__x)
 - [std::pair](#)< const_iterator, const_iterator > [equal_range](#) (const key_type &__x)
const
 - template<typename _K1, typename _C1, typename _A1 >
bool [operator==](#) (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)
 - template<typename _K1, typename _C1, typename _A1 >
bool [operator<](#) (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)

5.567.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::multiset<_Key, _Compare, _Alloc >

A standard container made up of elements, which can be retrieved in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an

associative container (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 84 of file `stl_multiset.h`.

5.567.2 Constructor & Destructor Documentation

5.567.2.1 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset () [inline]`

Default constructor creates no elements.

Definition at line 129 of file `stl_multiset.h`.

5.567.2.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset (const _Compare & __comp, const
allocator_type & __a = allocator_type()) [inline,
explicit]`

Creates a multiset with no elements.

Parameters

comp Comparator to use.

a An allocator object.

Definition at line 138 of file `stl_multiset.h`.

5.567.2.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset
(_InputIterator __first, _InputIterator __last) [inline]`

Builds a multiset from a range.

Parameters

first An input iterator.

last An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 152 of file stl_multiset.h.

```
5.567.2.4 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset
( _InputIterator __first, _InputIterator __last, const _Compare &
__comp, const allocator_type & __a = allocator_type() )
[inline]
```

Builds a multiset from a range.

Parameters

first An input iterator.

last An input iterator.

comp A comparison functor.

a An allocator object.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 168 of file stl_multiset.h.

```
5.567.2.5 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset ( const multiset< _Key, _Compare,
_Alloc > & __x ) [inline]
```

Multiset copy constructor.

Parameters

x A multiset of identical element and allocator types.

The newly-created multiset uses a copy of the allocation object used by *x*.

Definition at line 181 of file stl_multiset.h.

5.567.2.6 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset (multiset< _Key, _Compare, _Alloc >
&& __x) [inline]`

Multiset move constructor.

Parameters

x A multiset of identical element and allocator types.

The newly-created multiset contains the exact contents of *x*. The contents of *x* are a valid, but unspecified multiset.

Definition at line 192 of file stl_multiset.h.

5.567.2.7 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::multiset< _Key,
_Compare, _Alloc >::multiset (initializer_list< value_type > __l,
const _Compare & __comp = _Compare(), const allocator_type &
__a = allocator_type()) [inline]`

Builds a multiset from an [initializer_list](#).

Parameters

l An [initializer_list](#).

comp A comparison functor.

a An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *l.size()*).

Definition at line 205 of file stl_multiset.h.

5.567.3 Member Function Documentation

5.567.3.1 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 285 of file stl_multiset.h.

5.567.3.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 322 of file stl_multiset.h.

5.567.3.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 331 of file stl_multiset.h.

5.567.3.4 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::multiset< _Key,
_Compare, _Alloc >::clear () [inline]`

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 541 of file stl_multiset.h.

5.567.3.5 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::multiset<
_Key, _Compare, _Alloc >::count (const key_type & __x) const
[inline]`

Finds the number of elements with given key.

Parameters

x Key of elements to be located.

Returns

Number of elements with specified key.

Definition at line 552 of file stl_multiset.h.

5.567.3.6 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::crbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 340 of file stl_multiset.h.

5.567.3.7 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::crend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 349 of file stl_multiset.h.

5.567.3.8 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> bool std::multiset< _Key,
_Compare, _Alloc >::empty () const [inline]`

Returns true if the set is empty.

Definition at line 355 of file stl_multiset.h.

5.567.3.9 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 294 of file stl_multiset.h.

5.567.3.10 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::pair<iterator,
iterator> std::multiset< _Key, _Compare, _Alloc >::equal_range (
const key_type & __x) [inline]`

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 632 of file stl_multiset.h.

```
5.567.3.11  template<typename _Key, typename _Compare = std::less<_Key>,
                typename _Alloc = std::allocator<_Key>>> std::pair<const_iterator,
                const_iterator> std::multiset<_Key, _Compare, _Alloc
                >::equal_range( const key_type & __x ) const  [inline]
```

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 636 of file stl_multiset.h.

```
5.567.3.12  template<typename _Key, typename _Compare = std::less<_Key>,
                typename _Alloc = std::allocator<_Key>>> iterator std::multiset<
                _Key, _Compare, _Alloc >::erase( iterator __position )
                [inline]
```

Erases an element from a multiset.

Parameters

position An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 466 of file `stl_multiset.h`.

```
5.567.3.13 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::multiset<
_Key, _Compare, _Alloc >::erase ( const key_type & __x )
[inline]
```

Erases elements according to the provided key.

Parameters

x Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 496 of file `stl_multiset.h`.

```
5.567.3.14 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::erase ( iterator __first, iterator __last
) [inline]
```

Erases a `[first,last)` range of elements from a multiset.

Parameters

first Iterator pointing to the start of the range to be erased.

last Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 515 of file `stl_multiset.h`.

```
5.567.3.15  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> iterator std::multiset<
              _Key, _Compare, _Alloc >::find ( const key_type & __x )
              [inline]
```

Tries to locate an element in a set.

Parameters

x Element to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 570 of file `stl_multiset.h`.

```
5.567.3.16  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> const_iterator
              std::multiset< _Key, _Compare, _Alloc >::find ( const key_type &
              __x )const [inline]
```

Tries to locate an element in a set.

Parameters

x Element to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

5.567 std::multiset< _Key, _Compare, _Alloc > Class Template Reference 2779

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 574 of file `stl_multiset.h`.

```
5.567.3.17 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> allocator_type
std::multiset< _Key, _Compare, _Alloc >::get_allocator ( ) const
[inline]
```

Returns the memory allocation object.

Definition at line 276 of file `stl_multiset.h`.

```
5.567.3.18 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::insert ( const value_type & __x )
[inline]
```

Inserts an element into the multiset.

Parameters

x Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 396 of file `stl_multiset.h`.

```
5.567.3.19 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::insert ( iterator __position, const
value_type & __x ) [inline]
```

Inserts an element into the multiset.

Parameters

position An iterator that serves as a hint as to where the element should be inserted.

x Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a [std::set](#) the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 420 of file `stl_multiset.h`.

```
5.567.3.20  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>>> template<typename
              _InputIterator > void std::multiset< _Key, _Compare, _Alloc
              >::insert ( _InputIterator __first, _InputIterator __last )
              [inline]
```

A template function that tries to insert a range of elements.

Parameters

first Iterator pointing to the start of the range to be inserted.

last Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 433 of file `stl_multiset.h`.

```
5.567.3.21  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>>> void std::multiset<
              _Key, _Compare, _Alloc >::insert ( initializer_list< value_type >
              __l ) [inline]
```

Attempts to insert a list of elements into the multiset.

Parameters

list A `std::initializer_list<value_type>` of elements to be inserted.

5.567 std::multiset< _Key, _Compare, _Alloc > Class Template Reference 2781

Complexity similar to that of the range constructor.

Definition at line 445 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc >::insert().

Referenced by std::multiset< _Key, _Compare, _Alloc >::insert().

5.567.3.22 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> key_compare
std::multiset< _Key, _Compare, _Alloc >::key_comp () const
[inline]`

Returns the comparison object.

Definition at line 268 of file stl_multiset.h.

5.567.3.23 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::multiset<
_Key, _Compare, _Alloc >::lower_bound (const key_type & __x)
[inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 591 of file stl_multiset.h.

5.567.3.24 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> const_iterator
std::multiset< _Key, _Compare, _Alloc >::lower_bound (const
key_type & __x) const [inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 595 of file `stl_multiset.h`.

5.567.3.25 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::multiset<
_Key, _Compare, _Alloc >::max_size () const [inline]`

Returns the maximum size of the set.

Definition at line 365 of file `stl_multiset.h`.

5.567.3.26 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> multiset&
std::multiset< _Key, _Compare, _Alloc >::operator= (multiset<
_Key, _Compare, _Alloc > && __x) [inline]`

Multiset move assignment operator.

Parameters

x A multiset of identical element and allocator types.

The contents of *x* are moved into this multiset (without copying). *x* is a valid, but unspecified multiset.

Definition at line 235 of file `stl_multiset.h`.

5.567.3.27 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> multiset&
std::multiset< _Key, _Compare, _Alloc >::operator= (
initializer_list< value_type > __l) [inline]`

Multiset list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a multiset with copies of the elements in the initializer list *l*.

5.567 std::multiset< _Key, _Compare, _Alloc > Class Template Reference 2783

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 256 of file stl_multiset.h.

```
5.567.3.28 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> multiset&
std::multiset< _Key, _Compare, _Alloc >::operator= ( const
multiset< _Key, _Compare, _Alloc > & __x ) [inline]
```

Multiset assignment operator.

Parameters

x A multiset of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 220 of file stl_multiset.h.

```
5.567.3.29 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::rbegin ( ) const
[inline]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 303 of file stl_multiset.h.

```
5.567.3.30 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::multiset< _Key, _Compare, _Alloc >::rend ( ) const
[inline]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 312 of file stl_multiset.h.

```
5.567.3.31 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::multiset<
_Key, _Compare, _Alloc >::size ( ) const [inline]
```

Returns the size of the set.

Definition at line 360 of file stl_multiset.h.

```
5.567.3.32  template<typename _Key, typename _Compare = std::less<_Key>,
               typename _Alloc = std::allocator<_Key>> void std::multiset<
               _Key, _Compare, _Alloc >::swap ( multiset< _Key, _Compare,
               _Alloc > & __x ) [inline]
```

Swaps data with another multiset.

Parameters

x A multiset of the same element and allocator types.

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 380 of file stl_multiset.h.

Referenced by `std::swap()`.

```
5.567.3.33  template<typename _Key, typename _Compare = std::less<_Key>,
               typename _Alloc = std::allocator<_Key>> iterator std::multiset<
               _Key, _Compare, _Alloc >::upper_bound ( const key_type & __x )
               [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 607 of file stl_multiset.h.

```
5.567.3.34  template<typename _Key, typename _Compare = std::less<_Key>,
               typename _Alloc = std::allocator<_Key>> const_iterator
               std::multiset< _Key, _Compare, _Alloc >::upper_bound ( const
               key_type & __x ) const [inline]
```

Finds the end of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to the first element greater than key, or [end\(\)](#).

Definition at line 611 of file stl_multiset.h.

```
5.567.3.35 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> value_compare
std::multiset< _Key, _Compare, _Alloc >::value_comp ( ) const
[inline]
```

Returns the comparison object.

Definition at line 272 of file stl_multiset.h.

5.567.4 Friends And Related Function Documentation

```
5.567.4.1 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename _K1
, typename _C1, typename _A1 > bool operator< ( const multiset<
_K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > & )
[friend]
```

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

5.567.4.2 `template<typename _Key, typename _Compare = std::less<_Key>,
 typename _Alloc = std::allocator<_Key>> template<typename
 _K1, typename _C1, typename _A1 > bool operator==(const
 multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &
) [friend]`

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),  
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

The documentation for this class was generated from the following file:

- [stl_multiset.h](#)

5.568 std::mutex Class Reference

mutex

Public Types

- `typedef __native_type * native_handle_type`

Public Member Functions

- **mutex** (const [mutex](#) &)
- void **lock** ()
- native_handle_type **native_handle** ()
- [mutex](#) & **operator=** (const [mutex](#) &)
- bool **try_lock** ()
- void **unlock** ()

5.568.1 Detailed Description

mutex

Definition at line 62 of file mutex.

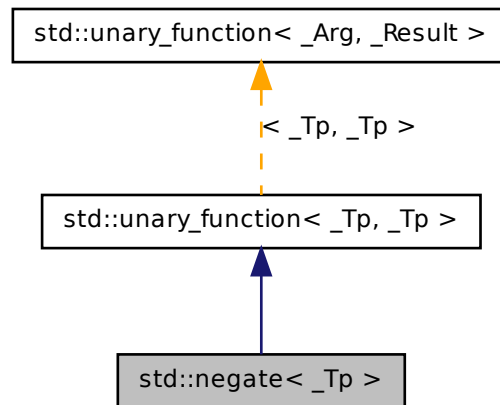
The documentation for this class was generated from the following file:

- [mutex](#)

5.569 std::negate< _Tp > Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::negate< _Tp >:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Tp` [result_type](#)

Public Member Functions

- `_Tp operator() (const _Tp &__x) const`

5.569.1 Detailed Description

`template<typename _Tp> struct std::negate< _Tp >`

One of the [math functors](#).

Definition at line 180 of file `stl_function.h`.

5.569.2 Member Typedef Documentation

5.569.2.1 `typedef _Tp std::unary_function< _Tp , _Tp >::argument_type` `[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.569.2.2 `typedef _Tp std::unary_function< _Tp , _Tp >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.570 `std::negative_binomial_distribution< _IntType >` Class Template Reference

A [negative_binomial_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` `result_type`

Public Member Functions

- `negative_binomial_distribution` (`_IntType __k=1`, `double __p=0.5`)
- `negative_binomial_distribution` (const `param_type` &__p)
- `_IntType k` () const
- `result_type max` () const
- `result_type min` () const
- template<typename `_UniformRandomNumberGenerator` >
`result_type operator()` (`_UniformRandomNumberGenerator` &__urng, const `param_type` &__p)
- template<typename `_UniformRandomNumberGenerator` >
`result_type operator()` (`_UniformRandomNumberGenerator` &__urng)
- double `p` () const
- void `param` (const `param_type` &__param)
- `param_type param` () const
- void `reset` ()

Friends

- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >
`std::basic_ostream<_CharT, _Traits>` & `operator<<` (`std::basic_ostream<_CharT, _Traits>` &, const `std::negative_binomial_distribution<_IntType1>` &)
- template<typename `_IntType1` >
bool `operator==` (const `std::negative_binomial_distribution<_IntType1>` &__d1, const `std::negative_binomial_distribution<_IntType1>` &__d2)
- template<typename `_IntType1`, typename `_CharT`, typename `_Traits` >
`std::basic_istream<_CharT, _Traits>` & `operator>>` (`std::basic_istream<_CharT, _Traits>` &, `std::negative_binomial_distribution<_IntType1>` &)

5.570.1 Detailed Description

`template<typename _IntType = int> class std::negative_binomial_distribution<_IntType>`

A `negative_binomial_distribution` random number distribution. The formula for the negative binomial probability mass function is $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Definition at line 3795 of file random.h.

5.570.2 Member Typedef Documentation

5.570.2.1 `template<typename _IntType = int> typedef _IntType
std::negative_binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 3802 of file random.h.

5.570.3 Member Function Documentation

5.570.3.1 `template<typename _IntType = int> _IntType
std::negative_binomial_distribution< _IntType >::k () const
[inline]`

Return the k parameter of the distribution.

Definition at line 3851 of file random.h.

5.570.3.2 `template<typename _IntType = int> result_type
std::negative_binomial_distribution< _IntType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3887 of file random.h.

5.570.3.3 `template<typename _IntType = int> result_type
std::negative_binomial_distribution< _IntType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3880 of file random.h.

5.570.3.4 `template<typename _IntType > template<typename
UniformRandomNumberGenerator > negative
binomial_distribution< _IntType >::result_type
std::negative_binomial_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng)`

Generating functions.

Definition at line 1078 of file random.tcc.

5.570.3.5 `template<typename _IntType = int> double
std::negative_binomial_distribution< _IntType >::p () const
[inline]`

Return the p parameter of the distribution.

Definition at line 3858 of file random.h.

5.570.3.6 `template<typename _IntType = int> param_type
std::negative_binomial_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 3865 of file random.h.

5.570.3.7 `template<typename _IntType = int> void std::negative_binomial_
distribution< _IntType >::param (const param_type & __param)
[inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 3873 of file random.h.

5.570.3.8 `template<typename _IntType = int> void
std::negative_binomial_distribution< _IntType >::reset ()
[inline]`

Resets the distribution state.

Definition at line 3844 of file random.h.

References std::gamma_distribution< _RealType >::reset().

5.570.4 Friends And Related Function Documentation

5.570.4.1 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_ostream<_CharT,
_Traits>& operator<< (std::basic_ostream< _CharT, _Traits > &
, const std::negative_binomial_distribution< _IntType1 > &)
[friend]`

Inserts a `negative_binomial_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `negative_binomial_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.570.4.2 `template<typename _IntType = int> template<typename _IntType1
> bool operator==(const std::negative_binomial_distribution<
_IntType1 > & __d1, const std::negative_binomial_distribution<
_IntType1 > & __d2) [friend]`

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3909 of file `random.h`.

5.570.4.3 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> (std::basic_istream< _CharT, _Traits > & ,
std::negative_binomial_distribution< _IntType1 > &) [friend]`

Extracts a `negative_binomial_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `negative_binomial_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

5.571 `std::negative_binomial_distribution< _IntType >::param_type` Struct Reference 2793

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.571 `std::negative_binomial_distribution< _IntType >::param_type` Struct Reference

Public Types

- typedef [negative_binomial_distribution< _IntType >](#) **distribution_type**

Public Member Functions

- **param_type** (`_IntType __k=1, double __p=0.5`)
- `_IntType k` () const
- `double p` () const

Friends

- `bool operator==` (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.571.1 Detailed Description

`template<typename _IntType = int> struct std::negative_binomial_distribution< _IntType >::param_type`

Parameter type.

Definition at line 3804 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.572 `std::nested_exception` Class Reference

Exception class with `exception_ptr` data member.

Inherited by `std::_Nested_exception< _Except >`.

Public Member Functions

- **nested_exception** (const [nested_exception](#) &)
- exception_ptr **nested_ptr** () const
- [nested_exception](#) & **operator=** (const [nested_exception](#) &)
- void **rethrow_nested** () const __attribute__((__noreturn__))

5.572.1 Detailed Description

Exception class with exception_ptr data member.

Definition at line 55 of file nested_exception.h.

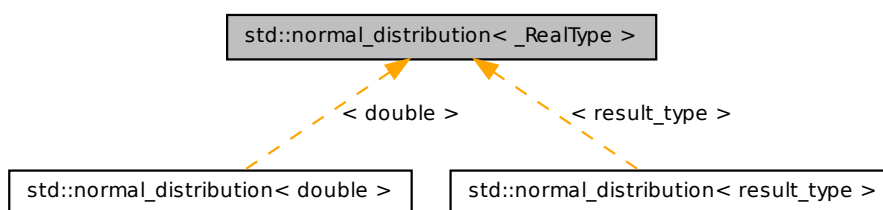
The documentation for this class was generated from the following file:

- [nested_exception.h](#)

5.573 std::normal_distribution< _RealType > Class Template Reference

A normal continuous distribution for random numbers.

Inheritance diagram for std::normal_distribution< _RealType >:



Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- [normal_distribution](#) ([result_type](#) __mean=[result_type](#)(0), [result_type](#) __stddev=[result_type](#)(1))
- **normal_distribution** (const [param_type](#) &__p)
- [result_type](#) max () const
- _RealType mean () const
- [result_type](#) min () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) operator() (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) operator() (_UniformRandomNumberGenerator &__urng)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- void [reset](#) ()
- _RealType [stddev](#) () const

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_ostream](#)< _CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::normal_distribution](#)< _RealType1 > &)
- template<typename _RealType1 >
bool [operator==](#) (const [std::normal_distribution](#)< _RealType1 > &__d1, const [std::normal_distribution](#)< _RealType1 > &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &, [std::normal_distribution](#)< _RealType1 > &)

5.573.1 Detailed Description

template<typename _RealType = double> class std::normal_distribution< _RealType >

A normal continuous distribution for random numbers. The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

Definition at line 1993 of file random.h.

5.573.2 Member Typedef Documentation

5.573.2.1 `template<typename _RealType = double> typedef _RealType std::normal_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2000 of file random.h.

5.573.3 Constructor & Destructor Documentation

5.573.3.1 `template<typename _RealType = double> std::normal_distribution< _RealType >::normal_distribution (result_type __mean = result_type (0), result_type __stddev = result_type (1)) [inline, explicit]`

Constructs a normal distribution with parameters *mean* and standard deviation.

Definition at line 2038 of file random.h.

5.573.4 Member Function Documentation

5.573.4.1 `template<typename _RealType = double> result_type std::normal_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2095 of file random.h.

Referenced by `std::normal_distribution< result_type >::max()`.

5.573.4.2 `template<typename _RealType = double> _RealType std::normal_distribution< _RealType >::mean () const [inline]`

Returns the mean of the distribution.

Definition at line 2059 of file random.h.

5.573.4.3 `template<typename _RealType = double> result_type
std::normal_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2088 of file random.h.

Referenced by std::normal_distribution< result_type >::min().

5.573.4.4 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::normal_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2103 of file random.h.

Referenced by std::normal_distribution< result_type >::operator()().

5.573.4.5 `template<typename _RealType > template<typename
_UniformRandomNumberGenerator > normal_distribution<
_RealType >::result_type std::normal_distribution< _RealType
>::operator() (_UniformRandomNumberGenerator & __urng,
const param_type & __param)`

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

Definition at line 1644 of file random.tcc.

References std::log(), and std::sqrt().

5.573.4.6 `template<typename _RealType = double> void
std::normal_distribution< _RealType >::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 2081 of file random.h.

5.573.4.7 `template<typename _RealType = double> param_type
std::normal_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 2073 of file random.h.

Referenced by `std::normal_distribution< result_type >::operator()()`.

5.573.4.8 `template<typename _RealType = double> void
std::normal_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 2052 of file random.h.

Referenced by `std::poisson_distribution< _IntType >::reset()`, `std::binomial_distribution< _IntType >::reset()`, `std::student_t_distribution< _RealType >::reset()`, `std::gamma_distribution< result_type >::reset()`, and `std::lognormal_distribution< _RealType >::reset()`.

5.573.4.9 `template<typename _RealType = double> _RealType
std::normal_distribution< _RealType >::stddev () const
[inline]`

Returns the standard deviation of the distribution.

Definition at line 2066 of file random.h.

5.573.5 Friends And Related Function Documentation

5.573.5.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::normal_distribution< _RealType1 > &) [friend]`

Inserts a `normal_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `normal_distribution` random number distribution.

5.574 `std::normal_distribution<_RealType>::param_type` Struct Reference 2799

Returns

The output stream with the state of `__x` inserted or in an error state.

5.573.5.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::normal_distribution<
_RealType1 > & __d1, const std::normal_distribution<_RealType1
> & __d2) [friend]`

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

5.573.5.3 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream<_CharT, _Traits> & ,
std::normal_distribution<_RealType1> &) [friend]`

Extracts a normal_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A normal_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.574 `std::normal_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [normal_distribution<_RealType>](#) `distribution_type`

Public Member Functions

- **param_type** (_RealType __mean=_RealType(0), _RealType __stddev=_RealType(1))
- _RealType **mean** () const
- _RealType **stddev** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.574.1 Detailed Description

template<typename _RealType = double> struct std::normal_distribution< _RealType >::param_type

Parameter type.

Definition at line 2002 of file random.h.

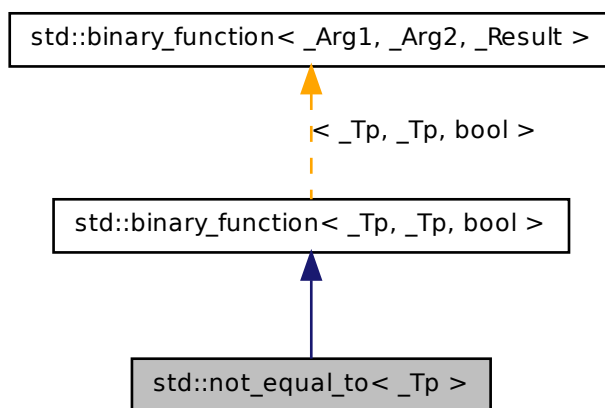
The documentation for this struct was generated from the following file:

- [random.h](#)

5.575 std::not_equal_to< _Tp > Struct Template Reference

One of the [comparison functors](#).

Inheritance diagram for std::not_equal_to< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__x, const _Tp &__y) const`

5.575.1 Detailed Description

```
template<typename _Tp> struct std::not_equal_to< _Tp >
```

One of the [comparison functors](#).

Definition at line 208 of file `stl_function.h`.

5.575.2 Member Typedef Documentation

5.575.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.575.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.575.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

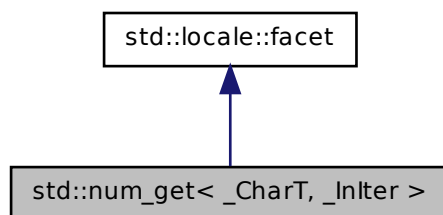
- [stl_function.h](#)

5.576 `std::num_get< _CharT, _InIter >` Class Template Reference

Primary class template [num_get](#).

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

Inheritance diagram for std::num_get< _CharT, _InIter >:



Public Types

- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

Public Member Functions

- `num_get` (`size_t __refs=0`)
- `template<typename _ValueT >`
`_InIter _M_extract_int` (`_InIter __beg, _InIter __end, ios_base &__io, ios_base::iostate &__err, _ValueT &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, void *&__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, bool &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v`) `const`
- `iter_type get` (`iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v`) `const`

- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, float &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, double &__v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long double &__v) const`

Static Public Attributes

- static `locale::id id`

Protected Member Functions

- virtual `~num_get ()`
- `iter_type _M_extract_float (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const`
- `template<typename _ValueT > iter_type _M_extract_int (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const`
- `template<typename _CharT2 > __gnu_cxx::__enable_if<__is_char<_CharT2>::__value, int>::__type _M_find (const _CharT2 *, size_t __len, _CharT2 __c) const`
- `template<typename _CharT2 > __gnu_cxx::__enable_if<!__is_char<_CharT2>::__value, int>::__type _M_find (const _CharT2 * __zero, size_t __len, _CharT2 __c) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &__err, float &) const`

- virtual [iter_type do_get](#) ([iter_type](#), [iter_type](#), [ios_base &](#), [ios_base::iostate &__err](#), double &) const
- virtual [iter_type do_get](#) ([iter_type](#), [iter_type](#), [ios_base &](#), [ios_base::iostate &__err](#), long double &) const
- virtual [iter_type do_get](#) ([iter_type](#), [iter_type](#), [ios_base &](#), [ios_base::iostate &__err](#), void *&) const

Static Protected Member Functions

- static [__c_locale _S_clone_c_locale](#) ([__c_locale &__cloc](#)) throw ()
- static void [_S_create_c_locale](#) ([__c_locale &__cloc](#), const char *__s, [__c_locale __old](#)=0)
- static void [_S_destroy_c_locale](#) ([__c_locale &__cloc](#))
- static [__c_locale _S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static [__c_locale _S_lc_ctype_c_locale](#) ([__c_locale __cloc](#), const char *__s)

Friends

- class [locale::_Impl](#)

5.576.1 Detailed Description

template<typename _CharT, typename _InIter> class std::num_get< _CharT, _InIter >

Primary class template [num_get](#).

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators. The [num_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [num_get](#) facet.

Definition at line 1911 of file [locale_facets.h](#).

5.576.2 Member Typedef Documentation

5.576.2.1 **template<typename _CharT, typename _InIter> typedef _CharT std::num_get< _CharT, _InIter >::char_type**

Public typedefs.

Definition at line 1917 of file [locale_facets.h](#).

5.576.2.2 `template<typename _CharT, typename _InIter > typedef _InIter
std::num_get< _CharT, _InIter >::iter_type`

Public typedefs.

Definition at line 1918 of file locale_facets.h.

5.576.3 Constructor & Destructor Documentation

5.576.3.1 `template<typename _CharT, typename _InIter > std::num_get<
_CharT, _InIter >::num_get (size_t __refs = 0) [inline,
explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 1932 of file locale_facets.h.

5.576.3.2 `template<typename _CharT, typename _InIter > virtual
std::num_get< _CharT, _InIter >::~~num_get () [inline,
protected, virtual]`

Destructor.

Definition at line 2101 of file locale_facets.h.

5.576.4 Member Function Documentation

5.576.4.1 `template<typename _CharT, typename _InIter > _InIter
std::num_get< _CharT, _InIter >::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, float &
__v) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 686 of file locale_facets.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::reserve().

5.576.4.2 `template<typename _CharT, typename _InIter > _InIter
std::num_get< _CharT, _InIter >::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, bool &
__v) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 590 of file locale_facets.tcc.

References std::ios_base::M_getloc(), std::ios_base::boolalpha, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::flags(), and std::ios_base::goodbit.

Referenced by std::num_get< _CharT, _InIter >::get().

5.576.4.3 `template<typename _CharT, typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, long &
__v) const [inline, protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2169 of file locale_facets.h.

5.576.4.4 `template<typename _CharT, typename _InIter > virtual iter_type
std::num_get< _CharT, _InIter >::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
unsigned short & __v) const [inline, protected,
virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2174 of file locale_facets.h.

```
5.576.4.5 template<typename _CharT , typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
unsigned int & __v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2179 of file locale_facets.h.

```
5.576.4.6 template<typename _CharT , typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
unsigned long & __v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2184 of file locale_facets.h.

```
5.576.4.7 template<typename _CharT, typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, long  
long & __v ) const [inline, protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2190 of file locale_facets.h.

```
5.576.4.8 template<typename _CharT , typename _InIter > virtual iter_type  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err,  
unsigned long long & __v ) const [inline, protected,  
virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2195 of file locale_facets.h.

```
5.576.4.9 template<typename _CharT , typename _InIter > _InIter  
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, long  
double & __v ) const [protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.
end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 733 of file locale_facets.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::ios_base::eofbit`, and `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`.

5.576.4.10 `template<typename _CharT, typename _InIter> _InIter
std::num_get<_CharT, _InIter>::do_get (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err,
double & __v) const [protected, virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 701 of file locale_facets.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::ios_base::eofbit`, and `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`.


```
5.576.4.11 template<typename _CharT, typename _InIter > _InIter
std::num_get< _CharT, _InIter >::do_get ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, void
*& __v ) const [protected, virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

[get\(\)](#) for more details.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 748 of file locale_facets.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, and `std::ios_base::hex`.

```
5.576.4.12 template<typename _CharT, typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long & __v )
const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & [ios_base::basefield](#). If equal to [ios_base::oct](#), parses like the `scanf` `o` specifier. Else if equal to [ios_base::hex](#), parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios_base::failbit](#) and leaves *v* unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 1994 of file locale_facets.h.

References [std::num_get<_CharT, _InIter >::do_get\(\)](#).

```
5.576.4.13 template<typename _CharT , typename _InIter > iter_type
std::num_get<_CharT, _InIter >::get ( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, double & __v )
const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

The input characters are parsed like the scanf *g* specifier. The matching type length modifier is also used.

The decimal point character used is [numpunct::decimal_point\(\)](#). Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios_base::failbit](#) and leaves *v* unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2058 of file locale_facets.h.

References std::num_get< _CharT, _InIter >::do_get().

```
5.576.4.14 template<typename _CharT , typename _InIter > iter_type  
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type  
__end, ios_base & __io, ios_base::iostate & __err, void *& __v )  
const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling [num_get::do_get\(\)](#).

The input characters are parsed like the scanf *p* specifier.

Digit grouping is interpreted according to [num_punct::grouping\(\)](#) and [num_punct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets *err* to [ios_base::failbit](#).

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to [ios_base::failbit](#) and leaves *v* unaltered. Sets *err* to [ios_base::eofbit](#) if the stream is emptied.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2095 of file locale_facets.h.

References std::num_get< _CharT, _InIter >::do_get().

5.576.4.15 `template<typename _CharT, typename _InIter> iter_type
std::num_get<_CharT, _InIter>::get(iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned long
& __v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & [ios_base::basefield](#). If equal to [ios_base::oct](#), parses like the `scanf o` specifier. Else if equal to [ios_base::hex](#), parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [num_punct::grouping\(\)](#) and [num_punct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets `err` to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to [ios_base::failbit](#) and leaves *v* unaltered. Sets `err` to [ios_base::eofbit](#) if the stream is emptied.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

Returns

Iterator after reading.

Definition at line 2009 of file `locale_facets.h`.

References [std::num_get<_CharT, _InIter>::do_get\(\)](#).

5.576.4.16 `template<typename _CharT, typename _InIter> iter_type
std::num_get<_CharT, _InIter>::get(iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, bool & __v)
const [inline]`

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truenamename()` or `ctype<CharT>::falsenamename()`. Sets *v* to true or false if successful. Sets *err* to `ios_base::failbit` if reading the string fails. Sets *err* to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set *err* to `ios_base::failbit`.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

Returns

Iterator after reading.

Definition at line 1958 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

Referenced by `std::basic_istream< _CharT, _Traits >::operator>>()`.

5.576.4.17 `template<typename _CharT, typename _InIter> iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned long
long & __v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for v , v is set. Otherwise, sets `err` to `ios_base::failbit` and leaves v unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2020 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter >::do_get()`.

```
5.576.4.18 template<typename _CharT, typename _InIter > iter_type
std::num_get<_CharT, _InIter >::get( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, float & __v )
const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable v . It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for v , v is set. Otherwise, sets `err` to `ios_base::failbit` and leaves v unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

in Start of input stream.
end End of input stream.
io Source of locale and flags.
err Error flags to set.
v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2053 of file locale_facets.h.

References std::num_get< _CharT, _InIter >::do_get().

5.576.4.19 `template<typename _CharT , typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned int &
__v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & [ios_base::basefield](#). If equal to [ios_base::oct](#), parses like the scanf o specifier. Else if equal to [ios_base::hex](#), parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios_base::failbit](#) and leaves *v* unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

in Start of input stream.

end End of input stream.

io Source of locale and flags.

err Error flags to set.

v Value to format and insert.

Returns

Iterator after reading.

Definition at line 2004 of file locale_facets.h.

References std::num_get< _CharT, _InIter >::do_get().

5.576.4.20 `template<typename _CharT, typename _InIter> iter_type
std::num_get<_CharT, _InIter>::get(iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long double &
__v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is [numpunct::decimal_point\(\)](#). Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets *err* to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to [ios_base::failbit](#) and leaves *v* unaltered. Sets *err* to [ios_base::eofbit](#) if the stream is emptied.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

Returns

Iterator after reading.

Definition at line 2063 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

5.576.4.21 `template<typename _CharT, typename _InIter> iter_type
std::num_get<_CharT, _InIter>::get(iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, long long & __v
) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling [num_get::do_get\(\)](#).

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

Returns

Iterator after reading.

Definition at line 2015 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

```
5.576.4.22 template<typename _CharT , typename _InIter > iter_type
std::num_get< _CharT, _InIter >::get ( iter_type __in, iter_type
__end, ios_base & __io, ios_base::iostate & __err, unsigned short
& __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to [numpunct::grouping\(\)](#) and [numpunct::thousands_sep\(\)](#). If the pattern of digit groups isn't consistent, sets err to [ios_base::failbit](#).

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to [ios_base::failbit](#) and leaves *v* unaltered. Sets err to [ios_base::eofbit](#) if the stream is emptied.

Parameters

- in* Start of input stream.
- end* End of input stream.
- io* Source of locale and flags.
- err* Error flags to set.
- v* Value to format and insert.

Returns

Iterator after reading.

Definition at line 1999 of file locale_facets.h.

References [std::num_get<_CharT, _InIter>::do_get\(\)](#).

5.576.5 Member Data Documentation

5.576.5.1 `template<typename _CharT, typename _InIter> locale::id std::num_get<_CharT, _InIter>::id` [static]

Numpunct facet id.

Definition at line 1922 of file locale_facets.h.

The documentation for this class was generated from the following files:

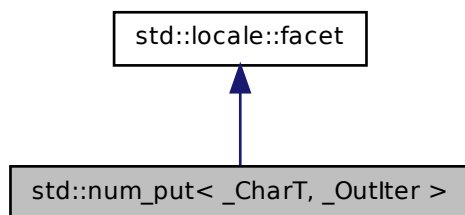
- [locale_facets.h](#)
- [locale_facets.tcc](#)

5.577 `std::num_put<_CharT, _OutIter>` Class Template Reference

Primary class template [num_put](#).

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

Inheritance diagram for std::num_put< _CharT, _OutIter >:



Public Types

- typedef `_CharT` `char_type`
- typedef `_OutIter` `iter_type`

Public Member Functions

- `num_put` (`size_t __refs=0`)
- `template<typename _ValueT >`
`_OutIter _M_insert_float` (`_OutIter __s`, `ios_base &__io`, `_CharT __fill`, `char __mod`, `_ValueT __v`) `const`
- `template<typename _ValueT >`
`_OutIter _M_insert_int` (`_OutIter __s`, `ios_base &__io`, `_CharT __fill`, `_ValueT __v`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__f`, `char_type __fill`, `const void *__v`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__f`, `char_type __fill`, `bool __v`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__f`, `char_type __fill`, `long __v`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__f`, `char_type __fill`, `unsigned long __v`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__f`, `char_type __fill`, `long long __v`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__f`, `char_type __fill`, `unsigned long long __v`) `const`

- `iter_type put (iter_type __s, ios_base &__f, char_type __fill, double __v) const`
- `iter_type put (iter_type __s, ios_base &__f, char_type __fill, long double __v) const`

Static Public Attributes

- static `locale::id id`

Protected Member Functions

- virtual `~num_put ()`
- void `_M_group_float (const char *__grouping, size_t __grouping_size, char_type __sep, const char_type *__p, char_type *__new, char_type *__cs, int &__len) const`
- void `_M_group_int (const char *__grouping, size_t __grouping_size, char_type __sep, ios_base &__io, char_type *__new, char_type *__cs, int &__len) const`
- template<typename _ValueT >
`iter_type _M_insert_float (iter_type, ios_base &__io, char_type __fill, char __mod, _ValueT __v) const`
- template<typename _ValueT >
`iter_type _M_insert_int (iter_type, ios_base &__io, char_type __fill, _ValueT __v) const`
- void `_M_pad (char_type __fill, streamsize __w, ios_base &__io, char_type *__new, const char_type *__cs, int &__len) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, bool __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const`
- virtual `iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, double __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, long double __v) const`
- virtual `iter_type do_put (iter_type, ios_base &, char_type __fill, const void *__v) const`

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static _GLIBCXX_CONST const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Friends

- class **locale::_Impl**

5.577.1 Detailed Description

template<typename _CharT, typename _OutIter> class std::num_put<_CharT, _OutIter>

Primary class template [num_put](#).

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators. The [num_put](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [num_put](#) facet.

Definition at line 2249 of file locale_facets.h.

5.577.2 Member Typedef Documentation

5.577.2.1 **template<typename _CharT, typename _OutIter> typedef _CharT std::num_put<_CharT, _OutIter>::char_type**

Public typedefs.

Definition at line 2255 of file locale_facets.h.

5.577.2.2 **template<typename _CharT, typename _OutIter> typedef _OutIter std::num_put<_CharT, _OutIter>::iter_type**

Public typedefs.

Definition at line 2256 of file locale_facets.h.

5.577.3 Constructor & Destructor Documentation

5.577.3.1 `template<typename _CharT, typename _OutIter > std::num_put<_CharT, _OutIter >::num_put (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 2270 of file locale_facets.h.

5.577.3.2 `template<typename _CharT, typename _OutIter > virtual std::num_put< _CharT, _OutIter >::~~num_put () [inline, protected, virtual]`

Destructor.

Definition at line 2449 of file locale_facets.h.

5.577.4 Member Function Documentation

5.577.4.1 `template<typename _CharT, typename _OutIter > virtual iter_type std::num_put< _CharT, _OutIter >::do_put (iter_type __s, ios_base & __io, char_type __fill, unsigned long __v) const [inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

s Stream to write to.

io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2473 of file locale_facets.h.

5.577.4.2 `template<typename _CharT, typename _OutIter > virtual
iter_type std::num_put<_CharT, _OutIter >::do_put (iter_type
__s, ios_base & __io, char_type __fill, long long __v) const
[inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

s Stream to write to.

io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2479 of file locale_facets.h.

5.577.4.3 `template<typename _CharT, typename _OutIter > virtual iter_type
std::num_put<_CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, long __v) const [inline,
protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

s Stream to write to.

io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2469 of file locale_facets.h.

5.577.4.4 `template<typename _CharT, typename _OutIter > virtual iter_type
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, unsigned long long __v) const
[inline, protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

s Stream to write to.
io Source of locale and flags.
fill Char_type to use for filling.
v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2484 of file locale_facets.h.

5.577.4.5 `template<typename _CharT, typename _OutIter > _OutIter
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, bool __v) const [protected,
virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

s Stream to write to.
io Source of locale and flags.
fill Char_type to use for filling.
v Value to format and insert.

Returns

Iterator after writing.

Definition at line 1089 of file locale_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::adjustfield`, `std::ios_base::boolalpha`, `std::ios_base::flags()`, `std::ios_base::left`, and `std::ios_base::width()`.

Referenced by `std::num_put< _CharT, _OutIter >::put()`.

5.577.4.6 `template<typename _CharT, typename _OutIter> _OutIter
std::num_put<_CharT, _OutIter>::do_put (iter_type __s,
ios_base & __io, char_type __fill, long double __v) const
[protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 1155 of file locale_facets.tcc.

5.577.4.7 `template<typename _CharT, typename _OutIter> _OutIter
std::num_put<_CharT, _OutIter>::do_put (iter_type
__s, ios_base & __io, char_type __fill, double __v) const
[protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 1141 of file locale_facets.tcc.

5.577.4.8 `template<typename _CharT, typename _OutIter > _OutIter
std::num_put< _CharT, _OutIter >::do_put (iter_type __s,
ios_base & __io, char_type __fill, const void * __v) const
[protected, virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 1162 of file locale_facets.tcc.

References `std::ios_base::flags()`, `std::ios_base::hex`, and `std::ios_base::uppercase`.

5.577.4.9 `template<typename _CharT, typename _OutIter > iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base &
__f, char_type __fill, long __v) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`. The decimal point character used is `numput::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2330 of file locale_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

5.577.4.10 `template<typename _CharT, typename _OutIter > iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, long long __v) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf *o* specifier. Else if equal to `ios_base::hex`, formats like *x* or *X* with `ios_base::uppercase` unset or set respectively. Otherwise, formats like *d*, *ld*, *lld* for signed and *u*, *lu*, *llu* for unsigned values. Note that if both *oct* and *hex* are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

s Stream to write to.
io Source of locale and flags.
fill Char_type to use for filling.
v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2340 of file locale_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

5.577.4.11 `template<typename _CharT, typename _OutIter> iter_type
 std::num_put<_CharT, _OutIter>::put (iter_type __s, ios_base
 & __f, char_type __fill, unsigned long __v) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf` `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

Parameters

s Stream to write to.
io Source of locale and flags.
fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2334 of file locale_facets.h.

References std::num_put< _CharT, _OutIter >::do_put().

5.577.4.12 `template<typename _CharT, typename _OutIter > iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, bool __v) const [inline]`

Numeric formatting.

Formats the boolean *v* and inserts it into a stream. It does so by calling [num_put::do_put\(\)](#).

If [ios_base::boolalpha](#) is set, writes ctype<CharT>::truenam() or ctype<CharT>::falsenam(). Otherwise formats *v* as an int.

Parameters

s Stream to write to.

io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2288 of file locale_facets.h.

References std::num_put< _CharT, _OutIter >::do_put().

5.577.4.13 `template<typename _CharT, typename _OutIter > iter_type
std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, long double __v) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling [num_put::do_put\(\)](#).

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2397 of file `locale_facets.h`.

References `std::num_put<_CharT, _OutIter>::do_put()`.

```
5.577.4.14  template<typename _CharT, typename _OutIter> iter_type
             std::num_put<_CharT, _OutIter>::put ( iter_type __s, ios_base
             & __f, char_type __fill, const void * __v ) const  [inline]
```

Numeric formatting.

Formats the pointer value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats *v* as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.
- fill* Char_type to use for filling.
- v* Value to format and insert.

Returns

Iterator after writing.

Definition at line 2418 of file locale_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

```
5.577.4.15 template<typename _CharT, typename _OutIter > iter_type  
std::num_put< _CharT, _OutIter >::put ( iter_type __s, ios_base  
& __f, char_type __fill, unsigned long long __v ) const  
[inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::basefield`. If equal to `ios_base::oct`, formats like the printf `o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

Parameters

- s* Stream to write to.
- io* Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2344 of file locale_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

5.577.4.16 `template<typename _CharT, typename _OutIter> iter_type
std::num_put<_CharT, _OutIter >::put (iter_type __s, ios_base
& __f, char_type __fill, double __v) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the printf *f* specifier. Else if equal to `ios_base::scientific`, formats like *e* or *E* with `ios_base::uppercase` unset or set respectively. Otherwise, formats like *g* or *G* depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like *g* or *G*.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. The decimal point character used is `num_punct::decimal_point()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

s Stream to write to.

io Source of locale and flags.

fill Char_type to use for filling.

v Value to format and insert.

Returns

Iterator after writing.

Definition at line 2393 of file locale_facets.h.

References std::num_put< _CharT, _OutIter >::do_put().

5.577.5 Member Data Documentation

5.577.5.1 `template<typename _CharT, typename _OutIter > locale::id
std::num_put< _CharT, _OutIter >::id [static]`

Numpunct facet id.

Definition at line 2260 of file locale_facets.h.

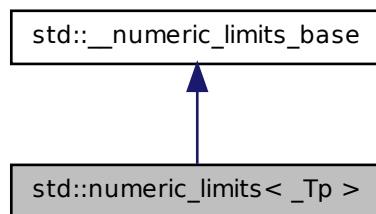
The documentation for this class was generated from the following files:

- [locale_facets.h](#)
- [locale_facets.tcc](#)

5.578 std::numeric_limits< _Tp > Struct Template Reference

Properties of fundamental types.

Inheritance diagram for std::numeric_limits< _Tp >:



Static Public Member Functions

- static `_Tp` [denorm_min](#) () throw ()
- static `_Tp` [epsilon](#) () throw ()
- static `_Tp` [infinity](#) () throw ()
- static `_Tp` [lowest](#) () throw ()
- static `_Tp` [max](#) () throw ()
- static `_Tp` [min](#) () throw ()
- static `_Tp` [quiet_NaN](#) () throw ()
- static `_Tp` [round_error](#) () throw ()
- static `_Tp` [signaling_NaN](#) () throw ()

Static Public Attributes

- static const int [digits](#)
- static const int [digits10](#)
- static const [float_denorm_style](#) [has_denorm](#)
- static const bool [has_denorm_loss](#)
- static const bool [has_infinity](#)
- static const bool [has_quiet_NaN](#)
- static const bool [has_signaling_NaN](#)
- static const bool [is_bounded](#)
- static const bool [is_exact](#)
- static const bool [is_iec559](#)
- static const bool [is_integer](#)
- static const bool [is_modulo](#)
- static const bool [is_signed](#)
- static const bool [is_specialized](#)
- static const int [max_digits10](#)
- static const int [max_exponent](#)
- static const int [max_exponent10](#)
- static const int [min_exponent](#)
- static const int [min_exponent10](#)
- static const int [radix](#)
- static const [float_round_style](#) [round_style](#)
- static const bool [tinyness_before](#)
- static const bool [traps](#)

5.578.1 Detailed Description

template<typename _Tp> struct std::numeric_limits< _Tp >

Properties of fundamental types. This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

_GLIBCXX_RESOLVE_LIB_DEFECTS: DRs 201 and 184 (hi Gaby!) are noted, but not incorporated in this documented (yet).

Definition at line 284 of file limits.

5.578.2 Member Function Documentation

5.578.2.1 **template<typename _Tp > static _Tp std::numeric_limits< _Tp >::denorm_min () throw () [inline, static]**

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

Definition at line 313 of file limits.

5.578.2.2 **template<typename _Tp > static _Tp std::numeric_limits< _Tp >::epsilon () throw () [inline, static]**

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

Definition at line 298 of file limits.

5.578.2.3 **template<typename _Tp > static _Tp std::numeric_limits< _Tp >::infinity () throw () [inline, static]**

The representation of positive infinity, if `has_infinity`.

Definition at line 302 of file limits.

5.578.2.4 **template<typename _Tp > static _Tp std::numeric_limits< _Tp >::lowest () throw () [inline, static]**

A finite value `x` such that there is no other finite value `y` where `y < x`.

Definition at line 294 of file limits.

5.578.2.5 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::max () throw () [inline, static]`

The maximum finite value.

Definition at line 290 of file limits.

5.578.2.6 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::min () throw () [inline, static]`

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Definition at line 288 of file limits.

5.578.2.7 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::quiet_NaN () throw () [inline, static]`

The representation of a quiet *Not a Number*, if has_quiet_NaN.

Definition at line 306 of file limits.

5.578.2.8 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::round_error () throw () [inline, static]`

The maximum rounding error measurement (see LIA-1).

Definition at line 300 of file limits.

5.578.2.9 `template<typename _Tp > static _Tp std::numeric_limits< _Tp >::signaling_NaN () throw () [inline, static]`

The representation of a signaling *Not a Number*, if has_signaling_NaN.

Definition at line 309 of file limits.

5.578.3 Member Data Documentation

5.578.3.1 `const int std::__numeric_limits_base::digits [static, inherited]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 199 of file limits.

5.578.3.2 `const int std::__numeric_limits_base::digits10` `[static, inherited]`

The number of base 10 digits that can be represented without change.

Definition at line 201 of file limits.

5.578.3.3 `const float_denorm_style std::__numeric_limits_base::has_denorm` `[static, inherited]`

See [std::float_denorm_style](#) for more information.

Definition at line 244 of file limits.

5.578.3.4 `const bool std::__numeric_limits_base::has_denorm_loss` `[static, inherited]`

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result. [18.2.1.2]/42

Definition at line 247 of file limits.

5.578.3.5 `const bool std::__numeric_limits_base::has_infinity` `[static, inherited]`

True if the type has a representation for positive infinity.

Definition at line 236 of file limits.

5.578.3.6 `const bool std::__numeric_limits_base::has_quiet_NaN` `[static, inherited]`

True if the type has a representation for a quiet (non-signaling) *Not a Number*.

Definition at line 239 of file limits.

5.578.3.7 `const bool std::__numeric_limits_base::has_signaling_NaN` `[static, inherited]`

True if the type has a representation for a signaling *Not a Number*.

Definition at line 242 of file limits.

5.578.3.8 `const bool std::__numeric_limits_base::is_bounded` `[static, inherited]`

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types. [18.2.1.2]/54

Definition at line 255 of file limits.

5.578.3.9 `const bool std::__numeric_limits_base::is_exact` `[static, inherited]`

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer. [18.2.1.2]/15

Definition at line 216 of file limits.

5.578.3.10 `const bool std::__numeric_limits_base::is_iec559` `[static, inherited]`

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 251 of file limits.

5.578.3.11 `const bool std::__numeric_limits_base::is_integer` `[static, inherited]`

True if the type is integer. Is this supposed to be if the type is integral?

Definition at line 211 of file limits.

5.578.3.12 `const bool std::__numeric_limits_base::is_modulo` `[static, inherited]`

True if the type is modulo, that is, if it is possible to add two positive numbers and have a result that wraps around to a third number that is less. Typically false for floating types, true for unsigned integers, and true for signed integers.

Definition at line 260 of file limits.

5.578.3.13 `const bool std::__numeric_limits_base::is_signed` `[static, inherited]`

True if the type is signed.

Definition at line 208 of file limits.

5.578.3.14 `const bool std::__numeric_limits_base::is_specialized` `[static, inherited]`

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 194 of file limits.

5.578.3.15 `const int std::__numeric_limits_base::max_digits10` `[static, inherited]`

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 205 of file limits.

5.578.3.16 `const int std::__numeric_limits_base::max_exponent` `[static, inherited]`

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 230 of file limits.

5.578.3.17 `const int std::__numeric_limits_base::max_exponent10` `[static, inherited]`

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 233 of file limits.

5.578.3.18 `const int std::__numeric_limits_base::min_exponent` `[static, inherited]`

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 223 of file limits.

5.578.3.19 `const int std::__numeric_limits_base::min_exponent10` `[static, inherited]`

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 226 of file limits.

5.578.3.20 `const int std::__numeric_limits_base::radix` `[static, inherited]`

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 219 of file limits.

5.578.3.21 `const float_round_style std::__numeric_limits_base::round_style` `[static, inherited]`

See [std::float_round_style](#) for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 269 of file limits.

5.578.3.22 `const bool std::__numeric_limits_base::tinyness_before` `[static, inherited]`

True if tininess is detected before rounding. (see IEC 559)

Definition at line 265 of file limits.

5.578.3.23 `const bool std::__numeric_limits_base::traps` `[static, inherited]`

True if trapping is implemented for this type.

Definition at line 263 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.579 `std::numeric_limits< bool >` Struct Template Reference

[`numeric_limits<bool>`](#) specialization.

Static Public Member Functions

- static bool **denorm_min** () throw ()
- static bool **epsilon** () throw ()
- static bool **infinity** () throw ()
- static bool **lowest** () throw ()
- static bool **max** () throw ()
- static bool **min** () throw ()
- static bool **quiet_NaN** () throw ()
- static bool **round_error** () throw ()
- static bool **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [`float_denorm_style`](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [`float_round_style`](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.579.1 Detailed Description

`template<> struct std::numeric_limits< bool >`

[numeric_limits<bool>](#) specialization.

Definition at line 335 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.580 `std::numeric_limits< char >` Struct Template Reference

[numeric_limits<char>](#) specialization.

Static Public Member Functions

- static char **denorm_min** () throw ()
- static char **epsilon** () throw ()
- static char **infinity** () throw ()
- static char **lowest** () throw ()
- static char **max** () throw ()
- static char **min** () throw ()
- static char **quiet_NaN** () throw ()
- static char **round_error** () throw ()
- static char **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.580.1 Detailed Description

`template<> struct std::numeric_limits< char >`

[numeric_limits<char>](#) specialization.

Definition at line 395 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.581 `std::numeric_limits< char16_t >` Struct Template Reference

[numeric_limits<char16_t>](#) specialization.

Static Public Member Functions

- static `char16_t` **denorm_min** () throw ()
- static `char16_t` **epsilon** () throw ()
- static `char16_t` **infinity** () throw ()
- static `char16_t` **lowest** () throw ()
- static `char16_t` **max** () throw ()
- static `char16_t` **min** () throw ()
- static `char16_t` **quiet_NaN** () throw ()
- static `char16_t` **round_error** () throw ()
- static `char16_t` **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.581.1 Detailed Description

template<> struct std::numeric_limits< char16_t >

[numeric_limits<char16_t>](#) specialization.

Definition at line 628 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.582 std::numeric_limits< char32_t > Struct Template Reference

[numeric_limits<char32_t>](#) specialization.

Static Public Member Functions

- static char32_t **denorm_min** () throw ()
- static char32_t **epsilon** () throw ()
- static char32_t **infinity** () throw ()
- static char32_t **lowest** () throw ()
- static char32_t **max** () throw ()
- static char32_t **min** () throw ()
- static char32_t **quiet_NaN** () throw ()
- static char32_t **round_error** () throw ()
- static char32_t **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.582.1 Detailed Description

`template<> struct std::numeric_limits< char32_t >`

[numeric_limits<char32_t>](#) specialization.

Definition at line 686 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.583 `std::numeric_limits< double >` Struct Template Reference

[numeric_limits<double>](#) specialization.

Static Public Member Functions

- static double **denorm_min** () throw ()
- static double **epsilon** () throw ()
- static double **infinity** () throw ()
- static double **lowest** () throw ()
- static double **max** () throw ()
- static double **min** () throw ()
- static double **quiet_NaN** () throw ()
- static double **round_error** () throw ()
- static double **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.583.1 Detailed Description

`template<> struct std::numeric_limits< double >`

[numeric_limits<double>](#) specialization.

Definition at line 1274 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.584 `std::numeric_limits< float >` Struct Template Reference

[numeric_limits<float>](#) specialization.

Static Public Member Functions

- static float **denorm_min** () throw ()
- static float **epsilon** () throw ()
- static float **infinity** () throw ()
- static float **lowest** () throw ()
- static float **max** () throw ()
- static float **min** () throw ()
- static float **quiet_NaN** () throw ()
- static float **round_error** () throw ()
- static float **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.584.1 Detailed Description

template<> struct std::numeric_limits< float >

[numeric_limits<float>](#) specialization.

Definition at line 1209 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.585 std::numeric_limits< int > Struct Template Reference

[numeric_limits<int>](#) specialization.

Static Public Member Functions

- static int **denorm_min** () throw ()
- static int **epsilon** () throw ()
- static int **infinity** () throw ()
- static int **lowest** () throw ()
- static int **max** () throw ()
- static int **min** () throw ()
- static int **quiet_NaN** () throw ()
- static int **round_error** () throw ()
- static int **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.585.1 Detailed Description

`template<> struct std::numeric_limits< int >`

[numeric_limits<int>](#) specialization.

Definition at line 861 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.586 `std::numeric_limits< long >` Struct Template Reference

[numeric_limits<long>](#) specialization.

Static Public Member Functions

- static long **denorm_min** () throw ()
- static long **epsilon** () throw ()
- static long **infinity** () throw ()
- static long **lowest** () throw ()
- static long **max** () throw ()
- static long **min** () throw ()
- static long **quiet_NaN** () throw ()
- static long **round_error** () throw ()
- static long **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.586.1 Detailed Description

`template<> struct std::numeric_limits< long >`

[numeric_limits<long>](#) specialization.

Definition at line 977 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.587 `std::numeric_limits< long double >` Struct Template Reference

[numeric_limits<long double>](#) specialization.

Static Public Member Functions

- static long double **denorm_min** () throw ()
- static long double **epsilon** () throw ()
- static long double **infinity** () throw ()
- static long double **lowest** () throw ()
- static long double **max** () throw ()
- static long double **min** () throw ()
- static long double **quiet_NaN** () throw ()
- static long double **round_error** () throw ()
- static long double **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.587.1 Detailed Description

template<> struct std::numeric_limits< long double >

[numeric_limits<long double>](#) specialization.

Definition at line 1339 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.588 std::numeric_limits< long long > Struct Template Reference

[numeric_limits<long long>](#) specialization.

Static Public Member Functions

- static long long **denorm_min** () throw ()
- static long long **epsilon** () throw ()
- static long long **infinity** () throw ()
- static long long **lowest** () throw ()
- static long long **max** () throw ()
- static long long **min** () throw ()
- static long long **quiet_NaN** () throw ()
- static long long **round_error** () throw ()
- static long long **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.588.1 Detailed Description

`template<> struct std::numeric_limits< long long >`

[numeric_limits<long long>](#) specialization.

Definition at line 1093 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.589 std::numeric_limits< short > Struct Template Reference

[numeric_limits<short>](#) specialization.

Static Public Member Functions

- static short **denorm_min** () throw ()
- static short **epsilon** () throw ()
- static short **infinity** () throw ()
- static short **lowest** () throw ()
- static short **max** () throw ()
- static short **min** () throw ()
- static short **quiet_NaN** () throw ()
- static short **round_error** () throw ()
- static short **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.589.1 Detailed Description

`template<> struct std::numeric_limits< short >`

[numeric_limits<short>](#) specialization.

Definition at line 745 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.590 `std::numeric_limits< signed char >` Struct Template Reference

[numeric_limits<signed char>](#) specialization.

Static Public Member Functions

- static signed char **denorm_min** () throw ()
- static signed char **epsilon** () throw ()
- static signed char **infinity** () throw ()
- static signed char **lowest** () throw ()
- static signed char **max** () throw ()
- static signed char **min** () throw ()
- static signed char **quiet_NaN** () throw ()
- static signed char **round_error** () throw ()
- static signed char **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.590.1 Detailed Description

template<> struct std::numeric_limits< signed char >

[numeric_limits<signed char>](#) specialization.

Definition at line 453 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.591 std::numeric_limits< unsigned char > Struct Template Reference

[numeric_limits<unsigned char>](#) specialization.

Static Public Member Functions

- static unsigned char **denorm_min** () throw ()
- static unsigned char **epsilon** () throw ()
- static unsigned char **infinity** () throw ()
- static unsigned char **lowest** () throw ()
- static unsigned char **max** () throw ()
- static unsigned char **min** () throw ()
- static unsigned char **quiet_NaN** () throw ()
- static unsigned char **round_error** () throw ()
- static unsigned char **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.591.1 Detailed Description

`template<> struct std::numeric_limits< unsigned char >`

[numeric_limits<unsigned char>](#) specialization.

Definition at line 511 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.592 `std::numeric_limits< unsigned int >` Struct Template Reference

[numeric_limits<unsigned int>](#) specialization.

Static Public Member Functions

- static unsigned int **denorm_min** () throw ()
- static unsigned int **epsilon** () throw ()
- static unsigned int **infinity** () throw ()
- static unsigned int **lowest** () throw ()
- static unsigned int **max** () throw ()
- static unsigned int **min** () throw ()
- static unsigned int **quiet_NaN** () throw ()
- static unsigned int **round_error** () throw ()
- static unsigned int **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.592.1 Detailed Description

`template<> struct std::numeric_limits< unsigned int >`

[numeric_limits<unsigned int>](#) specialization.

Definition at line 919 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.593 `std::numeric_limits< unsigned long >` Struct Template Reference

[numeric_limits<unsigned long>](#) specialization.

Static Public Member Functions

- static unsigned long **denorm_min** () throw ()
- static unsigned long **epsilon** () throw ()
- static unsigned long **infinity** () throw ()
- static unsigned long **lowest** () throw ()
- static unsigned long **max** () throw ()
- static unsigned long **min** () throw ()
- static unsigned long **quiet_NaN** () throw ()
- static unsigned long **round_error** () throw ()
- static unsigned long **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.593.1 Detailed Description

template<> struct std::numeric_limits< unsigned long >

[numeric_limits<unsigned long>](#) specialization.

Definition at line 1035 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.594 std::numeric_limits< unsigned long long > Struct Template Reference

[numeric_limits<unsigned long long>](#) specialization.

Static Public Member Functions

- static unsigned long long **denorm_min** () throw ()
- static unsigned long long **epsilon** () throw ()
- static unsigned long long **infinity** () throw ()
- static unsigned long long **lowest** () throw ()
- static unsigned long long **max** () throw ()
- static unsigned long long **min** () throw ()
- static unsigned long long **quiet_NaN** () throw ()
- static unsigned long long **round_error** () throw ()
- static unsigned long long **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.594.1 Detailed Description

`template<> struct std::numeric_limits< unsigned long long >`

[numeric_limits<unsigned long long>](#) specialization.

Definition at line 1151 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

5.595 `std::numeric_limits< unsigned short >` Struct Template Reference

[numeric_limits<unsigned short>](#) specialization.

Static Public Member Functions

- static unsigned short **denorm_min** () throw ()
- static unsigned short **epsilon** () throw ()
- static unsigned short **infinity** () throw ()
- static unsigned short **lowest** () throw ()
- static unsigned short **max** () throw ()
- static unsigned short **min** () throw ()
- static unsigned short **quiet_NaN** () throw ()
- static unsigned short **round_error** () throw ()
- static unsigned short **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**

- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.595.1 Detailed Description

`template<> struct std::numeric_limits< unsigned short >`

[numeric_limits<unsigned short>](#) specialization.

Definition at line 803 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

5.596 `std::numeric_limits< wchar_t >` Struct Template Reference

[numeric_limits<wchar_t>](#) specialization.

Static Public Member Functions

- static `wchar_t` **denorm_min** () throw ()
- static `wchar_t` **epsilon** () throw ()
- static `wchar_t` **infinity** () throw ()
- static `wchar_t` **lowest** () throw ()
- static `wchar_t` **max** () throw ()
- static `wchar_t` **min** () throw ()
- static `wchar_t` **quiet_NaN** () throw ()
- static `wchar_t` **round_error** () throw ()
- static `wchar_t` **signaling_NaN** () throw ()

Static Public Attributes

- static const int **digits**
- static const int **digits10**
- static const [float_denorm_style](#) **has_denorm**
- static const bool **has_denorm_loss**
- static const bool **has_infinity**
- static const bool **has_quiet_NaN**
- static const bool **has_signaling_NaN**
- static const bool **is_bounded**
- static const bool **is_exact**
- static const bool **is_iec559**
- static const bool **is_integer**
- static const bool **is_modulo**
- static const bool **is_signed**
- static const bool **is_specialized**
- static const int **max_digits10**
- static const int **max_exponent**
- static const int **max_exponent10**
- static const int **min_exponent**
- static const int **min_exponent10**
- static const int **radix**
- static const [float_round_style](#) **round_style**
- static const bool **tinyness_before**
- static const bool **traps**

5.596.1 Detailed Description

template<> struct std::numeric_limits< wchar_t >

[numeric_limits<wchar_t>](#) specialization.

Definition at line 569 of file limits.

The documentation for this struct was generated from the following file:

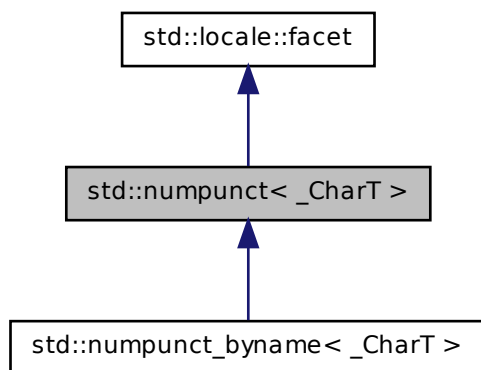
- [limits](#)

5.597 std::numpunct< _CharT > Class Template Reference

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

Inheritance diagram for `std::numpunct<_CharT>`:



Public Types

- `typedef __numpunct_cache<_CharT> __cache_type`
- `typedef _CharT char_type`
- `typedef basic_string<_CharT> string_type`

Public Member Functions

- `numpunct` (size_t __refs=0)
- `numpunct` (__cache_type *__cache, size_t __refs=0)
- `numpunct` (__c_locale __cloc, size_t __refs=0)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~numpunct](#) ()
- template<>
void [_M_initialize_numpunct](#) (__c_locale __cloc)
- template<>
void [_M_initialize_numpunct](#) (__c_locale __cloc)
- void [_M_initialize_numpunct](#) (__c_locale __cloc=0)
- virtual [char_type do_decimal_point](#) () const
- virtual [string_type do_falsename](#) () const
- virtual [string do_grouping](#) () const
- virtual [char_type do_thousands_sep](#) () const
- virtual [string_type do_truename](#) () const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- [__cache_type](#) * [_M_data](#)

Friends

- class [locale::_Impl](#)

5.597.1 Detailed Description

template<typename _CharT> class std::numpunct<_CharT>

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers. The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

Definition at line 1637 of file locale_facets.h.

5.597.2 Member Typedef Documentation

5.597.2.1 template<typename _CharT> typedef _CharT std::numpunct<_CharT>::char_type

Public typedefs.

Reimplemented in [std::numpunct_byname<_CharT>](#).

Definition at line 1643 of file locale_facets.h.

5.597.2.2 template<typename _CharT> typedef basic_string<_CharT> std::numpunct<_CharT>::string_type

Public typedefs.

Reimplemented in [std::numpunct_byname<_CharT>](#).

Definition at line 1644 of file locale_facets.h.

5.597.3 Constructor & Destructor Documentation

5.597.3.1 template<typename _CharT> std::numpunct<_CharT>::numpunct (size_t __refs = 0) [inline, explicit]

Numpunct constructor.

Parameters

refs Refcount to pass to the base class.

Definition at line 1661 of file locale_facets.h.

```
5.597.3.2  template<typename _CharT > std::num_punct< _CharT
            >::num_punct ( __cache_type * __cache, size_t __refs = 0 )
            [inline, explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

Parameters

cache __num_punct_cache object.

refs Refcount to pass to the base class.

Definition at line 1675 of file locale_facets.h.

```
5.597.3.3  template<typename _CharT > std::num_punct< _CharT
            >::num_punct ( __c_locale __cloc, size_t __refs = 0 ) [inline,
            explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

cloc The C locale.

refs Refcount to pass to the base class.

Definition at line 1689 of file locale_facets.h.

```
5.597.3.4  template<typename _CharT > virtual std::num_punct< _CharT
            >::~~num_punct ( ) [protected, virtual]
```

Destructor.

5.597.4 Member Function Documentation

```
5.597.4.1  template<typename _CharT > char_type std::num_punct< _CharT
            >::decimal_point ( ) const [inline]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning returning `num_punct<char_type>::do_decimal_point()`.

Returns

char_type representing a decimal point.

Definition at line 1703 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_decimal_point()`.

5.597.4.2 `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_decimal_point() const [inline, protected, virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1790 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::decimal_point()`.

5.597.4.3 `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_falsename() const [inline, protected, virtual]`

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

`string_type` representing printed form of false.

Definition at line 1841 of file `locale_facets.h`.

Referenced by `std::num_punct<_CharT>::falsename()`.

5.597.4.4 `template<typename _CharT > virtual string std::num_punct<_CharT >::do_grouping() const [inline, protected, virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

[grouping\(\)](#) for details.

Returns

String representing grouping specification.

Definition at line 1815 of file locale_facets.h.

Referenced by `std::num_punct< _CharT >::grouping()`.

5.597.4.5 `template<typename _CharT > virtual char_type std::num_punct<_CharT >::do_thousands_sep() const [inline, protected, virtual]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1802 of file locale_facets.h.

Referenced by `std::num_punct< _CharT >::thousands_sep()`.

5.597.4.6 `template<typename _CharT > virtual string_type std::num_punct<_CharT >::do_truename() const [inline, protected, virtual]`

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

`string_type` representing printed form of true.

Definition at line 1828 of file locale_facets.h.

Referenced by std::num_punct<_CharT>::true_name().

5.597.4.7 template<typename _CharT> string_type std::num_punct<_CharT>::false_name() const [inline]

Return string representation of bool false.

This function returns a string_type containing the text representation for false bool variables. It does so by calling [num_punct<char_type>::do_false_name\(\)](#).

Returns

string_type representing printed form of false.

Definition at line 1773 of file locale_facets.h.

References std::num_punct<_CharT>::do_false_name().

5.597.4.8 template<typename _CharT> string std::num_punct<_CharT>::grouping() const [inline]

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the [grouping\(\)](#) returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling [num_punct<char_type>::do_grouping\(\)](#).

Returns

string representing grouping specification.

Definition at line 1747 of file locale_facets.h.

References std::num_punct<_CharT>::do_grouping().

5.597.4.9 `template<typename _CharT > char_type std::numpunct< _CharT >::thousands_sep () const [inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `numpunct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1716 of file `locale_facets.h`.

References `std::numpunct< _CharT >::do_thousands_sep()`.

5.597.4.10 `template<typename _CharT > string_type std::numpunct< _CharT >::truename () const [inline]`

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `numpunct<char_type>::do_truename()`.

Returns

`string_type` representing printed form of true.

Definition at line 1760 of file `locale_facets.h`.

References `std::numpunct< _CharT >::do_truename()`.

5.597.5 Member Data Documentation

5.597.5.1 `template<typename _CharT > locale::id std::numpunct< _CharT >::id [static]`

Numpunct facet id.

Definition at line 1653 of file `locale_facets.h`.

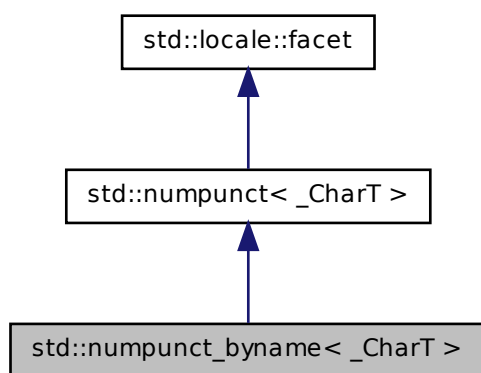
The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.598 std::num_punct_byname< _CharT > Class Template Reference

class [num_punct_byname](#) [22.2.3.2].

Inheritance diagram for std::num_punct_byname< _CharT >:



Public Types

- typedef `__num_punct_cache< _CharT > __cache_type`
- typedef `_CharT char_type`
- typedef `basic_string< _CharT > string_type`

Public Member Functions

- **num_punct_byname** (const char *__s, size_t __refs=0)
- [char_type decimal_point](#) () const
- [string_type falsename](#) () const
- [string grouping](#) () const
- [char_type thousands_sep](#) () const
- [string_type truename](#) () const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- `template<>`
void [_M_initialize_numpunct](#) (__c_locale __cloc)
- `template<>`
void [_M_initialize_numpunct](#) (__c_locale __cloc)
- void [_M_initialize_numpunct](#) (__c_locale __cloc=0)
- virtual [char_type](#) [do_decimal_point](#) () const
- virtual [string_type](#) [do_falsename](#) () const
- virtual [string](#) [do_grouping](#) () const
- virtual [char_type](#) [do_thousands_sep](#) () const
- virtual [string_type](#) [do_truename](#) () const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static _GLIBCXX_CONST const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __cache_type * [_M_data](#)

Friends

- class [locale::_Impl](#)

5.598.1 Detailed Description

`template<typename _CharT> class std::numpunct_byname< _CharT >`

class [numpunct_byname](#) [22.2.3.2].

Definition at line 1870 of file `locale_facets.h`.

5.598.2 Member Typedef Documentation

5.598.2.1 `template<typename _CharT > typedef _CharT std::num_punct_byname< _CharT >::char_type`

Public typedefs.

Reimplemented from [std::num_punct< _CharT >](#).

Definition at line 1873 of file locale_facets.h.

5.598.2.2 `template<typename _CharT > typedef basic_string<_CharT> std::num_punct_byname< _CharT >::string_type`

Public typedefs.

Reimplemented from [std::num_punct< _CharT >](#).

Definition at line 1874 of file locale_facets.h.

5.598.3 Member Function Documentation

5.598.3.1 `template<typename _CharT > char_type std::num_punct< _CharT >::decimal_point() const [inline, inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning [num_punct<char_type>::do_decimal_point\(\)](#).

Returns

char_type representing a decimal point.

Definition at line 1703 of file locale_facets.h.

References [std::num_punct< _CharT >::do_decimal_point\(\)](#).

5.598.3.2 `template<typename _CharT > virtual char_type std::num_punct< _CharT >::do_decimal_point() const [inline, protected, virtual, inherited]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1790 of file locale_facets.h.

Referenced by `std::num_punct<_CharT>::decimal_point()`.

5.598.3.3 template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_falsename() const [inline, protected, virtual, inherited]

Return string representation of bool false.

Returns a *string_type* containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Definition at line 1841 of file locale_facets.h.

Referenced by `std::num_punct<_CharT>::falsename()`.

5.598.3.4 template<typename _CharT> virtual string std::num_punct<_CharT>::do_grouping() const [inline, protected, virtual, inherited]

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

[grouping\(\)](#) for details.

Returns

String representing grouping specification.

Definition at line 1815 of file locale_facets.h.

Referenced by `std::num_punct<_CharT>::grouping()`.

5.598.3.5 `template<typename _CharT > virtual char_type std::num_punct<_CharT>::do_thousands_sep() const [inline, protected, virtual, inherited]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1802 of file `locale_facets.h`.

Referenced by `std::num_punct< _CharT >::thousands_sep()`.

5.598.3.6 `template<typename _CharT > virtual string_type std::num_punct<_CharT>::do_truename() const [inline, protected, virtual, inherited]`

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

`string_type` representing printed form of true.

Definition at line 1828 of file `locale_facets.h`.

Referenced by `std::num_punct< _CharT >::truename()`.

5.598.3.7 `template<typename _CharT > string_type std::num_punct<_CharT>::falsename() const [inline, inherited]`

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

Returns

`string_type` representing printed form of false.

Definition at line 1773 of file `locale_facets.h`.

References `std::num_punct< _CharT >::do_falsename()`.

5.598.3.8 `template<typename _CharT > string std::num_punct<_CharT>::grouping() const [inline, inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

Definition at line 1747 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_grouping()`.

5.598.3.9 `template<typename _CharT > char_type std::num_punct<_CharT>::thousands_sep() const [inline, inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `num_punct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1716 of file `locale_facets.h`.

References `std::num_punct<_CharT>::do_thousands_sep()`.

5.598.3.10 `template<typename _CharT > string_type std::num_punct<_CharT>::true_name() const [inline, inherited]`

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `numpunct<char_type>::do_truename()`.

Returns

`string_type` representing printed form of true.

Definition at line 1760 of file `locale_facets.h`.

References `std::numpunct<_CharT>::do_truename()`.

5.598.4 Member Data Documentation

5.598.4.1 `template<typename _CharT > locale::id std::numpunct<_CharT>::id [static, inherited]`

Numpunct facet id.

Definition at line 1653 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

5.599 std::once_flag Struct Reference

[once_flag](#)

Public Member Functions

- `once_flag` (const [once_flag](#) &)
- `once_flag` & `operator=` (const [once_flag](#) &)

Friends

- `template<typename _Callable, typename... _Args>`
void `call_once` ([once_flag](#) &__once, `_Callable` &&__f, `_Args` &&...__args)

5.599.1 Detailed Description

[once_flag](#)

Definition at line 675 of file `mutex`.

5.599.2 Friends And Related Function Documentation

5.599.2.1 `template<typename _Callable, typename... _Args> void call_once (`
`once_flag & __once, _Callable && __f, _Args &&... __args)`
`[friend]`

`call_once`

Definition at line 721 of file `mutex`.

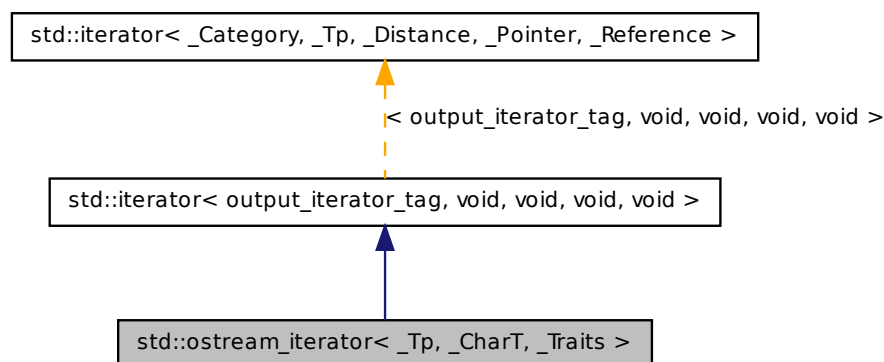
The documentation for this struct was generated from the following file:

- [mutex](#)

5.600 `std::ostream_iterator< _Tp, _CharT, _Traits >` Class Template Reference

Provides output iterator semantics for streams.

Inheritance diagram for `std::ostream_iterator< _Tp, _CharT, _Traits >`:



Public Types

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) [iterator_category](#)

5.600 `std::ostream_iterator<_Tp, _CharT, _Traits>` Class Template Reference 2885

- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)

- typedef [_CharT](#) [char_type](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [basic_ostream<_CharT, _Traits>](#) [ostream_type](#)

Public Member Functions

- [ostream_iterator](#) ([ostream_type](#) &__s)
- [ostream_iterator](#) ([ostream_type](#) &__s, const [_CharT](#) *__c)
- [ostream_iterator](#) (const [ostream_iterator](#) &__obj)
- [ostream_iterator](#) & [operator*](#) ()
- [ostream_iterator](#) & [operator++](#) (int)
- [ostream_iterator](#) & [operator++](#) ()
- [ostream_iterator](#) & [operator=](#) (const [_Tp](#) &__value)

5.600.1 Detailed Description

`template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> class std::ostream_iterator<_Tp, _CharT, _Traits>`

Provides output iterator semantics for streams. This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

Parameters

Tp The type to write to the ostream.

CharT The ostream `char_type`.

Traits The ostream [char_traits](#).

Definition at line 152 of file `stream_iterator.h`.

5.600.2 Member Typedef Documentation

5.600.2.1 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _CharT std::ostream_iterator<_Tp, _CharT, _Traits>::char_type`

Public typedef.

Definition at line 158 of file `stream_iterator.h`.

5.600.2.2 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 124 of file `stl_iterator_base_types.h`.

5.600.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 120 of file `stl_iterator_base_types.h`.

5.600.2.4 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type`

Public typedef.

Definition at line 160 of file `stream_iterator.h`.

5.600.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 126 of file `stl_iterator_base_types.h`.

5.600.2.6 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 128 of file `stl_iterator_base_types.h`.

5.600.2.7 `template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type`

Public typedef.

Definition at line 159 of file `stream_iterator.h`.

5.600.2.8 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file stl_iterator_base_types.h.

5.600.3 Constructor & Destructor Documentation

5.600.3.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (ostream_type & __s) [inline]`

Construct from an ostream.

Definition at line 169 of file stream_iterator.h.

5.600.3.2 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (ostream_type & __s, const _CharT * __c) [inline]`

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

Parameters

s Underlying ostream to write to.

c CharT delimiter string to insert.

Definition at line 181 of file stream_iterator.h.

5.600.3.3 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (const ostream_iterator< _Tp, _CharT, _Traits > & __obj) [inline]`

Copy constructor.

Definition at line 185 of file stream_iterator.h.

5.600.4 Member Function Documentation

5.600.4.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> ostream_iterator& std::ostream_iterator<_Tp, _CharT, _Traits>::operator= (const _Tp & __value) [inline]`

Writes *value* to underlying ostream using operator<<. If /// constructed with delimiter string, writes delimiter to ostream.

Definition at line 191 of file stream_iterator.h.

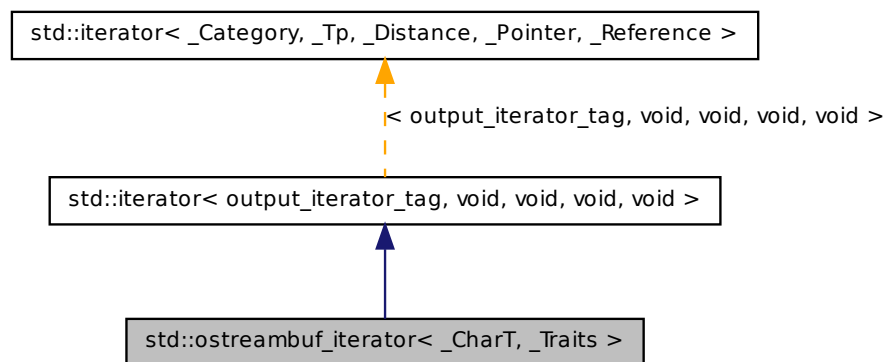
The documentation for this class was generated from the following file:

- [stream_iterator.h](#)

5.601 `std::ostreambuf_iterator< _CharT, _Traits >` Class Template Reference

Provides output iterator semantics for streambufs.

Inheritance diagram for `std::ostreambuf_iterator< _CharT, _Traits >`:



Public Types

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) [iterator_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)

- typedef [_CharT](#) [char_type](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [basic_streambuf< _CharT, _Traits >](#) [streambuf_type](#)
- typedef [basic_ostream< _CharT, _Traits >](#) [ostream_type](#)

Public Member Functions

- [ostreambuf_iterator](#) ([ostream_type](#) &__s) throw ()
- [ostreambuf_iterator](#) ([streambuf_type](#) *__s) throw ()
- [ostreambuf_iterator](#) & [_M_put](#) (const [_CharT](#) *__ws, [streamsize](#) __len)
- bool [failed](#) () const throw ()
- [ostreambuf_iterator](#) & [operator*](#) ()
- [ostreambuf_iterator](#) & [operator++](#) ()
- [ostreambuf_iterator](#) & [operator++](#) (int)
- [ostreambuf_iterator](#) & [operator=](#) ([_CharT](#) __c)

Friends

- `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 > >::__type` **copy** ([istreambuf_iterator< _CharT2 >](#),
[istreambuf_iterator< _CharT2 >](#), [ostreambuf_iterator< _CharT2 >](#))

5.601.1 Detailed Description

`template<typename _CharT, typename _Traits> class std::ostreambuf_iterator< _CharT, _Traits >`

Provides output iterator semantics for streambufs.

Definition at line 204 of file `streambuf_iterator.h`.

5.601.2 Member Typedef Documentation

5.601.2.1 `template<typename _CharT, typename _Traits> typedef _CharT std::ostreambuf_iterator< _CharT, _Traits>::char_type`

Public typedefs.

Definition at line 211 of file `streambuf_iterator.h`.

5.601.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::difference_type [inherited]`

Distance between iterators is represented as this type.

Definition at line 124 of file `stl_iterator_base_types.h`.

5.601.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void>::iterator_category [inherited]`

One of the [tag types](#).

Definition at line 120 of file `stl_iterator_base_types.h`.

5.601.2.4 `template<typename _CharT, typename _Traits> typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits>::ostream_type`

Public typedefs.

Definition at line 214 of file `streambuf_iterator.h`.

5.601.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 126 of file `stl_iterator_base_types.h`.

5.601.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 128 of file `stl_iterator_base_types.h`.

5.601 std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference

5.601.2.7 `template<typename _CharT , typename _Traits > typedef
basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator<
_CharT, _Traits >::streambuf_type`

Public typedefs.

Definition at line 213 of file streambuf_iterator.h.

5.601.2.8 `template<typename _CharT , typename _Traits > typedef _Traits
std::ostreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 212 of file streambuf_iterator.h.

5.601.2.9 `typedef void std::iterator< output_iterator_tag , void , void , void ,
void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file stl_iterator_base_types.h.

5.601.3 Constructor & Destructor Documentation

5.601.3.1 `template<typename _CharT , typename _Traits >
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
ostream_type & __s) throw () [inline]`

Construct output iterator from ostream.

Definition at line 229 of file streambuf_iterator.h.

5.601.3.2 `template<typename _CharT , typename _Traits >
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
streambuf_type * __s) throw () [inline]`

Construct output iterator from streambuf.

Definition at line 233 of file streambuf_iterator.h.

5.601.4 Member Function Documentation

5.601.4.1 `template<typename _CharT, typename _Traits > bool
std::ostreambuf_iterator< _CharT, _Traits >::failed () const throw
() [inline]`

Return true if previous [operator=\(\)](#) failed.

Definition at line 263 of file `streambuf_iterator.h`.

5.601.4.2 `template<typename _CharT, typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits
>::operator* () [inline]`

Return `*this`.

Definition at line 248 of file `streambuf_iterator.h`.

5.601.4.3 `template<typename _CharT, typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits
>::operator++ () [inline]`

Return `*this`.

Definition at line 258 of file `streambuf_iterator.h`.

5.601.4.4 `template<typename _CharT, typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits
>::operator++ (int) [inline]`

Return `*this`.

Definition at line 253 of file `streambuf_iterator.h`.

5.601.4.5 `template<typename _CharT, typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits
>::operator= (_CharT __c) [inline]`

Write character to streambuf. Calls [streambuf.sputc\(\)](#).

Definition at line 238 of file `streambuf_iterator.h`.

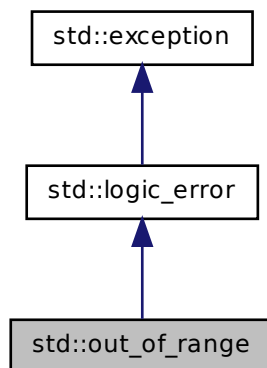
References `std::basic_streambuf< _CharT, _Traits >::sputc()`.

The documentation for this class was generated from the following file:

- [streambuf_iterator.h](#)

5.602 std::out_of_range Class Reference

Inheritance diagram for std::out_of_range:



Public Member Functions

- **out_of_range** (const [string](#) &__arg)
- virtual const char * **what** () const throw ()

5.602.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in [basic_string](#)).

Definition at line 96 of file stdexcept.

5.602.2 Member Function Documentation

5.602.2.1 virtual const char* std::logic_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.603 std::output_iterator_tag Struct Reference

Marking output iterators.

5.603.1 Detailed Description

Marking output iterators.

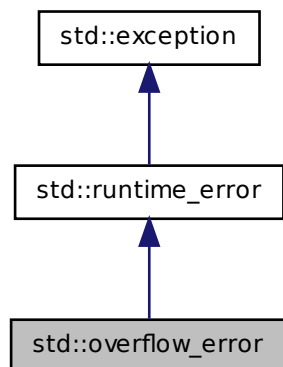
Definition at line 91 of file [stl_iterator_base_types.h](#).

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.604 `std::overflow_error` Class Reference

Inheritance diagram for `std::overflow_error`:



Public Member Functions

- **`overflow_error`** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.604.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 133 of file `stdexcept`.

5.604.2 Member Function Documentation

5.604.2.1 `virtual const char* std::runtime_error::what () const throw ()` [[virtual](#), [inherited](#)]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.605 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

Partial specialization of `owner_less` for [shared_ptr](#).

Inherits `std::_Sp_owner_less< shared_ptr< _Tp >, weak_ptr< _Tp > >`.

Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _Tp &__lhs, const _Tp &__rhs) const`
- `bool operator() (const _Tp1 &__lhs, const _Tp &__rhs) const`
- `bool operator() (const _Tp &__lhs, const _Tp1 &__rhs) const`

5.605.1 Detailed Description

`template<typename _Tp> struct std::owner_less< shared_ptr< _Tp > >`

Partial specialization of `owner_less` for [shared_ptr](#).

Definition at line 452 of file `shared_ptr.h`.

5.605.2 Member Typedef Documentation

5.605.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.605.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type`
`[inherited]`

type of the return type

Definition at line 118 of file stl_function.h.

5.605.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool`
`>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

5.606 std::owner_less< weak_ptr< _Tp > > Struct Template Reference

Partial specialization of owner_less for [weak_ptr](#).

Inherits std::_Sp_owner_less< weak_ptr< _Tp >, shared_ptr< _Tp > >.

Public Types

- typedef _Tp [first_argument_type](#)
- typedef bool [result_type](#)
- typedef _Tp [second_argument_type](#)

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const

5.606.1 Detailed Description

`template<typename _Tp> struct std::owner_less< weak_ptr< _Tp > >`

Partial specialization of owner_less for [weak_ptr](#).

Definition at line 458 of file shared_ptr.h.

5.606.2 Member Typedef Documentation

5.606.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.606.2.2 `typedef bool std::binary_function< _Tp , _Tp , bool >::result_type
[inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.606.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , bool
>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

5.607 `std::packaged_task< _Res(_ArgTypes...)>` Class Template Reference

`packaged_task`

Public Types

- `typedef _Res result_type`

Public Member Functions

- `template<typename _Fn >
packaged_task (const _Fn &__fn)`
- `packaged_task (_Res(*__fn)(_ArgTypes...))`
- `packaged_task (packaged_task &&__other)`

- `template<typename _Fn, typename _Allocator >`
packaged_task (`allocator_arg_t` __tag, const _Allocator &__a, _Fn __fn)
- `template<typename _Fn >`
packaged_task (_Fn &&__fn)
- **packaged_task** (packaged_task &)
- `future< _Res >` **get_future** ()
- **operator bool** () const
- void **operator**() (_ArgTypes...__args)
- packaged_task & **operator=** (packaged_task &)
- packaged_task & **operator=** (packaged_task &&__other)
- void **reset** ()
- void **swap** (packaged_task &__other)

5.607.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes> class std::packaged_task<
    _Res(_ArgTypes...)>
```

packaged_task

Definition at line 1164 of file future.

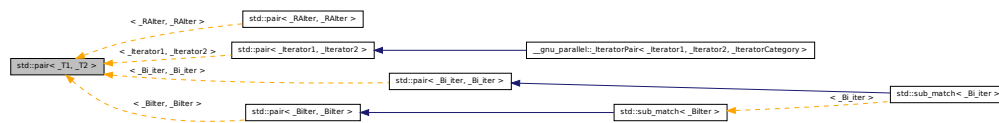
The documentation for this class was generated from the following file:

- future

5.608 std::pair< _T1, _T2 > Struct Template Reference

pair holds two objects of arbitrary type.

Inheritance diagram for `std::pair<_T1, _T2>`:



Public Types

- typedef _T1 first_type

- typedef `_T2` [second_type](#)

Public Member Functions

- [pair](#) ()
- [pair](#) (const `_T1` &__a, const `_T2` &__b)
- template<class `_U1` , class = typename std::enable_if<std::is_convertible<`_U1`, `_T1`>::value>::type>
pair (`_U1` &&__x, const `_T2` &__y)
- template<class `_U1` , class `_U2` >
[pair](#) (const [pair](#)< `_U1`, `_U2` > &__p)
- template<class `_U1` , class `_U2` >
pair ([pair](#)< `_U1`, `_U2` > &&__p)
- template<class `_U2` , class = typename std::enable_if<std::is_convertible<`_U2`, `_T2`>::value>::type>
pair (const `_T1` &__x, `_U2` &&__y)
- **pair** (const [pair](#) &)
- template<class... `_Args1`, class... `_Args2`>
pair (piecewise_construct_t, [tuple](#)< `_Args1...` > __first_args, [tuple](#)< `_Args2...` > __second_args)
- template<class `_U1` , class `_U2` , class = typename std::enable_if<std::is_convertible<`_U1`, `_T1`>::value && std::is_convertible<`_U2`, `_T2`>::value>::type>
pair (`_U1` &&__x, `_U2` &&__y)
- [pair](#) & **operator=** ([pair](#) &&__p)
- template<class `_U1` , class `_U2` >
[pair](#) & **operator=** (const [pair](#)< `_U1`, `_U2` > &__p)
- template<class `_U1` , class `_U2` >
[pair](#) & **operator=** ([pair](#)< `_U1`, `_U2` > &&__p)
- void **swap** ([pair](#) &__p)

Public Attributes

- `_T1` [first](#)
- `_T2` [second](#)

5.608.1 Detailed Description

template<class `_T1`, class `_T2`> struct std::pair< `_T1`, `_T2` >

pair holds two objects of arbitrary type.

Definition at line 84 of file stl_pair.h.

5.608.2 Member Typedef Documentation

5.608.2.1 `template<class _T1, class _T2> typedef _T1 std::pair< _T1, _T2 >::first_type`

`first_type` is the first bound type

Definition at line 86 of file `stl_pair.h`.

5.608.2.2 `template<class _T1, class _T2> typedef _T2 std::pair< _T1, _T2 >::second_type`

`second_type` is the second bound type

Definition at line 87 of file `stl_pair.h`.

5.608.3 Constructor & Destructor Documentation

5.608.3.1 `template<class _T1, class _T2> std::pair< _T1, _T2 >::pair ()
[inline]`

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 96 of file `stl_pair.h`.

5.608.3.2 `template<class _T1, class _T2> std::pair< _T1, _T2 >::pair (const
_T1 & __a, const _T2 & __b) [inline]`

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 100 of file `stl_pair.h`.

5.608.3.3 `template<class _T1, class _T2> template<class _U1, class _U2 >
std::pair< _T1, _T2 >::pair (const pair< _U1, _U2 > & __p)
[inline]`

There is also a templated copy ctor for the `pair` class itself.

Definition at line 136 of file `stl_pair.h`.

5.608.4 Member Data Documentation

5.608.4.1 `template<class _T1, class _T2> _T1 std::pair< _T1, _T2 >::first`

`first` is a copy of the first object

Definition at line 89 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer()`, `std::set< _StateIdT >::insert()`, and `std::operator==()`.

5.608.4.2 `template<class _T1, class _T2> _T2 std::pair< _T1, _T2 >::second`

`second` is a copy of the second object

Definition at line 90 of file `stl_pair.h`.

Referenced by `std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer()`, `std::set< _StateIdT >::insert()`, and `std::operator==()`.

The documentation for this struct was generated from the following files:

- [stl_pair.h](#)
- [tuple](#)

5.609 `std::piecewise_constant_distribution< _RealType >` Class Template Reference

A [piecewise_constant_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW > piecewise_constant_distribution (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)`

- `template<typename _Func>`
piecewise_constant_distribution (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- **piecewise_constant_distribution** (const [param_type](#) &__p)
- `template<typename _Func>`
piecewise_constant_distribution ([initializer_list](#)<_RealType> __bl, _Func __fw)
- `std::vector< double >` [densities](#) () const
- `std::vector<_RealType>` [intervals](#) () const
- [result_type](#) [max](#) () const
- [result_type](#) [min](#) () const
- `template<typename _UniformRandomNumberGenerator>`
[result_type](#) [operator\(\)](#) (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- `template<typename _UniformRandomNumberGenerator>`
[result_type](#) [operator\(\)](#) (_UniformRandomNumberGenerator &__urng)
- [param_type](#) [param](#) () const
- void [param](#) (const [param_type](#) &__param)
- void [reset](#) ()

Friends

- `template<typename _RealType1, typename _CharT, typename _Traits>`
[std::basic_ostream](#)<_CharT, _Traits> & [operator<<](#) ([std::basic_ostream](#)<_CharT, _Traits> &, const [std::piecewise_constant_distribution](#)<_RealType1> &)
- `template<typename _RealType1, typename _CharT, typename _Traits>`
[std::basic_istream](#)<_CharT, _Traits> & [operator>>](#) ([std::basic_istream](#)<_CharT, _Traits> &, [std::piecewise_constant_distribution](#)<_RealType1> &)

5.609.1 Detailed Description

`template<typename _RealType = double> class std::piecewise_constant_distribution<_RealType>`

A [piecewise_constant_distribution](#) random number distribution. The formula for the piecewise constant probability mass function is

Definition at line 4877 of file random.h.

5.609.2 Member Typedef Documentation

5.609.2.1 `template<typename _RealType = double> typedef _RealType
std::piecewise_constant_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 4884 of file random.h.

5.609.3 Member Function Documentation

5.609.3.1 `template<typename _RealType = double> std::vector<double>
std::piecewise_constant_distribution< _RealType >::densities ()
const [inline]`

Returns a vector of the probability densities.

Definition at line 4976 of file random.h.

5.609.3.2 `template<typename _RealType = double> std::vector<_RealType>
std::piecewise_constant_distribution< _RealType >::intervals ()
const [inline]`

Returns a vector of the intervals.

Definition at line 4969 of file random.h.

5.609.3.3 `template<typename _RealType = double> result_type
std::piecewise_constant_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5005 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`.

5.609.3.4 `template<typename _RealType = double> result_type
std::piecewise_constant_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4998 of file random.h.

References `std::vector< _Tp, _Alloc >::front()`.

5.609.3.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::piecewise_constant_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 5013 of file random.h.

References `std::piecewise_constant_distribution< _RealType >::operator()`, and
`std::piecewise_constant_distribution< _RealType >::param()`.

Referenced by `std::piecewise_constant_distribution< _RealType >::operator()`.

5.609.3.6 `template<typename _RealType = double> void
std::piecewise_constant_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 4991 of file random.h.

5.609.3.7 `template<typename _RealType = double> param_type
std::piecewise_constant_distribution< _RealType >::param ()
const [inline]`

Returns the parameter set of the distribution.

Definition at line 4983 of file random.h.

Referenced by `std::piecewise_constant_distribution< _RealType >::operator()`, and
`std::operator==()`.

5.609.3.8 `template<typename _RealType = double> void
std::piecewise_constant_distribution< _RealType >::reset ()
[inline]`

Resets the distribution state.

Definition at line 4962 of file random.h.

5.609.4 Friends And Related Function Documentation

5.609.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::piecewise_constant_distribution< _RealType1 > &)
[friend]`

Inserts a `piecewise_constan_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `piecewise_constan_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.609.4.2 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits > std::basic_
istream<_CharT, _Traits>& operator>> (std::basic_istream<
_CharT, _Traits > & , std::piecewise_constant_distribution<
_RealType1 > &) [friend]`

Extracts a `piecewise_constan_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `piecewise_constan_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.610 `std::piecewise_constant_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `piecewise_constant_distribution<_RealType>` `distribution_type`

Public Member Functions

- template<typename `_InputIteratorB`, typename `_InputIteratorW`>
`param_type` (`_InputIteratorB` `__bfirst`, `_InputIteratorB` `__bend`, `_InputIteratorW` `__wbegin`)
- template<typename `_Func`>
`param_type` (`size_t` `__nw`, `_RealType` `__xmin`, `_RealType` `__xmax`, `_Func` `__fw`)
- template<typename `_Func`>
`param_type` (`initializer_list<_RealType>` `__bi`, `_Func` `__fw`)
- `std::vector<double>` `densities` () const
- `std::vector<_RealType>` `intervals` () const

Friends

- bool `operator==` (const `param_type` &`__p1`, const `param_type` &`__p2`)
- class `piecewise_constant_distribution<_RealType>`

5.610.1 Detailed Description

`template<typename _RealType = double> struct std::piecewise_constant_distribution<_RealType>::param_type`

Parameter type.

Definition at line 4886 of file `random.h`.

The documentation for this struct was generated from the following files:

- `random.h`
- `random.tcc`

5.611 `std::piecewise_linear_distribution< _RealType >` Class Template Reference

A [piecewise_linear_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`
piecewise_linear_distribution (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`
piecewise_linear_distribution (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **piecewise_linear_distribution** (`const param_type &__p`)
- `template<typename _Func >`
piecewise_linear_distribution (`initializer_list< _RealType > __bl, _Func __fw`)
- `std::vector< double > densities () const`
- `std::vector< _RealType > intervals () const`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream<_CharT, _Traits> & operator<< (std::basic_ostream<_CharT, _Traits> &, const std::piecewise_linear_distribution<_RealType1> &)`
- `template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &, std::piecewise_linear_distribution<_RealType1> &)`

5.611.1 Detailed Description

`template<typename _RealType = double> class std::piecewise_linear_distribution<_RealType>`

A [piecewise_linear_distribution](#) random number distribution. The formula for the piecewise linear probability mass function is

Definition at line 5085 of file random.h.

5.611.2 Member Typedef Documentation

5.611.2.1 `template<typename _RealType = double> typedef _RealType
std::piecewise_linear_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 5092 of file random.h.

5.611.3 Member Function Documentation

5.611.3.1 `template<typename _RealType = double> std::vector<double>
std::piecewise_linear_distribution<_RealType>::densities () const
[inline]`

Return a vector of the probability densities of the distribution.

Definition at line 5187 of file random.h.

5.611.3.2 `template<typename _RealType = double> std::vector<_RealType>
std::piecewise_linear_distribution<_RealType>::intervals () const
[inline]`

Return the intervals of the distribution.

Definition at line 5179 of file random.h.

5.611.3.3 `template<typename _RealType = double> result_type
std::piecewise_linear_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5216 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`.

5.611.3.4 `template<typename _RealType = double> result_type
std::piecewise_linear_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5209 of file random.h.

References `std::vector< _Tp, _Alloc >::front()`.

5.611.3.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::piecewise_linear_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 5224 of file random.h.

References `std::piecewise_linear_distribution< _RealType >::operator()()`, and `std::piecewise_linear_distribution< _RealType >::param()`.

Referenced by `std::piecewise_linear_distribution< _RealType >::operator()()`.

5.611.3.6 `template<typename _RealType = double> void
std::piecewise_linear_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 5202 of file random.h.

5.611 std::piecewise_linear_distribution< _RealType > Class Template Reference

2911

5.611.3.7 `template<typename _RealType = double> param_type
std::piecewise_linear_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 5194 of file random.h.

Referenced by `std::piecewise_linear_distribution< _RealType >::operator()()`, and `std::operator==()`.

5.611.3.8 `template<typename _RealType = double> void
std::piecewise_linear_distribution< _RealType >::reset ()
[inline]`

Resets the distribution state.

Definition at line 5172 of file random.h.

5.611.4 Friends And Related Function Documentation

5.611.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::piecewise_linear_distribution< _RealType1 > &) [friend]`

Inserts a `piecewise_linear_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `piecewise_linear_distribution` random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.611.4.2 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>>
(std::basic_istream< _CharT, _Traits > & ,
std::piecewise_linear_distribution< _RealType1 > &) [friend]`

Extracts a `piecewise_linear_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `piecewise_linear_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.612 `std::piecewise_linear_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef `piecewise_linear_distribution< _RealType >` `distribution_type`

Public Member Functions

- `template<typename _InputIteratorB , typename _InputIteratorW >
param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW
__wbegin)`
- `template<typename _Func >
param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __-
fw)`
- `template<typename _Func >
param_type (initializer_list< _RealType > __bl, _Func __fw)`
- `std::vector< double > densities () const`
- `std::vector< _RealType > intervals () const`

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)
- class **piecewise_linear_distribution**< _RealType >

5.612.1 Detailed Description

template<typename _RealType = double> struct std::piecewise_linear_distribution< _RealType >::param_type

Parameter type.

Definition at line 5094 of file random.h.

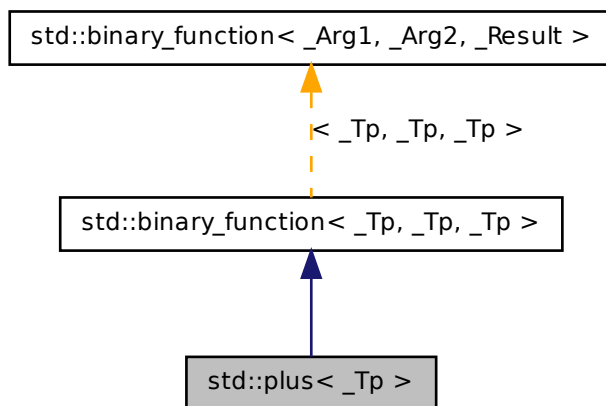
The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.613 std::plus< _Tp > Struct Template Reference

One of the [math functors](#).

Inheritance diagram for std::plus< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

5.613.1 Detailed Description

`template<typename _Tp> struct std::plus< _Tp >`

One of the [math functors](#).

Definition at line 135 of file `stl_function.h`.

5.613.2 Member Typedef Documentation

5.613.2.1 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type` [`inherited`]

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.613.2.2 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type` [`inherited`]

type of the return type

Definition at line 118 of file `stl_function.h`.

5.613.2.3 `typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type` [`inherited`]

the type of the second argument

Definition at line 117 of file `stl_function.h`.

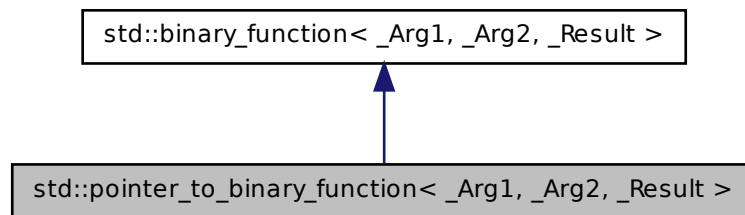
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.614 `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >` Class Template Reference

One of the [adaptors for function pointers](#).

Inheritance diagram for `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`:



Public Types

- typedef `_Arg1` [first_argument_type](#)
- typedef `_Result` [result_type](#)
- typedef `_Arg2` [second_argument_type](#)

Public Member Functions

- `pointer_to_binary_function` (`_Result`(*__x)(`_Arg1`, `_Arg2`))
- `_Result` `operator()` (`_Arg1` __x, `_Arg2` __y) const

Protected Attributes

- `_Result`(* `_M_ptr`)(`_Arg1`, `_Arg2`)

5.614.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result> class
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 442 of file `stl_function.h`.

5.614.2 Member Typedef Documentation

5.614.2.1 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result`
`>::first_argument_type [inherited]`

the type of the first argument /// (no surprises here)

Definition at line 114 of file `stl_function.h`.

5.614.2.2 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Result std::binary_function< _Arg1, _Arg2, _Result`
`>::result_type [inherited]`

type of the return type

Definition at line 118 of file `stl_function.h`.

5.614.2.3 `template<typename _Arg1, typename _Arg2, typename _Result>`
`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result`
`>::second_argument_type [inherited]`

the type of the second argument

Definition at line 117 of file `stl_function.h`.

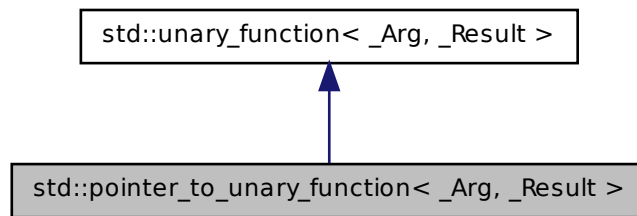
The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.615 `std::pointer_to_unary_function< _Arg, _Result >` Class Template Reference

One of the [adaptors for function pointers](#).

Inheritance diagram for std::pointer_to_unary_function< _Arg, _Result >:



Public Types

- typedef _Arg [argument_type](#)
- typedef _Result [result_type](#)

Public Member Functions

- **pointer_to_unary_function** (_Result(*__x)(_Arg))
- _Result **operator()** (_Arg __x) const

Protected Attributes

- _Result(* **M_ptr**)(_Arg)

5.615.1 Detailed Description

template<typename _Arg, typename _Result> class std::pointer_to_unary_function< _Arg, _Result >

One of the [adaptors for function pointers](#).

Definition at line 417 of file stl_function.h.

5.615.2 Member Typedef Documentation

5.615.2.1 `template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type
[inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.615.2.2 `template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type [inherited]`

`result_type` is the return type

Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.616 `std::poisson_distribution< _IntType >` Class Template Reference

A discrete Poisson random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- `poisson_distribution` (double `__mean=1.0`)
- `poisson_distribution` (const [param_type](#) &`__p`)
- [result_type](#) `max` () const
- double `mean` () const
- [result_type](#) `min` () const

- template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::poisson_distribution< _IntType1 > &)
- template<typename _IntType1 >
bool operator== (const std::poisson_distribution< _IntType1 > &__d1, const std::poisson_distribution< _IntType1 > &__d2)
- template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::poisson_distribution< _IntType1 > &)

5.616.1 Detailed Description

template<typename _IntType = int> class std::poisson_distribution< _IntType >

A discrete Poisson random number distribution. The formula for the Poisson probability density function is $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$ where μ is the parameter of the distribution.

Definition at line 3976 of file random.h.

5.616.2 Member Typedef Documentation

5.616.2.1 template<typename _IntType = int> typedef _IntType
std::poisson_distribution< _IntType >::result_type

The type of the range of the distribution.

Definition at line 3983 of file random.h.

5.616.3 Member Function Documentation

5.616.3.1 `template<typename _IntType = int> result_type
std::poisson_distribution< _IntType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4070 of file random.h.

Referenced by `std::poisson_distribution< _IntType >::operator()()`.

5.616.3.2 `template<typename _IntType = int> double
std::poisson_distribution< _IntType >::mean () const [inline]`

Returns the distribution parameter `mean`.

Definition at line 4041 of file random.h.

5.616.3.3 `template<typename _IntType = int> result_type
std::poisson_distribution< _IntType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4063 of file random.h.

5.616.3.4 `template<typename _IntType > template<typename
_UniformRandomNumberGenerator > poisson_distribution<
_IntType >::result_type std::poisson_distribution< _IntType
>::operator() (_UniformRandomNumberGenerator & __urng,
const param_type & __param)`

A rejection algorithm when `mean >= 12` and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

Definition at line 1195 of file random.tcc.

References `std::abs()`, `std::log()`, and `std::poisson_distribution< _IntType >::max()`.

5.616.3.5 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::poisson_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4078 of file random.h.

References std::poisson_distribution< _IntType >::operator>(), and std::poisson_distribution< _IntType >::param().

Referenced by std::poisson_distribution< _IntType >::operator().

5.616.3.6 `template<typename _IntType = int> param_type
std::poisson_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4048 of file random.h.

Referenced by std::poisson_distribution< _IntType >::operator().

5.616.3.7 `template<typename _IntType = int> void std::poisson_distribution<
_IntType >::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

Definition at line 4056 of file random.h.

5.616.3.8 `template<typename _IntType = int> void std::poisson_distribution<
_IntType >::reset () [inline]`

Resets the distribution state.

Definition at line 4034 of file random.h.

References std::normal_distribution< _RealType >::reset().

5.616.4 Friends And Related Function Documentation

5.616.4.1 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_ostream<_CharT,
_Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & ,
const std::poisson_distribution< _IntType1 > &) [friend]`

Inserts a poisson_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A poisson_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.616.4.2 `template<typename _IntType = int> template<typename _IntType1
> bool operator==(const std::poisson_distribution< _IntType1 >
& __d1, const std::poisson_distribution< _IntType1 > & __d2)
[friend]`

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4093 of file random.h.

5.616.4.3 `template<typename _IntType = int> template<typename _IntType1 ,
typename _CharT , typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> (std::basic_istream< _CharT, _Traits > & ,
std::poisson_distribution< _IntType1 > &) [friend]`

Extracts a poisson_distribution random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A poisson_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

5.617 `std::poisson_distribution<_IntType>::param_type` Struct Reference 2923

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.617 `std::poisson_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `poisson_distribution<_IntType>` `distribution_type`

Public Member Functions

- `param_type` (double `__mean`=1.0)
- double `mean` () const

Friends

- bool `operator==` (const `param_type` &`__p1`, const `param_type` &`__p2`)
- class `poisson_distribution<_IntType>`

5.617.1 Detailed Description

`template<typename _IntType = int> struct std::poisson_distribution<_IntType>::param_type`

Parameter type.

Definition at line 3985 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.618 `std::priority_queue<_Tp, _Sequence, _Compare>` Class Template Reference

A standard container automatically sorting its contents.

Public Types

- typedef `_Sequence::const_reference` **const_reference**
- typedef `_Sequence` **container_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size_type**
- typedef `_Sequence::value_type` **value_type**

Public Member Functions

- [`priority_queue`](#) (`const _Compare &__x`, `const _Sequence &__s`)
- **priority_queue** (`const _Compare &__x=_Compare()`, `_Sequence &&__s=_Sequence()`)
- `template<typename _InputIterator >`
priority_queue (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__x=_Compare()`, `_Sequence &&__s=_Sequence()`)
- `template<typename _InputIterator >`
[`priority_queue`](#) (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__x`, `const _Sequence &__s`)
- `template<typename... _Args>`
emplace (`_Args &&...__args`)
- `bool empty () const`
- `void pop ()`
- `void push (const value_type &__x)`
- `void push (value_type &&__x)`
- `size_type size () const`
- `void swap (priority_queue &__pq)`
- `const_reference top () const`

Protected Attributes

- `_Sequence` **c**
- `_Compare` **comp**

5.618.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _-
Compare = less<typename _Sequence::value_type>> class std::priority_queue<
_Tp, _Sequence, _Compare >
```

A standard container automatically sorting its contents. This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

Note

No equality/comparison operators are provided for `priority_queue`. Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

Definition at line 353 of file `stl_queue.h`.

5.618.2 Constructor & Destructor Documentation

5.618.2.1 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
const _Compare & __x, const _Sequence & __s) [inline,
explicit]`

Default constructor creates no elements.

Definition at line 388 of file `stl_queue.h`.

References `std::make_heap()`.

5.618.2.2 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
template<typename _InputIterator > std::priority_queue< _Tp,
_Sequence, _Compare >::priority_queue (_InputIterator __first,
_InputIterator __last, const _Compare & __x, const _Sequence &
__s) [inline]`

Builds a queue from a range.

Parameters

first An input iterator.
last An input iterator.
x A comparison functor describing a strict weak ordering.
s An initial sequence with which to start.

Begins by copying *s*, inserting a copy of the elements from [*first*,*last*) into the copy of *s*, then ordering the copy according to *x*.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 428 of file `stl_queue.h`.

References `std::make_heap()`.

5.618.3 Member Function Documentation

5.618.3.1 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
bool std::priority_queue<_Tp, _Sequence, _Compare>::empty ()
const [inline]`

Returns true if the queue is empty.

Definition at line 454 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

5.618.3.2 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
void std::priority_queue<_Tp, _Sequence, _Compare>::pop ()
[inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 517 of file `stl_queue.h`.

References `std::pop_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

5.618.3.3 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
void std::priority_queue< _Tp, _Sequence, _Compare >::push (
const value_type & __x) [inline]`

Add data to the queue.

Parameters

x Data to be added.

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 482 of file `stl_queue.h`.

References `std::push_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

5.618.3.4 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
size_type std::priority_queue< _Tp, _Sequence, _Compare >::size (
) const [inline]`

Returns the number of elements in the queue.

Definition at line 459 of file `stl_queue.h`.

5.618.3.5 `template<typename _Tp, typename _Sequence = vector<_Tp>,
typename _Compare = less<typename _Sequence::value_type>>
const_reference std::priority_queue< _Tp, _Sequence, _Compare
>::top () const [inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 467 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

The documentation for this class was generated from the following file:

- [stl_queue.h](#)

5.619 `std::promise< _Res >` Class Template Reference

Primary template for promise.

Public Member Functions

- `promise` (`promise` &&__rhs)
- `promise` (const `promise` &)
- `template<typename _Allocator > promise` (`allocator_arg_t`, const `_Allocator` &__a)
- `future< _Res > get_future` ()
- `promise` & `operator=` (const `promise` &)
- `promise` & `operator=` (`promise` &&__rhs)
- void `set_exception` (`exception_ptr` __p)
- void `set_value` (`_Res` &&__r)
- void `set_value` (const `_Res` &__r)
- void `swap` (`promise` &__rhs)

Friends

- class `_State::_Setter`

5.619.1 Detailed Description

`template<typename _Res> class std::promise< _Res >`

Primary template for promise.

Definition at line 820 of file `future`.

The documentation for this class was generated from the following file:

- `future`

5.620 `std::promise< _Res & >` Class Template Reference

Partial specialization for `promise<R&>`

Public Member Functions

- `promise` (`promise` &&__rhs)
- `promise` (const `promise` &)
- `template<typename _Allocator > promise` (`allocator_arg_t`, const `_Allocator` &__a)
- `future< _Res & > get_future` ()
- `promise` & `operator=` (const `promise` &)
- `promise` & `operator=` (`promise` &&__rhs)
- void `set_exception` (exception_ptr __p)
- void `set_value` (`_Res` &__r)
- void `swap` (`promise` &__rhs)

Friends

- class `_State::_Setter`

5.620.1 Detailed Description

`template<typename _Res> class std::promise< _Res & >`

Partial specialization for `promise<R&>`

Definition at line 912 of file `future`.

The documentation for this class was generated from the following file:

- `future`

5.621 `std::promise< void >` Class Template Reference

Explicit specialization for `promise<void>`

Public Member Functions

- `promise` (`promise` &&__rhs)
- `promise` (const `promise` &)
- `template<typename _Allocator > promise` (`allocator_arg_t`, const `_Allocator` &__a)
- `future< void > get_future` ()
- `promise` & `operator=` (const `promise` &)
- `promise` & `operator=` (`promise` &&__rhs)

- void **set_exception** (exception_ptr __p)
- void **set_value** ()
- void **swap** ([promise](#) &__rhs)

Friends

- class **_State::_Setter**

5.621.1 Detailed Description

template<> class std::promise< void >

Explicit specialization for [promise<void>](#)

Definition at line 987 of file future.

The documentation for this class was generated from the following file:

- [future](#)

5.622 std::queue< _Tp, _Sequence > Class Template Reference

A standard container giving FIFO behavior.

Public Types

- typedef _Sequence::const_reference **const_reference**
- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- [queue](#) (const _Sequence &__c)
- **queue** (_Sequence &&__c=_Sequence())
- const_reference [back](#) () const
- reference [back](#) ()
- template<typename... _Args>
void **emplace** (_Args &&...__args)

- bool `empty` () const
- reference `front` ()
- const_reference `front` () const
- void `pop` ()
- void `push` (value_type &&__x)
- void `push` (const value_type &__x)
- size_type `size` () const
- void `swap` (queue &__q)

Protected Attributes

- `_Sequence` `c`

Friends

- `template<typename _Tp1, typename _Seq1 >`
`bool operator< (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)`
- `template<typename _Tp1, typename _Seq1 >`
`bool operator== (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)`

5.622.1 Detailed Description

`template<typename _Tp, typename _Sequence = deque<_Tp>> class std::queue< _Tp, _Sequence >`

A standard container giving FIFO behavior. Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 89 of file `stl_queue.h`.

5.622.2 Constructor & Destructor Documentation

5.622.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
std::queue<_Tp, _Sequence>::queue (const _Sequence & __c)
[inline, explicit]`

Default constructor creates no elements.

Definition at line 134 of file `stl_queue.h`.

5.622.3 Member Function Documentation

5.622.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue<_Tp, _Sequence>::back () [inline]`

Returns a read/write reference to the data at the last element of the queue.

Definition at line 181 of file `stl_queue.h`.

5.622.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue<_Tp, _Sequence>::back () const
[inline]`

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 192 of file `stl_queue.h`.

5.622.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::queue<_Tp, _Sequence>::empty () const [inline]`

Returns true if the queue is empty.

Definition at line 146 of file `stl_queue.h`.

5.622.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue<_Tp, _Sequence>::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 170 of file `stl_queue.h`.

5.622.3.5 `template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::front () [inline]`

Returns a read/write reference to the data at the first element of the queue.

Definition at line 159 of file stl_queue.h.

5.622.3.6 `template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::pop () [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 234 of file stl_queue.h.

5.622.3.7 `template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::push (const value_type & __x
) [inline]`

Add data to the end of the queue.

Parameters

x Data to be added.

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 208 of file stl_queue.h.

5.622.3.8 `template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::queue< _Tp, _Sequence >::size () const [inline]`

Returns the number of elements in the queue.

Definition at line 151 of file stl_queue.h.

5.622.4 Member Data Documentation

5.622.4.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
_Sequence std::queue<_Tp, _Sequence>::c [protected]`

'c' is the underlying container. Maintainers wondering why this isn't uglified as per style guidelines should note that this name is specified in the standard, [23.2.3.1]. (Why? Presumably for the same reason that it's protected instead of private: to allow derivation. But none of the other containers allow for derivation. Odd.)

Definition at line 122 of file `stl_queue.h`.

Referenced by `std::operator==()`.

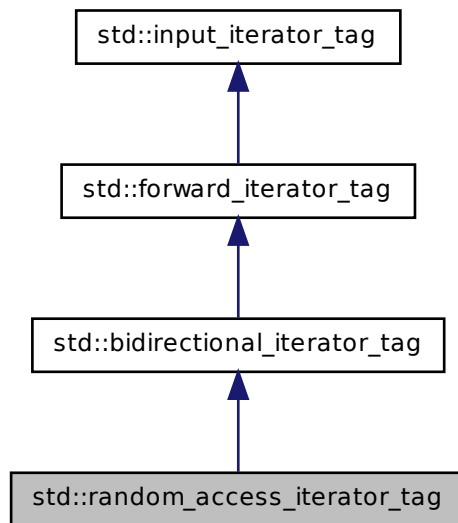
The documentation for this class was generated from the following file:

- [stl_queue.h](#)

5.623 `std::random_access_iterator_tag` Struct Reference

Random-access iterators support a superset of bidirectional /// iterator operations.

Inheritance diagram for `std::random_access_iterator_tag`:



5.623.1 Detailed Description

Random-access iterators support a superset of bidirectional /// iterator operations.

Definition at line 102 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

5.624 `std::random_device` Class Reference

Public Types

- typedef unsigned int [result_type](#)

Public Member Functions

- **random_device** (const [std::string](#) &__token="mt19937")
- **random_device** (const [random_device](#) &)
- double **entropy** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- [result_type](#) **operator**() ()
- void **operator=** (const [random_device](#) &)

5.624.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1524 of file random.h.

5.624.2 Member Typedef Documentation

5.624.2.1 typedef unsigned int std::random_device::result_type

The type of the generated random value.

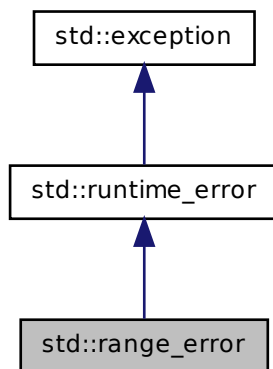
Definition at line 1528 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

5.625 `std::range_error` Class Reference

Inheritance diagram for `std::range_error`:



Public Member Functions

- **`range_error`** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.625.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 126 of file `stdexcept`.

5.625.2 Member Function Documentation

5.625.2.1 `virtual const char* std::runtime_error::what () const throw ()` [**virtual**, **inherited**]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.626 `std::ratio< _Num, _Den >` Struct Template Reference

Provides compile-time rational arithmetic.

Public Member Functions

- **static_assert** (`_Den!=0`, "denominator cannot be zero")
- **static_assert** (`_Num >= __INTMAX_MAX__ && _Den >= __INTMAX_MAX__`, "out of range")

Static Public Attributes

- static const intmax_t **den**
- static const intmax_t **num**

5.626.1 Detailed Description

`template<intmax_t _Num, intmax_t _Den = 1> struct std::ratio< _Num, _Den >`

Provides compile-time rational arithmetic. This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type intmax_t. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

Definition at line 151 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.627 std::ratio_add< _R1, _R2 > Struct Template Reference

[ratio_add](#)

Public Types

- typedef [ratio](#)< __safe_add< __safe_multiply< _R1::num,(_R2::den/ __gcd)>::value, __safe_multiply< _R2::num,(_R1::den/ __gcd)>::value >::value, __safe_multiply< _R1::den,(_R2::den/ __gcd)>::value > **type**

5.627.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_add< _R1, _R2 >
```

[ratio_add](#)

Definition at line 173 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.628 std::ratio_divide< _R1, _R2 > Struct Template Reference

[ratio_divide](#)

Public Types

- typedef [ratio_multiply](#)< _R1, [ratio](#)< _R2::den, _R2::num > >::[type](#) **type**

Public Member Functions

- **static_assert** (_R2::num!=0,"division by 0")

5.628.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_divide< _R1, _R2 >`

[ratio_divide](#)

Definition at line 216 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.629 std::ratio_equal< _R1, _R2 > Struct Template Reference

[ratio_equal](#)

Inherits `integral_constant< bool, _R1::num==_R2::num &&_R1::den==_R2::den >`.

5.629.1 Detailed Description

`template<typename _R1, typename _R2> struct std::ratio_equal< _R1, _R2 >`

[ratio_equal](#)

Definition at line 227 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.630 std::ratio_multiply< _R1, _R2 > Struct Template Reference

[ratio_multiply](#)

Public Types

- typedef [ratio](#)< `__safe_multiply<(_R1::num/__gcd1),(_R2::num/__gcd2)>::value,` `__safe_multiply<(_R1::den/__gcd2),(_R2::den/__gcd1)>::value` > **type**

5.630.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_multiply< _R1, _R2  
>
```

[ratio_multiply](#)

Definition at line 198 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.631 `std::ratio_not_equal< _R1, _R2 >` Struct Template Reference

[ratio_not_equal](#)

Inherits `integral_constant< bool,!ratio_equal< _R1, _R2 >::value >`.

5.631.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_not_equal< _R1, _R2  
>
```

[ratio_not_equal](#)

Definition at line 233 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.632 `std::ratio_subtract< _R1, _R2 >` Struct Template Reference

[ratio_subtract](#)

Public Types

- typedef [ratio_add](#)< _R1, [ratio](#)<-_R2::num, _R2::den > >::type type

5.632.1 Detailed Description

```
template<typename _R1, typename _R2> struct std::ratio_subtract< _R1, _R2
>
```

[ratio_subtract](#)

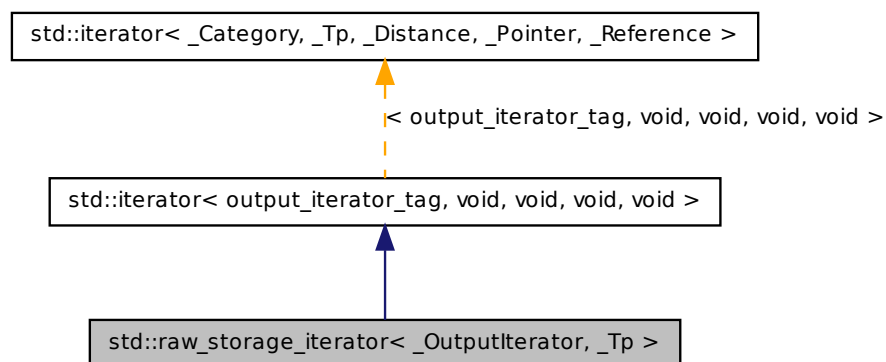
Definition at line 189 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

5.633 `std::raw_storage_iterator< _OutputIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::raw_storage_iterator< _OutputIterator, _Tp >`:



Public Types

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) `iterator_category`
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)

Public Member Functions

- `raw_storage_iterator` (`_OutputIterator __x`)
- `raw_storage_iterator` & `operator*` ()
- `raw_storage_iterator`< `_OutputIterator, _Tp` > & `operator++` ()
- `raw_storage_iterator`< `_OutputIterator, _Tp` > `operator++` (int)
- `raw_storage_iterator` & `operator=` (const `_Tp` & `__element`)

Protected Attributes

- `_OutputIterator _M_iter`

5.633.1 Detailed Description

`template<class _OutputIterator, class _Tp> class std::raw_storage_iterator<_OutputIterator, _Tp>`

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 67 of file `stl_raw_storage_iter.h`.

5.633.2 Member Typedef Documentation

5.633.2.1 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type` [*inherited*]

Distance between iterators is represented as this type.

Definition at line 124 of file `stl_iterator_base_types.h`.

5.633.2.2 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category` [*inherited*]

One of the [tag types](#).

Definition at line 120 of file `stl_iterator_base_types.h`.

5.633.2.3 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer` [*inherited*]

This type represents a pointer-to-value_type.

Definition at line 126 of file `stl_iterator_base_types.h`.

5.633.2.4 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 128 of file stl_iterator_base_types.h.

5.633.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file stl_iterator_base_types.h.

The documentation for this class was generated from the following file:

- [stl_raw_storage_iter.h](#)

5.634 std::recursive_mutex Class Reference

[recursive_mutex](#)

Public Types

- `typedef __native_type * native_handle_type`

Public Member Functions

- `recursive_mutex` (const [recursive_mutex](#) &)
- `void lock ()`
- `native_handle_type native_handle ()`
- `recursive_mutex & operator=` (const [recursive_mutex](#) &)
- `bool try_lock ()`
- `void unlock ()`

5.634.1 Detailed Description

[recursive_mutex](#)

Definition at line 114 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

5.635 `std::recursive_timed_mutex` Class Reference

[`recursive_timed_mutex`](#)

Public Types

- typedef `__native_type *` **`native_handle_type`**

Public Member Functions

- **`recursive_timed_mutex`** (const [`recursive_timed_mutex`](#) &)
- void **`lock`** ()
- `native_handle_type` **`native_handle`** ()
- [`recursive_timed_mutex`](#) & **`operator=`** (const [`recursive_timed_mutex`](#) &)
- bool **`try_lock`** ()
- template<class `_Rep`, class `_Period` >
bool **`try_lock_for`** (const [`chrono::duration`](#)< `_Rep`, `_Period` > &__rtime)
- template<class `_Clock`, class `_Duration` >
bool **`try_lock_until`** (const [`chrono::time_point`](#)< `_Clock`, `_Duration` > &__-
atime)
- void **`unlock`** ()

5.635.1 Detailed Description

[`recursive_timed_mutex`](#)

Definition at line 270 of file `mutex`.

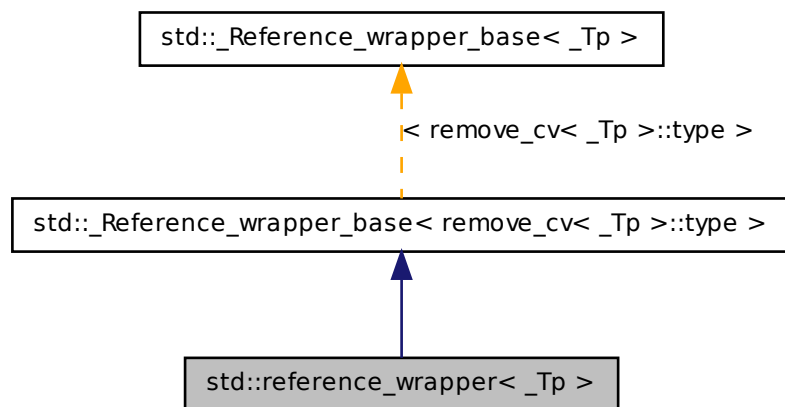
The documentation for this class was generated from the following file:

- [`mutex`](#)

5.636 `std::reference_wrapper<_Tp>` Class Template Reference

Primary class template for [`reference_wrapper`](#).

Inheritance diagram for `std::reference_wrapper<_Tp>`:



Public Types

- `typedef _Tp type`

Public Member Functions

- `reference_wrapper` (`_Tp &__indata`)
- `reference_wrapper` (`_Tp &&`)
- `reference_wrapper` (`const reference_wrapper<_Tp> &__inref`)
- `_Tp & get ()` `const`
- `operator _Tp & ()` `const`
- `template<typename... _Args>`
`result_of< _M_func_type(_Args...)>::type operator() (_Args &&...__args)`
`const`
- `reference_wrapper & operator=` (`const reference_wrapper<_Tp> &__inref`)

5.636.1 Detailed Description

`template<typename _Tp> class std::reference_wrapper< _Tp >`

Primary class template for [reference_wrapper](#).

Definition at line 420 of file functional.

The documentation for this class was generated from the following file:

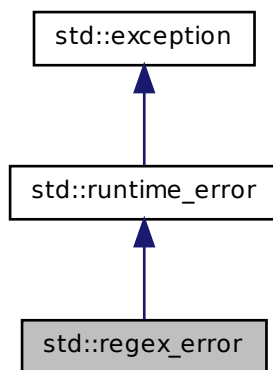
- [functional](#)

5.637 std::regex_error Class Reference

A regular expression exception class.

The regular expression library throws objects of this class on error.

Inheritance diagram for std::regex_error:



Public Member Functions

- [regex_error](#) ([regex_constants::error_type](#) __ecode)
- [regex_constants::error_type](#) code () const
- virtual const char * [what](#) () const throw ()

Protected Attributes

- [regex_constants::error_type](#) `_M_code`

5.637.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 127 of file `regex_error.h`.

5.637.2 Constructor & Destructor Documentation

5.637.2.1 `std::regex_error::regex_error (regex_constants::error_type __ecode) [inline, explicit]`

Constructs a [regex_error](#) object.

Parameters

ecode the regex error code.

Definition at line 137 of file `regex_error.h`.

5.637.3 Member Function Documentation

5.637.3.1 `regex_constants::error_type std::regex_error::code () const [inline]`

Gets the regex error code.

Returns

the regex error code.

Definition at line 147 of file `regex_error.h`.

5.637.3.2 `virtual const char* std::runtime_error::what () const throw () [virtual, inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

5.638 `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference 2949

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [regex_error.h](#)

5.638 `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef const [value_type](#) * **pointer**
- typedef const [value_type](#) & **reference**
- typedef [basic_regex< _Ch_type, _Rx_traits >](#) **regex_type**
- typedef [match_results< _Bi_iter >](#) **value_type**

Public Member Functions

- [regex_iterator](#) ()
- [regex_iterator](#) ([_Bi_iter](#) __a, [_Bi_iter](#) __b, const [regex_type](#) &__re, [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- [regex_iterator](#) (const [regex_iterator](#) &__rhs)
- bool [operator!=](#) (const [regex_iterator](#) &__rhs)
- const [value_type](#) & [operator*](#) ()
- [regex_iterator](#) & [operator++](#) ()
- [regex_iterator](#) [operator++](#) (int)
- const [value_type](#) * [operator->](#) ()
- [regex_iterator](#) & [operator=](#) (const [regex_iterator](#) &__rhs)
- bool [operator==](#) (const [regex_iterator](#) &__rhs)

5.638.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> class
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Definition at line 2149 of file `regex.h`.

5.638.2 Constructor & Destructor Documentation

5.638.2.1 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type,
_Rx_traits >::regex_iterator ()`

Provides a singular iterator, useful for indicating one-past-the-end of a range.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.638.2.2 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type,
_Rx_traits >::regex_iterator (_Bi_iter __a, _Bi_iter __b, const
regex_type & __re, regex_constants::match_flag_type __m =
regex_constants::match_default)`

Constructs a `regex_iterator`...

Parameters

a [IN] The start of a text range to search.

b [IN] One-past-the-end of the text range to search.

re [IN] The regular expression to match.

m [IN] Policy flags for match rules.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.638.2.3 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type,
_Rx_traits >::regex_iterator (const regex_iterator< _Bi_iter,
_Ch_type, _Rx_traits > & __rhs)`

Copy constructs a `regex_iterator`.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.638.3 Member Function Documentation

5.638.3.1 `template<typename _Bi_iter, typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::operator!=(const regex_iterator< _Bi_iter,
_Ch_type, _Rx_traits > & __rhs)`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.638.3.2 `template<typename _Bi_iter, typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> const value_type& std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator*()`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.638.3.3 `template<typename _Bi_iter, typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> regex_iterator std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator++(int)`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

5.638.3.4 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> regex_iterator& std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator++ ()`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.638.3.5 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> const value_type* std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator-> ()`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.638.3.6 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> regex_iterator& std::regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::operator= (const regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits > & __rhs)`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

5.638.3.7 `template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::operator== (const regex_iterator<
_Bi_iter, _Ch_type, _Rx_traits > & __rhs)`

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

The documentation for this class was generated from the following file:

- [regex.h](#)

5.639 `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef `std::forward_iterator_tag` **iterator_category**
- typedef const `value_type` * **pointer**
- typedef const `value_type` & **reference**
- typedef `basic_regex< _Ch_type, _Rx_traits >` **regex_type**
- typedef `sub_match< _Bi_iter >` **value_type**

Public Member Functions

- `regex_token_iterator` ()
- `regex_token_iterator` (`_Bi_iter` __a, `_Bi_iter` __b, const `regex_type` &__re, int __submatch=0, `regex_constants::match_flag_type` __m=`regex_constants::match_default`)
- `template<std::size_t _Nm>`
`regex_token_iterator` (`_Bi_iter` __a, `_Bi_iter` __b, const `regex_type` &__re, const int(&__submatches)[_Nm], `regex_constants::match_flag_type` __m=`regex_constants::match_default`)
- `regex_token_iterator` (const `regex_token_iterator` &__rhs)
- `regex_token_iterator` (`_Bi_iter` __a, `_Bi_iter` __b, const `regex_type` &__re, const `std::vector< int >` &__submatches, `regex_constants::match_flag_type` __m=`regex_constants::match_default`)
- bool `operator!=` (const `regex_token_iterator` &__rhs)
- const `value_type` & `operator*` ()
- `regex_token_iterator` & `operator++` ()
- `regex_token_iterator` `operator++` (int)
- const `value_type` * `operator->` ()
- `regex_token_iterator` & `operator=` (const `regex_token_iterator` &__rhs)
- bool `operator==` (const `regex_token_iterator` &__rhs)

5.639.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> class
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a [std::sub_match](#) object.

Definition at line 2264 of file `regex.h`.

5.639.2 Constructor & Destructor Documentation

5.639.2.1 `template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ()`

Default constructs a `regex_token_iterator`.

Todo

Implement this function.

A default-constructed `regex_token_iterator` is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

5.639.2.2 `template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type & __re, int __submatch = 0, regex_constants::match_flag_type __m = regex_constants::match_default)`

Constructs a `regex_token_iterator`...

Parameters

a [IN] The start of the text to search.

b [IN] One-past-the-end of the text to search.

re [IN] The regular expression to search for.

submatch [IN] Which submatch to return. There are some special values for this parameter:

- -1 each enumerated subexpression does NOT match the regular expression (aka field splitting)
- 0 the entire string matching the subexpression is returned for each match within the text.
- >0 enumerates only the indicated subexpression from a match within the text.

m [IN] Policy flags for match rules.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

```
5.639.2.3 template<typename _Bi_iter, typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> std::regex_token_iterator<_Bi_iter,
_Ch_type, _Rx_traits>::regex_token_iterator ( _Bi_iter __a,
_Bi_iter __b, const regex_type & __re, const std::vector<int
> & __submatches, regex_constants::match_flag_type __m =
regex_constants::match_default )
```

Constructs a `regex_token_iterator`...

Parameters

a [IN] The start of the text to search.

b [IN] One-past-the-end of the text to search.

re [IN] The regular expression to search for.

submatches [IN] A list of subexpressions to return for each regular expression match within the text.

m [IN] Policy flags for match rules.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

```

5.639.2.4 template<typename _Bi_iter , typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> template<std::size_t
_Nm> std::regex_token_iterator< _Bi_iter, _Ch_type,
_Rx_traits >::regex_token_iterator ( _Bi_iter __a,
_Bi_iter __b, const regex_type & __re, const int(&)
__submatches[_Nm], regex_constants::match_flag_type __m =
regex_constants::match_default )

```

Constructs a `regex_token_iterator`...

Parameters

- a* [IN] The start of the text to search.
- b* [IN] One-past-the-end of the text to search.
- re* [IN] The regular expression to search for.
- submatches* [IN] A list of subexpressions to return for each regular expression match within the text.
- m* [IN] Policy flags for match rules.

Todo

Implement this function.

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

```

5.639.2.5 template<typename _Bi_iter , typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> std::regex_token_iterator<
_Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( const
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs )

```

Copy constructs a `regex_token_iterator`.

Parameters

- rhs* [IN] A `regex_token_iterator` to copy.

Todo

Implement this function.

5.639.3 Member Function Documentation

5.639.3.1 `template<typename _Bi_iter, typename _Ch_type = typename
iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
regex_traits<_Ch_type>> bool std::regex_token_iterator< _Bi_iter,
_Ch_type, _Rx_traits >::operator!=(const regex_token_iterator<
_Bi_iter, _Ch_type, _Rx_traits > & __rhs)`

Compares a regex_token_iterator to another for inequality.

Todo

Implement this function.

5.639.3.2 `template<typename _Bi_iter, typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> const value_type&
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator* ()`

Dereferences a regex_token_iterator.

Todo

Implement this function.

5.639.3.3 `template<typename _Bi_iter, typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator++ (int)`

Postincrements a regex_token_iterator.

Todo

Implement this function.

5.639.3.4 `template<typename _Bi_iter , typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator&
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator++ ()`

Increments a `regex_token_iterator`.

Todo

Implement this function.

5.639.3.5 `template<typename _Bi_iter , typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> const value_type*
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator-> ()`

Selects a `regex_token_iterator` member.

Todo

Implement this function.

5.639.3.6 `template<typename _Bi_iter , typename _Ch_type =
typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> regex_token_iterator&
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits
>::operator= (const regex_token_iterator< _Bi_iter, _Ch_type,
_Rx_traits > & __rhs)`

Assigns a `regex_token_iterator` to another.

Parameters

rhs [IN] A `regex_token_iterator` to copy.

Todo

Implement this function.

```
5.639.3.7  template<typename _Bi_iter , typename _Ch_type = typename
            iterator_traits<_Bi_iter>::value_type, typename _Rx_traits =
            regex_traits<_Ch_type>> bool std::regex_token_iterator< _Bi_iter,
            _Ch_type, _Rx_traits >::operator==( const regex_token_iterator<
            _Bi_iter, _Ch_type, _Rx_traits > & __rhs )
```

Compares a `regex_token_iterator` to another for equality.

Todo

Implement this function.

The documentation for this class was generated from the following file:

- [regex.h](#)

5.640 `std::regex_traits< _Ch_type >` Struct Template Reference

Describes aspects of a regular expression.

Public Types

- typedef `std::ctype_base::mask` **char_class_type**
- typedef `_Ch_type` **char_type**
- typedef [std::locale](#) **locale_type**
- typedef [std::basic_string](#)< `char_type` > **string_type**

Public Member Functions

- [regex_traits](#) ()
- [locale_type](#) [getloc](#) () const
- [locale_type](#) [imbue](#) ([locale_type](#) __loc)
- bool [isctype](#) (`_Ch_type` __c, `char_class_type` __f) const
- template<typename `_Fwd_iter` >
`char_class_type` [lookup_classname](#) (`_Fwd_iter` __first, `_Fwd_iter` __last, bool __-
__icase=false) const
- template<typename `_Fwd_iter` >
[string_type](#) [lookup_collatename](#) (`_Fwd_iter` __first, `_Fwd_iter` __last) const
- template<typename `_Fwd_iter` >
[string_type](#) [transform](#) (`_Fwd_iter` __first, `_Fwd_iter` __last) const

- `template<typename _Fwd_iter >`
`string_type transform_primary (_Fwd_iter __first, _Fwd_iter __last) const`
- `char_type translate (char_type __c) const`
- `char_type translate_nocase (char_type __c) const`
- `int value (_Ch_type __ch, int __radix) const`

Static Public Member Functions

- `static std::size_t length (const char_type *__p)`

Protected Attributes

- `locale_type _M_locale`

5.640.1 Detailed Description

`template<typename _Ch_type> struct std::regex_traits< _Ch_type >`

Describes aspects of a regular expression. A regular expression traits class that satisfies the requirements of section [28.7].

The class `regex` is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 52 of file `regex.h`.

5.640.2 Constructor & Destructor Documentation

5.640.2.1 `template<typename _Ch_type > std::regex_traits< _Ch_type >::regex_traits () [inline]`

Constructs a default traits object.

Definition at line 64 of file `regex.h`.

5.640.3 Member Function Documentation

5.640.3.1 `template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::getloc () const [inline]`

Gets a copy of the current locale in use by the `regex_traits` object.

Definition at line 273 of file `regex.h`.

5.640.3.2 `template<typename _Ch_type > locale_type std::regex_traits<
_Ch_type >::imbue (locale_type __loc) [inline]`

Imbues the [regex_traits](#) object with a copy of a new locale.

Parameters

loc A locale.

Returns

a copy of the previous locale in use by the [regex_traits](#) object.

Note

Calling imbue with a different locale than the one currently in use invalidates all cached data held by *this.

Definition at line 262 of file regex.h.

References std::swap().

5.640.3.3 `template<typename _Ch_type > static std::size_t std::regex_traits<
_Ch_type >::length (const char_type * __p) [inline,
static]`

Gives the length of a C-style string starting at __p.

Parameters

__p a pointer to the start of a character sequence.

Returns

the number of characters between *__p and the first default-initialized value of type char_type. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 78 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::length().

5.640.3.4 `template<typename _Ch_type > template<typename _Fwd_iter >
char_class_type std::regex_traits< _Ch_type >::lookup_classname (
_Fwd_iter __first, _Fwd_iter __last, bool __icase = false) const
[inline]`

Maps one or more characters to a named character classification.

Parameters

first beginning of the character sequence.

last one-past-the-end of the character sequence.

icase ignores the case of the classification name.

Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[first, last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[:lower:]` is the same as `[:alpha:]`), otherwise a case-dependant classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- `d`
- `w`
- `s`
- `alnum`
- `alpha`
- `blank`
- `cntrl`
- `digit`
- `graph`
- `lower`
- `print`
- `punct`
- `space`
- `upper`
- `xdigit`

Todo

Implement this function.

Definition at line 218 of file `regex.h`.

Referenced by `std::regex_traits<_Ch_type>::isctype()`.

5.640.3.5 `template<typename _Ch_type > template<typename _Fwd_iter >
string_type std::regex_traits< _Ch_type >::lookup_collatename (
_Fwd_iter __first, _Fwd_iter __last) const [inline]`

Gets a collation element by name.

Parameters

first beginning of the collation element name.

last one-past-the-end of the collation element name.

Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [first, last). Returns an empty string if the character sequence is not a valid collating element.

Todo

Implement this function.

Definition at line 175 of file regex.h.

5.640.3.6 `template<typename _Ch_type > template<typename _Fwd_iter >
string_type std::regex_traits< _Ch_type >::transform (_Fwd_iter
__first, _Fwd_iter __last) const [inline]`

Gets a sort key for a character sequence.

Parameters

first beginning of the character sequence.

last one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range [F1, F2) such that if the character sequence [G1, G2) sorts before the character sequence [H1, H2) then `v.transform(G1, G2) < v.transform(H1, H2)`.

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

Returns

a locale-specific sort key equivalent to the input range.

Exceptions

std::bad_cast if the current locale does not have a collate facet.

Definition at line 131 of file regex.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::size()`, `std::collate< _CharT >::transform()`, and `std::use_facet()`.

5.640.3.7 `template<typename _Ch_type > template<typename _Fwd_iter > string_type std::regex_traits< _Ch_type >::transform_primary (_Fwd_iter __first, _Fwd_iter __last) const [inline]`

Gets a sort key for a character sequence, independant of case.

Parameters

first beginning of the character sequence.

last one-past-the-end of the character sequence.

Effects: if `typeid(use_facet<collate<_Ch_type> >) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(first, last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

Todo

Implement this function.

Definition at line 157 of file regex.h.

5.640.3.8 `template<typename _Ch_type > char_type std::regex_traits< _Ch_type >::translate (char_type __c) const [inline]`

Performs the identity translation.

Parameters

c A character to the locale-specific character set.

Returns

c.

Definition at line 89 of file regex.h.

5.640.3.9 `template<typename _Ch_type > char_type std::regex_traits< _Ch_type >::translate_nocase (char_type __c) const [inline]`

Translates a character into a case-insensitive equivalent.

Parameters

c A character to the locale-specific character set.

Returns

the locale-specific lower-case equivalent of *c*.

Exceptions

[*std::bad_cast*](#) if the imbued locale does not support the ctype facet.

Definition at line 102 of file `regex.h`.

References `std::tolower()`.

The documentation for this struct was generated from the following file:

- [regex.h](#)

5.641 `std::remove_reference< _Tp >` Struct Template Reference

[remove_reference](#)

Public Types

- `typedef _Tp type`

5.641.1 Detailed Description

`template<typename _Tp> struct std::remove_reference< _Tp >`

[remove_reference](#)

Definition at line 98 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type_traits](#)

5.642 `std::reverse_iterator< _Iterator >` Class Template Reference

Inheritance diagram for `std::reverse_iterator< _Iterator >`:



Public Types

- `typedef __traits_type::difference_type` [difference_type](#)
- `typedef iterator_traits< _Iterator >::iterator_category` [iterator_category](#)
- `typedef _Iterator` **iterator_type**
- `typedef __traits_type::pointer` [pointer](#)
- `typedef __traits_type::reference` [reference](#)
- `typedef iterator_traits< _Iterator >::value_type` [value_type](#)

Public Member Functions

- [reverse_iterator](#) ()
- [reverse_iterator](#) (iterator_type __x)
- `template<typename _Iter >`
 [reverse_iterator](#) (const [reverse_iterator](#)< _Iter > &__x)
- [reverse_iterator](#) (const [reverse_iterator](#) &__x)
- iterator_type [base](#) () const
- [reference](#) [operator*](#) () const
- [reverse_iterator](#) [operator+](#) (difference_type __n) const
- [reverse_iterator](#) & [operator++](#) ()
- [reverse_iterator](#) [operator++](#) (int)
- [reverse_iterator](#) & [operator+=](#) (difference_type __n)
- [reverse_iterator](#) [operator-](#) (difference_type __n) const
- [reverse_iterator](#) & [operator--](#) ()
- [reverse_iterator](#) [operator--](#) (int)
- [reverse_iterator](#) & [operator-=](#) (difference_type __n)
- [pointer](#) [operator->](#) () const
- [reference](#) [operator\[\]](#) (difference_type __n) const

Protected Types

- `typedef iterator_traits< _Iterator > __traits_type`

Protected Attributes

- `_Iterator current`

5.642.1 Detailed Description

template<typename _Iterator> class std::reverse_iterator< _Iterator >

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &(i - 1)
```

This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 95 of file `stl_iterator.h`.

5.642.2 Member Typedef Documentation

5.642.2.1 **template<typename _Iterator> typedef __traits_difference_type std::reverse_iterator< _Iterator >::difference_type**

Distance between iterators is represented as this type.

Reimplemented from [std::iterator< iterator_traits< _Iterator >::iterator_category, iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type, iterator_traits< _Iterator >::pointer, iterator_traits< _Iterator >::reference >](#).

Definition at line 109 of file `stl_iterator.h`.

5.642.2.2 **typedef iterator_traits< _Iterator >::iterator_category std::iterator< iterator_traits< _Iterator >::iterator_category, iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type, iterator_traits< _Iterator >::pointer, iterator_traits< _Iterator >::reference >::iterator_category [inherited]**

One of the [tag types](#).

Definition at line 120 of file `stl_iterator_base_types.h`.

5.642.2.3 `template<typename _Iterator> typedef __traits_type::pointer
std::reverse_iterator< _Iterator >::pointer`

This type represents a pointer-to-value_type.

Reimplemented from `std::iterator< iterator_traits< _Iterator >::iterator_category,
iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type,
iterator_traits< _Iterator >::pointer, iterator_traits< _Iterator >::reference >`.

Definition at line 110 of file `stl_iterator.h`.

5.642.2.4 `template<typename _Iterator> typedef __traits_type::reference
std::reverse_iterator< _Iterator >::reference`

This type represents a reference-to-value_type.

Reimplemented from `std::iterator< iterator_traits< _Iterator >::iterator_category,
iterator_traits< _Iterator >::value_type, iterator_traits< _Iterator >::difference_type,
iterator_traits< _Iterator >::pointer, iterator_traits< _Iterator >::reference >`.

Definition at line 111 of file `stl_iterator.h`.

5.642.2.5 `typedef iterator_traits< _Iterator >::value_type std::iterator<
iterator_traits< _Iterator >::iterator_category , iterator_traits<
_Iterator >::value_type , iterator_traits< _Iterator
>::difference_type , iterator_traits< _Iterator >::pointer
, iterator_traits< _Iterator >::reference >::value_type
[inherited]`

The type "pointed to" by the iterator.

Definition at line 122 of file `stl_iterator_base_types.h`.

5.642.3 Constructor & Destructor Documentation

5.642.3.1 `template<typename _Iterator> std::reverse_iterator< _Iterator
>::reverse_iterator () [inline]`

The default constructor default-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 119 of file `stl_iterator.h`.

5.642.3.2 `template<typename _Iterator> std::reverse_iterator< _Iterator
>::reverse_iterator (iterator_type __x) [inline, explicit]`

This iterator will move in the opposite direction that `x` does.

Definition at line 125 of file `stl_iterator.h`.

5.642.3.3 `template<typename _Iterator> std::reverse_iterator< _Iterator
>::reverse_iterator (const reverse_iterator< _Iterator > & __x)
[inline]`

The copy constructor is normal.

Definition at line 130 of file `stl_iterator.h`.

5.642.3.4 `template<typename _Iterator> template<typename _Iter >
std::reverse_iterator< _Iterator >::reverse_iterator (const
reverse_iterator< _Iter > & __x) [inline]`

A [reverse_iterator](#) across other types can be copied in the normal fashion.

Definition at line 138 of file `stl_iterator.h`.

5.642.4 Member Function Documentation

5.642.4.1 `template<typename _Iterator> iterator_type std::reverse_iterator<
_Iterator >::base () const [inline]`

Returns

`current`, the iterator used for underlying work.

Definition at line 145 of file `stl_iterator.h`.

Referenced by `std::operator==()`.

5.642.4.2 `template<typename _Iterator> reference std::reverse_iterator<
_Iterator >::operator* () const [inline]`

Returns

TODO

Todo

Doc me! See `doc/doxygen/TODO` and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 154 of file stl_iterator.h.

5.642.4.3 `template<typename _Iterator> reverse_iterator
std::reverse_iterator< _Iterator >::operator+ (difference_type __n
) const [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 225 of file stl_iterator.h.

5.642.4.4 `template<typename _Iterator> reverse_iterator&
std::reverse_iterator< _Iterator >::operator++ () [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 175 of file stl_iterator.h.

5.642.4.5 `template<typename _Iterator> reverse_iterator
std::reverse_iterator< _Iterator >::operator++ (int) [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 187 of file stl_iterator.h.

5.642.4.6 `template<typename _Iterator> reverse_iterator&
std::reverse_iterator< _Iterator >::operator+=(difference_type
__n) [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 234 of file stl_iterator.h.

5.642.4.7 `template<typename _Iterator> reverse_iterator
std::reverse_iterator< _Iterator >::operator-(difference_type __n
) const [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 246 of file stl_iterator.h.

5.642.4.8 `template<typename _Iterator> reverse_iterator
std::reverse_iterator< _Iterator >::operator--(int) [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 212 of file stl_iterator.h.

5.642.4.9 `template<typename _Iterator> reverse_iterator&
std::reverse_iterator< _Iterator >::operator-- () [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 200 of file stl_iterator.h.

5.642.4.10 `template<typename _Iterator> reverse_iterator&
std::reverse_iterator< _Iterator >::operator= (difference_type
__n) [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 255 of file stl_iterator.h.

5.642.4.11 `template<typename _Iterator> pointer std::reverse_iterator<
_Iterator >::operator-> () const [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg0000> for more.

Definition at line 166 of file stl_iterator.h.

5.642.4.12 `template<typename _Iterator> reference std::reverse_iterator<
_Iterator >::operator[] (difference_type __n) const [inline]`

Returns

TODO

Todo

Doc me! See doc/doxygen/TODO and <http://gcc.gnu.org/ml/libstdc++/2002-02/msg00003.html> for more.

Definition at line 267 of file stl_iterator.h.

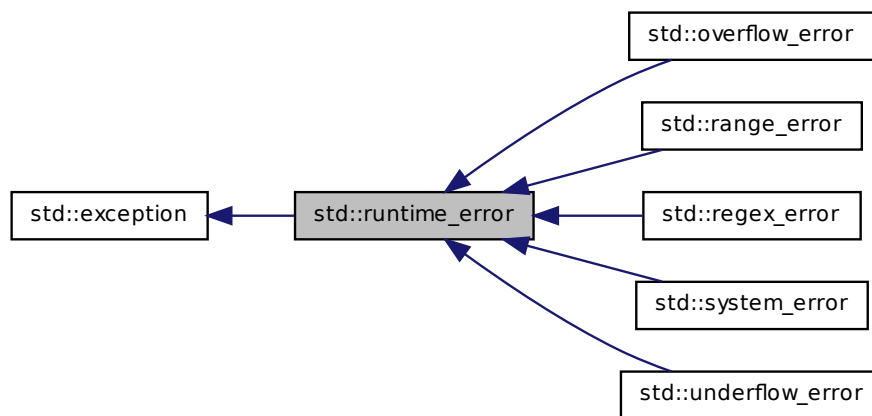
The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

5.643 std::runtime_error Class Reference

One of two subclasses of exception.

Inheritance diagram for std::runtime_error:



Public Member Functions

- `runtime_error` (const `string` &__arg)
- virtual const char * `what` () const throw ()

5.643.1 Detailed Description

One of two subclasses of exception. Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

Definition at line 107 of file `stdexcept`.

5.643.2 Constructor & Destructor Documentation

5.643.2.1 `std::runtime_error::runtime_error (const string & __arg)`
[**explicit**]

Takes a character string describing the error.

5.643.3 Member Function Documentation

5.643.3.1 `virtual const char* std::runtime_error::what () const throw ()`
[**virtual**]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.644 std::seed_seq Class Reference

The [seed_seq](#) class generates sequences of seeds for random number generators.

Public Types

- typedef uint_least32_t [result_type](#)

Public Member Functions

- [seed_seq](#) ()
- template<typename _IntType >
[seed_seq](#) ([std::initializer_list](#)< _IntType > il)

- `template<typename _InputIterator >`
`seed_seq` (`_InputIterator __begin`, `_InputIterator __end`)
- `template<typename _RandomAccessIterator >`
`void generate` (`_RandomAccessIterator __begin`, `_RandomAccessIterator __end`)
- `template<typename OutputIterator >`
`void param` (`OutputIterator __dest`) `const`
- `size_t size` () `const`

5.644.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Definition at line 5303 of file `random.h`.

5.644.2 Member Typedef Documentation

5.644.2.1 `typedef uint_least32_t std::seed_seq::result_type`

The type of the seed vales.

Definition at line 5308 of file `random.h`.

5.644.3 Constructor & Destructor Documentation

5.644.3.1 `std::seed_seq::seed_seq () [inline]`

Default constructor.

Definition at line 5311 of file `random.h`.

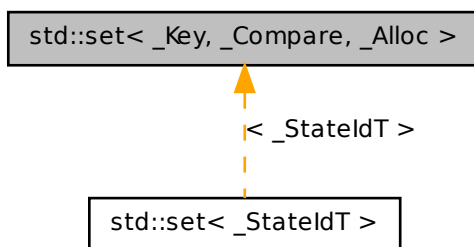
The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.645 `std::set< _Key, _Compare, _Alloc >` Class Template Reference

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Inheritance diagram for `std::set< _Key, _Compare, _Alloc >`:



Public Types

- typedef `_Key` [key_type](#)
- typedef `_Key` [value_type](#)
- typedef `_Compare` [key_compare](#)
- typedef `_Compare` [value_compare](#)
- typedef `_Alloc` [allocator_type](#)
- typedef `_Key_alloc_type::pointer` [pointer](#)
- typedef `_Key_alloc_type::const_pointer` [const_pointer](#)
- typedef `_Key_alloc_type::reference` [reference](#)
- typedef `_Key_alloc_type::const_reference` [const_reference](#)
- typedef `_Rep_type::const_iterator` [iterator](#)
- typedef `_Rep_type::const_iterator` [const_iterator](#)
- typedef `_Rep_type::const_reverse_iterator` [reverse_iterator](#)
- typedef `_Rep_type::const_reverse_iterator` [const_reverse_iterator](#)
- typedef `_Rep_type::size_type` [size_type](#)
- typedef `_Rep_type::difference_type` [difference_type](#)

Public Member Functions

- [set](#) ()
- [set](#) (const `_Compare` &__comp, const [allocator_type](#) &__a=[allocator_type](#)())
- template<typename `_InputIterator` >
[set](#) (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp, const [allocator_type](#) &__a=[allocator_type](#)())
- [set](#) (const [set](#) &__x)

- `template<typename _InputIterator >`
`set` (`_InputIterator` __first, `_InputIterator` __last)
- `set` (`set` &&__x)
- `set` (`initializer_list`< `value_type` > __l, `const _Compare` &__comp=`_Compare`(),
`const allocator_type` &__a=`allocator_type`())
- `iterator begin` () `const`
- `iterator cbegin` () `const`
- `iterator cend` () `const`
- `void clear` ()
- `size_type count` (`const key_type` &__x) `const`
- `reverse_iterator crbegin` () `const`
- `reverse_iterator crend` () `const`
- `bool empty` () `const`
- `iterator end` () `const`
- `iterator erase` (`iterator` __first, `iterator` __last)
- `size_type erase` (`const key_type` &__x)
- `iterator erase` (`iterator` __position)
- `allocator_type get_allocator` () `const`
- `std::pair`< `iterator`, `bool` > `insert` (`const value_type` &__x)
- `iterator insert` (`iterator` __position, `const value_type` &__x)
- `template<typename _InputIterator >`
`void insert` (`_InputIterator` __first, `_InputIterator` __last)
- `void insert` (`initializer_list`< `value_type` > __l)
- `key_compare key_comp` () `const`
- `size_type max_size` () `const`
- `set` & `operator=` (`const set` &__x)
- `set` & `operator=` (`set` &&__x)
- `set` & `operator=` (`initializer_list`< `value_type` > __l)
- `reverse_iterator rbegin` () `const`
- `reverse_iterator rend` () `const`
- `size_type size` () `const`
- `void swap` (`set` &__x)
- `value_compare value_comp` () `const`
- `iterator find` (`const key_type` &__x)
- `const_iterator find` (`const key_type` &__x) `const`
- `iterator lower_bound` (`const key_type` &__x)
- `const_iterator lower_bound` (`const key_type` &__x) `const`
- `iterator upper_bound` (`const key_type` &__x)
- `const_iterator upper_bound` (`const key_type` &__x) `const`
- `std::pair`< `iterator`, `iterator` > `equal_range` (`const key_type` &__x)
- `std::pair`< `const_iterator`, `const_iterator` > `equal_range` (`const key_type` &__x) `const`

Friends

- `template<typename _K1, typename _C1, typename _A1 >`
`bool operator< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 >`
`&)`
- `template<typename _K1, typename _C1, typename _A1 >`
`bool operator== (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 >`
`&)`

5.645.1 Detailed Description

`template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::set< _Key, _Compare, _Alloc >`

A standard container made up of unique keys, which can be retrieved in logarithmic time. Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

Parameters

Key Type of key objects.

Compare Comparison function object type, defaults to `less<Key>`.

Alloc Allocator type, defaults to `allocator<Key>`.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 87 of file `stl_set.h`.

5.645.2 Member Typedef Documentation

5.645.2.1 `template<typename _Key, typename _Compare = std::less<_Key>,`
`typename _Alloc = std::allocator<_Key>> typedef _Alloc std::set<`
`_Key, _Compare, _Alloc >::allocator_type`

Public typedefs.

Definition at line 104 of file `stl_set.h`.

5.645.2.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_iterator std::set< _Key, _Compare, _Alloc
>::const_iterator`

Iterator-related typedefs.

Definition at line 125 of file stl_set.h.

5.645.2.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Key_alloc_type::const_pointer std::set< _Key, _Compare, _Alloc
>::const_pointer`

Iterator-related typedefs.

Definition at line 118 of file stl_set.h.

5.645.2.4 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Key_alloc_type::const_reference std::set< _Key, _Compare, _Alloc
>::const_reference`

Iterator-related typedefs.

Definition at line 120 of file stl_set.h.

5.645.2.5 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc
>::const_reverse_iterator`

Iterator-related typedefs.

Definition at line 127 of file stl_set.h.

5.645.2.6 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::difference_type std::set< _Key, _Compare, _Alloc
>::difference_type`

Iterator-related typedefs.

Definition at line 129 of file stl_set.h.

5.645.2.7 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_iterator std::set< _Key, _Compare, _Alloc
>::iterator`

Iterator-related typedefs.

Definition at line 124 of file `stl_set.h`.

5.645.2.8 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef _Compare
std::set< _Key, _Compare, _Alloc >::key_compare`

Public typedefs.

Definition at line 102 of file `stl_set.h`.

5.645.2.9 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef _Key std::set<
_Key, _Compare, _Alloc >::key_type`

Public typedefs.

Definition at line 100 of file `stl_set.h`.

5.645.2.10 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Key_alloc_type::pointer std::set< _Key, _Compare, _Alloc
>::pointer`

Iterator-related typedefs.

Definition at line 117 of file `stl_set.h`.

5.645.2.11 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Key_alloc_type::reference std::set< _Key, _Compare, _Alloc
>::reference`

Iterator-related typedefs.

Definition at line 119 of file `stl_set.h`.

5.645.2.12 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_reverse_iterator std::set< _Key, _Compare,
_Alloc >::reverse_iterator`

Iterator-related typedefs.

Definition at line 126 of file stl_set.h.

5.645.2.13 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::size_type std::set< _Key, _Compare, _Alloc
>::size_type`

Iterator-related typedefs.

Definition at line 128 of file stl_set.h.

5.645.2.14 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef _Compare
std::set< _Key, _Compare, _Alloc >::value_compare`

Public typedefs.

Definition at line 103 of file stl_set.h.

5.645.2.15 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> typedef _Key std::set<
_Key, _Compare, _Alloc >::value_type`

Public typedefs.

Definition at line 101 of file stl_set.h.

5.645.3 Constructor & Destructor Documentation

5.645.3.1 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::set< _Key,
_Compare, _Alloc >::set () [inline]`

Default constructor creates no elements.

Definition at line 136 of file stl_set.h.

5.645.3.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::set<_Key,
_Compare, _Alloc >::set (const _Compare & __comp, const
allocator_type & __a = allocator_type ()) [inline,
explicit]`

Creates a set with no elements.

Parameters

comp Comparator to use.

a An allocator object.

Definition at line 145 of file stl_set.h.

5.645.3.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > std::set<_Key, _Compare, _Alloc >::set (
_InputIterator __first, _InputIterator __last) [inline]`

Builds a set from a range.

Parameters

first An input iterator.

last An input iterator.

Create a set consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 159 of file stl_set.h.

5.645.3.4 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > std::set<_Key, _Compare, _Alloc >::set (
_InputIterator __first, _InputIterator __last, const _Compare
& __comp, const allocator_type & __a = allocator_type ())
[inline]`

Builds a set from a range.

Parameters

first An input iterator.

last An input iterator.

comp A comparison functor.

a An allocator object.

Create a set consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(first,last)).

Definition at line 175 of file stl_set.h.

```
5.645.3.5  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> std::set< _Key,
              _Compare, _Alloc >::set ( const set< _Key, _Compare, _Alloc > &
              __x ) [inline]
```

Set copy constructor.

Parameters

x A set of identical element and allocator types.

The newly-created set uses a copy of the allocation object used by *x*.

Definition at line 188 of file stl_set.h.

```
5.645.3.6  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> std::set< _Key,
              _Compare, _Alloc >::set ( set< _Key, _Compare, _Alloc > && __x
              ) [inline]
```

Set move constructor

Parameters

x A set of identical element and allocator types.

The newly-created set contains the exact contents of *x*. The contents of *x* are a valid, but unspecified set.

Definition at line 199 of file stl_set.h.

```
5.645.3.7  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>> std::set< _Key,
              _Compare, _Alloc >::set ( initializer_list< value_type > __l, const
              _Compare & __comp = _Compare(), const allocator_type & __a =
              allocator_type() ) [inline]
```

Builds a set from an [initializer_list](#).

Parameters

l An [initializer_list](#).

comp A comparison functor.

a An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *l.size()*).

Definition at line 212 of file `stl_set.h`.

5.645.4 Member Function Documentation

5.645.4.1 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::begin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 292 of file `stl_set.h`.

5.645.4.2 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::cbegin () const [inline]`

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 329 of file `stl_set.h`.

5.645.4.3 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::cend () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 338 of file `stl_set.h`.

5.645.4.4 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::set< _Key,
_Compare, _Alloc >::clear () [inline]`

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 552 of file stl_set.h.

5.645.4.5 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::set<
_Key, _Compare, _Alloc >::count (const key_type & __x) const
[inline]`

Finds the number of elements.

Parameters

x Element to located.

Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 566 of file stl_set.h.

5.645.4.6 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<
_Key, _Compare, _Alloc >::crbegin () const [inline]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 347 of file stl_set.h.

5.645.4.7 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<
_Key, _Compare, _Alloc >::crend () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 356 of file stl_set.h.

5.645.4.8 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> bool std::set<_Key,
_Compare, _Alloc>::empty () const [inline]`

Returns true if the set is empty.

Definition at line 362 of file `stl_set.h`.

5.645.4.9 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 301 of file `stl_set.h`.

5.645.4.10 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::pair<iterator,
iterator> std::set<_Key, _Compare, _Alloc>::equal_range (const
key_type & __x) [inline]`

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 646 of file `stl_set.h`.

```
5.645.4.11 template<typename _Key, typename _Compare =
std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<const_iterator, const_iterator> std::set< _Key,
_Compare, _Alloc >::equal_range ( const key_type & __x ) const
[inline]
```

Finds a subsequence matching given key.

Parameters

x Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 650 of file stl_set.h.

```
5.645.4.12 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,
_Compare, _Alloc >::erase ( iterator __position ) [inline]
```

Erases an element from a set.

Parameters

position An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, [end\(\)](#) is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 478 of file stl_set.h.

5.645.4.13 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::set<_Key,
_Compare, _Alloc >::erase (const key_type & __x) [inline]`

Erases elements according to the provided key.

Parameters

x Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 507 of file `stl_set.h`.

5.645.4.14 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc >::erase (iterator __first, iterator __last)
[inline]`

Erases a `[first,last)` range of elements from a set.

Parameters

first Iterator pointing to the start of the range to be erased.

last Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 526 of file `stl_set.h`.

5.645.4.15 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc >::find (const key_type & __x) [inline]`

Tries to locate an element in a set.

Parameters

x Element to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 584 of file `stl_set.h`.

```
5.645.4.16 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> const_iterator std::set<
_Key, _Compare, _Alloc >::find ( const key_type & __x ) const
[inline]
```

Tries to locate an element in a set.

Parameters

x Element to be located.

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 588 of file `stl_set.h`.

```
5.645.4.17 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> allocator_type std::set<
_Key, _Compare, _Alloc >::get_allocator ( ) const [inline]
```

Returns the allocator object with which the set was constructed.

Definition at line 283 of file `stl_set.h`.

```
5.645.4.18 template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> std::pair<iterator,
bool> std::set< _Key, _Compare, _Alloc >::insert ( const
value_type & __x ) [inline]
```

Attempts to insert an element into the set.

Parameters

x Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 405 of file `stl_set.h`.

5.645.4.19 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc>::insert (iterator __position, const value_type
& __x) [inline]`

Attempts to insert an element into the set.

Parameters

position An iterator that serves as a hint as to where the element should be inserted.

x Element to be inserted.

Returns

An iterator that points to the element with key of *x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 432 of file `stl_set.h`.

5.645.4.20 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> template<typename
_InputIterator > void std::set< _Key, _Compare, _Alloc >::insert (
_InputIterator __first, _InputIterator __last) [inline]`

A template function that attempts to insert a range of elements.

Parameters

first Iterator pointing to the start of the range to be inserted.

last Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 446 of file stl_set.h.

5.645.4.21 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::set< _Key,
_Compare, _Alloc >::insert (initializer_list< value_type > __l)
[inline]`

Attempts to insert a list of elements into the set.

Parameters

list A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 458 of file stl_set.h.

Referenced by std::set< _StateIdT >::insert().

5.645.4.22 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> key_compare std::set<
_Key, _Compare, _Alloc >::key_comp () const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 275 of file stl_set.h.

5.645.4.23 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set< _Key,
_Compare, _Alloc >::lower_bound (const key_type & __x)
[inline]`

Finds the beginning of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 605 of file `stl_set.h`.

```
5.645.4.24  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>>> const_iterator std::set<
              _Key, _Compare, _Alloc >::lower_bound ( const key_type & __x )
              const [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

x Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or [end\(\)](#).

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or [end\(\)](#) if no such element exists.

Definition at line 609 of file `stl_set.h`.

```
5.645.4.25  template<typename _Key, typename _Compare = std::less<_Key>,
              typename _Alloc = std::allocator<_Key>>> size_type std::set<
              _Key, _Compare, _Alloc >::max_size ( ) const [inline]
```

Returns the maximum size of the set.

Definition at line 372 of file `stl_set.h`.

5.645.4.26 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> set& std::set< _Key,
_Compare, _Alloc >::operator= (set< _Key, _Compare, _Alloc >
&& __x) [inline]`

Set move assignment operator.

Parameters

x A set of identical element and allocator types.

The contents of *x* are moved into this set (without copying). *x* is a valid, but unspecified set.

Definition at line 242 of file stl_set.h.

5.645.4.27 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> set& std::set< _Key,
_Compare, _Alloc >::operator= (initializer_list< value_type > __l
) [inline]`

Set list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a set with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 263 of file stl_set.h.

5.645.4.28 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> set& std::set< _Key,
_Compare, _Alloc >::operator= (const set< _Key, _Compare,
_Alloc > & __x) [inline]`

Set assignment operator.

Parameters

x A set of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 227 of file stl_set.h.

5.645.4.29 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::set<_Key, _Compare, _Alloc>::rbegin () const [inline]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 310 of file `stl_set.h`.

5.645.4.30 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> reverse_iterator
std::set<_Key, _Compare, _Alloc>::rend () const [inline]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 319 of file `stl_set.h`.

5.645.4.31 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> size_type std::set<
_Key, _Compare, _Alloc>::size () const [inline]`

Returns the size of the set.

Definition at line 367 of file `stl_set.h`.

5.645.4.32 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> void std::set<_Key,
_Compare, _Alloc>::swap (set<_Key, _Compare, _Alloc> & __x
) [inline]`

Swaps data with another set.

Parameters

x A set of the same element and allocator types.

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global [std::swap\(\)](#) function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 387 of file `stl_set.h`.

Referenced by `std::swap()`.

5.645.4.33 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> iterator std::set<_Key,
_Compare, _Alloc >::upper_bound (const key_type & __x)
[inline]`

Finds the end of a subsequence matching given key.

Parameters

`x` Key to be located.

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 621 of file `stl_set.h`.

5.645.4.34 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> const_iterator std::set<
_Key, _Compare, _Alloc >::upper_bound (const key_type & __x)
const [inline]`

Finds the end of a subsequence matching given key.

Parameters

`x` Key to be located.

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 625 of file `stl_set.h`.

5.645.4.35 `template<typename _Key, typename _Compare = std::less<_Key>,
typename _Alloc = std::allocator<_Key>> value_compare std::set<
_Key, _Compare, _Alloc >::value_comp () const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 279 of file `stl_set.h`.

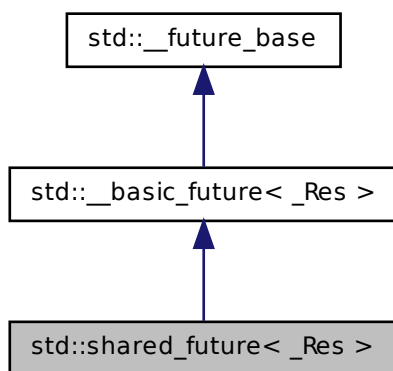
The documentation for this class was generated from the following file:

- [stl_set.h](#)

5.646 `std::shared_future<_Res>` Class Template Reference

Primary template for `shared_future`.

Inheritance diagram for `std::shared_future<_Res>`:



Public Member Functions

- `shared_future` (const `shared_future` &__sf)
- `shared_future` (`shared_future` &&__sf)
- `shared_future` (`future`<_Res> &&__uf)
- const _Res & `get` ()
- `shared_future` & `operator=` (const `shared_future` &__sf)
- `shared_future` & `operator=` (`shared_future` &&__sf)
- bool `valid` () const
- void `wait` () const
- template<typename _Rep, typename _Period>
bool `wait_for` (const `chrono::duration`<_Rep, _Period> &__rel) const
- template<typename _Clock, typename _Duration>
bool `wait_until` (const `chrono::time_point`<_Clock, _Duration> &__abs)
const

Static Public Member Functions

- `template<typename _Res, typename _Allocator >
static _Ptr< _Result_alloc< _Res, _Allocator > >::type _S_allocate_result
(const _Allocator &__a)`

Protected Types

- `typedef __future_base::_Result< _Res > & __result_type`
- `typedef shared_ptr< _State > __state_type`

Protected Member Functions

- `__result_type _M_get_result ()`
- `void _M_swap (__basic_future &__that)`

5.646.1 Detailed Description

`template<typename _Res> class std::shared_future< _Res >`

Primary template for [shared_future](#).

Definition at line 678 of file future.

5.646.2 Constructor & Destructor Documentation

5.646.2.1 `template<typename _Res > std::shared_future< _Res
>::shared_future (const shared_future< _Res > & __sf)
[inline]`

Copy constructor.

Definition at line 686 of file future.

5.646.2.2 `template<typename _Res > std::shared_future< _Res
>::shared_future (future< _Res > && __uf) [inline]`

Construct from a future rvalue.

Definition at line 689 of file future.

5.646.2.3 `template<typename _Res > std::shared_future< _Res
>::shared_future(shared_future< _Res > && __sf) [inline]`

Construct from a [shared_future](#) rvalue.

Definition at line 694 of file future.

5.646.3 Member Function Documentation

5.646.3.1 `template<typename _Res> __result_type std::__basic_future< _Res
>::__M_get_result() [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 508 of file future.

Referenced by `std::shared_future< _Res >::get()`, `std::future< void >::get()`, and `std::future< _Res >::get()`.

5.646.3.2 `template<typename _Res > const _Res& std::shared_future< _Res
>::get() [inline]`

Retrieving the value.

Definition at line 712 of file future.

References `std::__basic_future< _Res >::__M_get_result()`.

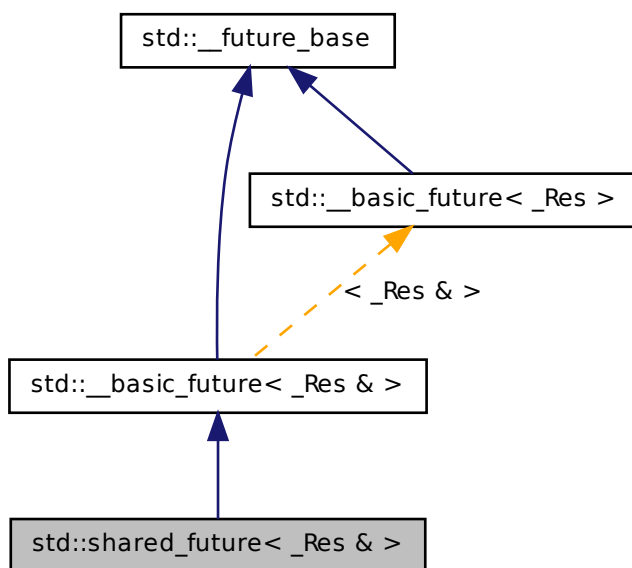
The documentation for this class was generated from the following file:

- [future](#)

5.647 `std::shared_future< _Res & >` Class Template Reference

Partial specialization for `shared_future<R&>`

Inheritance diagram for `std::shared_future< _Res & >`:



Public Member Functions

- `shared_future` (const `shared_future` &__sf)
- `shared_future` (`shared_future` &&__sf)
- `shared_future` (`future< _Res & >` &&__uf)
- `_Res` & `get` ()
- `shared_future` & `operator=` (const `shared_future` &__sf)
- `shared_future` & `operator=` (`shared_future` &&__sf)
- bool `valid` () const
- void `wait` () const
- bool `wait_for` (const `chrono::duration< _Rep, _Period >` &__rel) const
- bool `wait_until` (const `chrono::time_point< _Clock, _Duration >` &__abs) const

Static Public Member Functions

- `template<typename _Res, typename _Allocator >
static _Ptr< _Result_alloc< _Res, _Allocator > >::type _S_allocate_result
(const _Allocator &__a)`

Protected Types

- `typedef __future_base::_Result< _Res & > & __result_type`
- `typedef shared_ptr< _State > __state_type`

Protected Member Functions

- `__result_type _M_get_result ()`
- `void _M_swap (__basic_future &__that)`

5.647.1 Detailed Description

`template<typename _Res> class std::shared_future< _Res & >`

Partial specialization for `shared_future<R&>`

Definition at line 722 of file `future`.

5.647.2 Constructor & Destructor Documentation

5.647.2.1 `template<typename _Res > std::shared_future< _Res &
>::shared_future (const shared_future< _Res & > & __sf)
[inline]`

Copy constructor.

Definition at line 730 of file `future`.

5.647.2.2 `template<typename _Res > std::shared_future< _Res &
>::shared_future (future< _Res & > && __uf) [inline]`

Construct from a future rvalue.

Definition at line 733 of file `future`.

5.647.2.3 `template<typename _Res > std::shared_future< _Res &
>::shared_future (shared_future< _Res & > && __sf)
[inline]`

Construct from a [shared_future](#) rvalue.

Definition at line 738 of file `future`.

5.647.3 Member Function Documentation

5.647.3.1 `__result_type std::__basic_future< _Res & >::M_get_result ()
[inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 508 of file `future`.

References `std::rethrow_exception()`.

5.647.3.2 `template<typename _Res > _Res& std::shared_future< _Res &
>::get () [inline]`

Retrieving the value.

Definition at line 756 of file `future`.

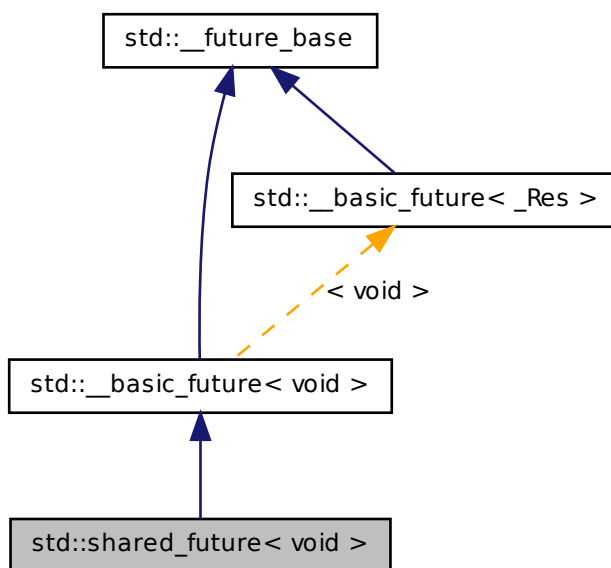
The documentation for this class was generated from the following file:

- [future](#)

5.648 `std::shared_future< void >` Class Template Reference

Explicit specialization for [shared_future<void>](#)

Inheritance diagram for `std::shared_future< void >`:



Public Member Functions

- `shared_future` (const `shared_future` &__sf)
- `shared_future` (`shared_future` &&__sf)
- `shared_future` (`future`< void > &&__uf)
- void **get** ()
- `shared_future` & **operator=** (const `shared_future` &__sf)
- `shared_future` & **operator=** (`shared_future` &&__sf)
- bool **valid** () const
- void **wait** () const
- bool **wait_for** (const `chrono::duration`< _Rep, _Period > &__rel) const
- bool **wait_until** (const `chrono::time_point`< _Clock, _Duration > &__abs) const

Static Public Member Functions

- `template<typename _Res, typename _Allocator >
static _Ptr< _Result_alloc< _Res, _Allocator > >::type _S_allocate_result
(const _Allocator &__a)`

Protected Types

- `typedef __future_base::_Result< void > & __result_type`
- `typedef shared_ptr< _State > __state_type`

Protected Member Functions

- `__result_type _M_get_result ()`
- `void _M_swap (__basic_future &__that)`

5.648.1 Detailed Description

`template<> class std::shared_future< void >`

Explicit specialization for [shared_future<void>](#)

Definition at line 761 of file future.

5.648.2 Constructor & Destructor Documentation

5.648.2.1 `std::shared_future< void >::shared_future (const shared_future< void > & __sf) [inline]`

Copy constructor.

Definition at line 769 of file future.

5.648.2.2 `std::shared_future< void >::shared_future (future< void > && __uf) [inline]`

Construct from a future rvalue.

Definition at line 772 of file future.

5.648.2.3 `std::shared_future< void >::shared_future (shared_future< void > && __sf) [inline]`

Construct from a [shared_future](#) rvalue.

Definition at line 777 of file future.

5.648.3 Member Function Documentation

5.648.3.1 `__result_type std::__basic_future< void >::M_get_result () [inline, protected, inherited]`

Wait for the state to be ready and rethrow any stored exception.

Definition at line 508 of file future.

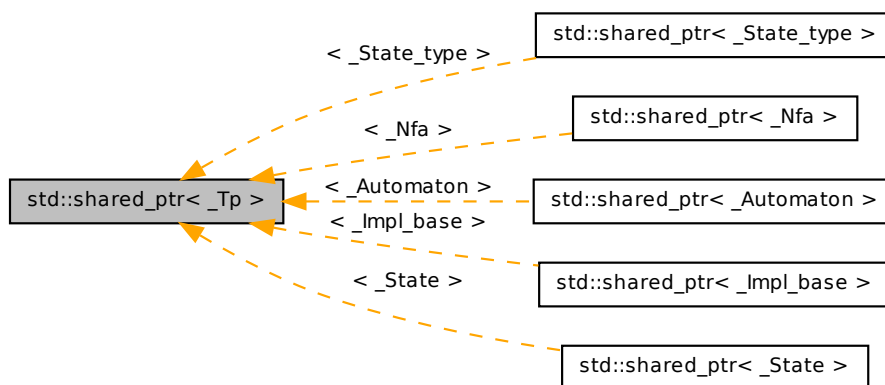
The documentation for this class was generated from the following file:

- [future](#)

5.649 `std::shared_ptr< _Tp >` Class Template Reference

A smart pointer with reference-counted copy semantics.

Inheritance diagram for `std::shared_ptr< _Tp >`:



Public Types

- `typedef _Tp element_type`

Public Member Functions

- `shared_ptr()`
- `template<typename _Tp1 > shared_ptr(_Tp1 *__p)`
- `template<typename _Deleter > shared_ptr(nullptr_t __p, _Deleter __d)`
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> shared_ptr(const shared_ptr<_Tp1 > &__r)`
- `shared_ptr(shared_ptr &&__r)`
- `template<typename _Tp1, typename _Deleter, typename _Alloc > shared_ptr(_Tp1 *__p, _Deleter __d, const _Alloc &__a)`
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> shared_ptr(shared_ptr<_Tp1 > &&__r)`
- `template<typename _Tp1 > shared_ptr(const weak_ptr<_Tp1 > &__r)`
- `template<typename _Tp1, typename _Deleter > shared_ptr(_Tp1 *__p, _Deleter __d)`
- `template<typename _Deleter, typename _Alloc > shared_ptr(nullptr_t __p, _Deleter __d, const _Alloc &__a)`
- `template<typename _Tp1 > shared_ptr(std::auto_ptr<_Tp1 > &&__r)`
- `template<typename _Tp1, typename _Del > shared_ptr(std::unique_ptr<_Tp1, _Del > &&__r)`
- `template<typename _Tp1 > shared_ptr(const shared_ptr<_Tp1 > &__r, _Tp *__p)`
- `shared_ptr(nullptr_t __p)`
- `_Tp * get() const`
- `operator bool() const`
- `std::add_lvalue_reference<_Tp>::type operator*() const`
- `_Tp * operator->() const`
- `template<typename _Tp1 > shared_ptr & operator=(std::auto_ptr<_Tp1 > &&__r)`
- `shared_ptr & operator=(shared_ptr &&__r)`
- `template<class _Tp1 > shared_ptr & operator=(shared_ptr<_Tp1 > &&__r)`
- `template<typename _Tp1, typename _Del > shared_ptr & operator=(std::unique_ptr<_Tp1, _Del > &&__r)`

- `template<typename _Tp1 >`
`shared_ptr & operator= (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp1 >`
`bool owner_before (__shared_ptr< _Tp1, _Lp > const &__rhs) const`
- `template<typename _Tp1 >`
`bool owner_before (__weak_ptr< _Tp1, _Lp > const &__rhs) const`
- `template<typename _Tp1 >`
`void reset (_Tp1 *__p)`
- `template<typename _Tp1, typename _Deleter >`
`void reset (_Tp1 *__p, _Deleter __d)`
- `void reset ()`
- `template<typename _Tp1, typename _Deleter, typename _Alloc >`
`void reset (_Tp1 *__p, _Deleter __d, const _Alloc &__a)`
- `void swap (__shared_ptr< _Tp, _Lp > &__other)`
- `bool unique () const`
- `long use_count () const`

Friends

- `template<typename _Tp1, _Lock_policy _Lp1, typename _Alloc, typename... _Args>`
`__shared_ptr< _Tp1, _Lp1 > __allocate_shared (_Alloc __a, _Args &&...__args)`
- `template<typename _Tp1, typename _Alloc, typename... _Args>`
`shared_ptr< _Tp1 > allocate_shared (_Alloc __a, _Args &&...__args)`

5.649.1 Detailed Description

`template<typename _Tp> class std::shared_ptr< _Tp >`

A smart pointer with reference-counted copy semantics. The object pointed to is deleted when the last `shared_ptr` pointing to it is destroyed or reset.

Definition at line 91 of file `shared_ptr.h`.

5.649.2 Constructor & Destructor Documentation

5.649.2.1 `template<typename _Tp> std::shared_ptr< _Tp >::shared_ptr ()`
`[inline]`

Construct an empty `shared_ptr`.

Postcondition

`use_count()==0 && get()==0`

Definition at line 98 of file shared_ptr.h.

5.649.2.2 `template<typename _Tp> template<typename _Tp1 >
std::shared_ptr< _Tp >::shared_ptr (_Tp1 * __p) [inline,
explicit]`

Construct a shared_ptr that owns the pointer __p.

Parameters

__p A pointer that is convertible to element_type*.

Postcondition

`use_count() == 1 && get() == __p`

Exceptions

std::bad_alloc, in which case `delete __p` is called.

Definition at line 107 of file shared_ptr.h.

5.649.2.3 `template<typename _Tp> template<typename _Tp1 , typename
_Deleter > std::shared_ptr< _Tp >::shared_ptr (_Tp1 * __p,
_Deleter __d) [inline]`

Construct a shared_ptr that owns the pointer __p and the deleter __d.

Parameters

__p A pointer.

__d A deleter.

Postcondition

`use_count() == 1 && get() == __p`

Exceptions

std::bad_alloc, in which case `__d(__p)` is called.

Requirements: _Deleter's copy constructor and destructor must not throw

__shared_ptr will release __p by calling __d(__p)

Definition at line 123 of file shared_ptr.h.

5.649.2.4 `template<typename _Tp> template<typename _Deleter >
std::shared_ptr< _Tp >::shared_ptr (nullptr_t __p, _Deleter __d
) [inline]`

Construct a shared_ptr that owns a null pointer and the deleter __d.

Parameters

__p A null pointer constant.

__d A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc, in which case __d(__p) is called.

Requirements: _Deleter's copy constructor and destructor must not throw

The last owner will call __d(__p)

Definition at line 139 of file shared_ptr.h.

5.649.2.5 `template<typename _Tp> template<typename _Tp1 , typename
_Deleter , typename _Alloc > std::shared_ptr< _Tp >::shared_ptr (
_Tp1 * __p, _Deleter __d, const _Alloc & __a) [inline]`

Construct a shared_ptr that owns the pointer __p and the deleter __d.

Parameters

__p A pointer.

__d A deleter.

__a An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc, in which case __d(__p) is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.

__shared_ptr will release __p by calling __d(__p)

Definition at line 158 of file shared_ptr.h.

5.649.2.6 `template<typename _Tp> template<typename _Deleter, typename
_Alloc> std::shared_ptr<_Tp>::shared_ptr (nullptr_t __p,
_Deleter __d, const _Alloc & __a) [inline]`

Construct a shared_ptr that owns a null pointer and the deleter __d.

Parameters

__p A null pointer constant.

__d A deleter.

__a An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

[std::bad_alloc](#), in which case __d(__p) is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.

The last owner will call __d(__p)

Definition at line 177 of file shared_ptr.h.

5.649.2.7 `template<typename _Tp> template<typename _Tp1>
std::shared_ptr<_Tp>::shared_ptr (const shared_ptr<_Tp1> &
__r, _Tp* __p) [inline]`

Constructs a shared_ptr instance that stores __p and shares ownership with __r.

Parameters

__r A shared_ptr.

__p A pointer that will remain valid while *__r is valid.

Postcondition

get() == __p && use_count() == __r.use_count()

This can be used to construct a [shared_ptr](#) to a sub-object of an object managed by an existing [shared_ptr](#).

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 199 of file shared_ptr.h.

5.649.2.8 `template<typename _Tp> template<typename _Tp1, typename
= typename std::enable_if<std::is_convertible<_Tp1*,
_Tp*>::value>::type> std::shared_ptr<_Tp>::shared_ptr (const
shared_ptr<_Tp1> & __r) [inline]`

If `__r` is empty, constructs an empty `shared_ptr`; otherwise construct a `shared_ptr` that shares ownership with `__r`.

Parameters

`__r` A `shared_ptr`.

Postcondition

`get() == __r.get() && use_count() == __r.use_count()`

Definition at line 211 of file `shared_ptr.h`.

5.649.2.9 `template<typename _Tp> std::shared_ptr<_Tp>::shared_ptr (
shared_ptr<_Tp> && __r) [inline]`

Move-constructs a `shared_ptr` instance from `__r`.

Parameters

`__r` A `shared_ptr` rvalue.

Postcondition

*this contains the old value of `__r`, `__r` is empty.

Definition at line 219 of file `shared_ptr.h`.

5.649.2.10 `template<typename _Tp> template<typename _Tp1, typename
= typename std::enable_if<std::is_convertible<_Tp1*,
_Tp*>::value>::type> std::shared_ptr<_Tp>::shared_ptr (
shared_ptr<_Tp1> && __r) [inline]`

Move-constructs a `shared_ptr` instance from `__r`.

Parameters

`__r` A `shared_ptr` rvalue.

Postcondition

*this contains the old value of `__r`, `__r` is empty.

Definition at line 229 of file `shared_ptr.h`.

5.649.2.11 `template<typename _Tp> template<typename _Tp1 >
std::shared_ptr<_Tp>::shared_ptr (const weak_ptr<_Tp1> &
__r) [inline, explicit]`

Constructs a `shared_ptr` that shares ownership with `__r` and stores a copy of the pointer stored in `__r`.

Parameters

`__r` A [weak_ptr](#).

Postcondition

`use_count() == __r.use_count()`

Exceptions

bad_weak_ptr when `__r.expired()`, in which case the constructor has no effect.

Definition at line 241 of file `shared_ptr.h`.

5.649.2.12 `template<typename _Tp> std::shared_ptr<_Tp>::shared_ptr (
nullptr_t __p) [inline]`

Construct an empty `shared_ptr`.

Parameters

`__p` A null pointer constant.

Postcondition

`use_count() == 0 && get() == nullptr`

Definition at line 259 of file `shared_ptr.h`.

5.649.3 Friends And Related Function Documentation

5.649.3.1 `template<typename _Tp> template<typename _Tp1 , typename
_Alloc , typename... _Args> shared_ptr<_Tp1> allocate_shared (
_Alloc __a, _Args &&... __args) [friend]`

Create an object that is owned by a [shared_ptr](#).

Parameters

`__a` An allocator.

`__args` Arguments for the `_Tp` object's constructor.

Returns

A [shared_ptr](#) that owns the newly created object.

Exceptions

An exception thrown from `_Alloc::allocate` or from the constructor of `_Tp`.

A copy of `__a` will be used to allocate memory for the [shared_ptr](#) and the new object.

Definition at line 520 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared_ptr.h](#)

5.650 `std::shuffle_order_engine<_RandomNumberEngine, __k >` Class Template Reference

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Public Types

- typedef `_RandomNumberEngine::result_type` [result_type](#)

Public Member Functions

- [shuffle_order_engine](#) ()
- [shuffle_order_engine](#) (const `_RandomNumberEngine` &__rne)
- [shuffle_order_engine](#) ([result_type](#) __s)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value>::type>`
[shuffle_order_engine](#) (`_Sseq` &__q)
- [shuffle_order_engine](#) (`_RandomNumberEngine` &&__rne)
- const `_RandomNumberEngine` & [base](#) () const
- void [discard](#) (unsigned long long __z)
- [result_type](#) [max](#) () const
- [result_type](#) [min](#) () const

- [result_type](#) [operator\(\)](#) ()
- void [seed](#) ([result_type](#) __s)
- template<typename _Sseq >
void [seed](#) (_Sseq &__q)
- void [seed](#) ()

Static Public Attributes

- static const size_t [table_size](#)

Friends

- template<typename _RandomNumberEngine1 , size_t __k1, typename _CharT , typename _Traits
>
[std::basic_ostream](#)< _CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)< _CharT, _Traits > &, const [std::shuffle_order_engine](#)< _RandomNumberEngine1, __k1 > &)
- bool [operator==](#) (const [shuffle_order_engine](#) &__lhs, const [shuffle_order_engine](#) &__rhs)
- template<typename _RandomNumberEngine1 , size_t __k1, typename _CharT , typename _Traits
>
[std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _CharT, _Traits > &, [std::shuffle_order_engine](#)< _RandomNumberEngine1, __k1 > &)

5.650.1 Detailed Description

template<typename _RandomNumberEngine, size_t __k> class std::shuffle_order_engine< _RandomNumberEngine, __k >

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits __w.

Definition at line 1237 of file random.h.

5.650.2 Member Typedef Documentation

**5.650.2.1 template<typename _RandomNumberEngine, size_t __k> typedef
_RandomNumberEngine::result_type std::shuffle_order_engine<
_RandomNumberEngine, __k >::result_type**

The type of the generated random value.

Definition at line 1244 of file random.h.

5.650.3 Constructor & Destructor Documentation

5.650.3.1 `template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k
>::shuffle_order_engine () [inline]`

Constructs a default shuffle_order_engine engine.

The underlying engine is default constructed as well.

Definition at line 1253 of file random.h.

5.650.3.2 `template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k
>::shuffle_order_engine (const _RandomNumberEngine & __rne)
[inline, explicit]`

Copy constructs a shuffle_order_engine engine.

Copies an existing base class random number generator.

Parameters

rng An existing (base class) engine object.

Definition at line 1264 of file random.h.

5.650.3.3 `template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k
>::shuffle_order_engine (_RandomNumberEngine && __rne)
[inline, explicit]`

Move constructs a shuffle_order_engine engine.

Copies an existing base class random number generator.

Parameters

rng An existing (base class) engine object.

Definition at line 1275 of file random.h.

5.650.3.4 `template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k
>::shuffle_order_engine (result_type __s) [inline,
explicit]`

Seed constructs a shuffle_order_engine engine.

Constructs the underlying generator engine seeded with __s.

Parameters

__s A seed value for the base class engine.

Definition at line 1286 of file random.h.

```
5.650.3.5  template<typename _RandomNumberEngine, size_t  
            __k> template<typename _Sseq , typename = typename  
            std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value  
            && !std::is_same<_Sseq, _RandomNumberEngine>::value>  
            ::type> std::shuffle_order_engine< _RandomNumberEngine, __k  
            >::shuffle_order_engine( _Sseq & __q ) [inline, explicit]
```

Generator construct a shuffle_order_engine engine.

Parameters

__q A seed sequence.

Definition at line 1300 of file random.h.

5.650.4 Member Function Documentation

```
5.650.4.1  template<typename _RandomNumberEngine, size_t __k>  
            const _RandomNumberEngine& std::shuffle_order_engine<  
            _RandomNumberEngine, __k >::base( ) const [inline]
```

Gets a const reference to the underlying generator engine object.

Definition at line 1343 of file random.h.

```
5.650.4.2  template<typename _RandomNumberEngine, size_t __k> void  
            std::shuffle_order_engine< _RandomNumberEngine, __k >::discard  
            ( unsigned long long __z ) [inline]
```

Discard a sequence of random numbers.

Todo

Look for a faster way to do discard.

Definition at line 1370 of file random.h.

5.650.4.3 `template<typename _RandomNumberEngine, size_t __k>
result_type std::shuffle_order_engine< _RandomNumberEngine, __k
>::max () const [inline]`

Gets the maximum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 1361 of file random.h.

5.650.4.4 `template<typename _RandomNumberEngine, size_t __k>
result_type std::shuffle_order_engine< _RandomNumberEngine, __k
>::min () const [inline]`

Gets the minimum value in the generated random number range.

Todo

This should be constexpr.

Definition at line 1352 of file random.h.

5.650.4.5 `template<typename _RandomNumberEngine , size_t __k>
shuffle_order_engine< _RandomNumberEngine, __k >::result_type
std::shuffle_order_engine< _RandomNumberEngine, __k
>::operator() ()`

Gets the next value in the generated random number sequence.

Definition at line 770 of file random.tcc.

5.650.4.6 `template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq > void std::shuffle_order_engine<
_RandomNumberEngine, __k >::seed (_Sseq & __q) [inline]`

Reseeds the shuffle_order_engine object with the given seed sequence.

Parameters

`__q` A seed generator function.

Definition at line 1333 of file random.h.

5.650.4.7 `template<typename _RandomNumberEngine, size_t __k> void
std::shuffle_order_engine<_RandomNumberEngine, __k>::seed (
result_type __s) [inline]`

Reseeds the `shuffle_order_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1320 of file `random.h`.

5.650.4.8 `template<typename _RandomNumberEngine, size_t __k> void
std::shuffle_order_engine<_RandomNumberEngine, __k>::seed (
) [inline]`

Reseeds the `shuffle_order_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1309 of file `random.h`.

5.650.5 Friends And Related Function Documentation

5.650.5.1 `template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1,
typename _CharT, typename _Traits > std::basic_ostream<_CharT,
_Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & ,
const std::shuffle_order_engine<_RandomNumberEngine1, __k1> &) [friend]`

Inserts the current state of a `shuffle_order_engine` random number generator engine `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A `shuffle_order_engine` random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.650.5.2 `template<typename _RandomNumberEngine, size_t __k> bool
operator==(const shuffle_order_engine< _RandomNumberEngine,
__k > & __lhs, const shuffle_order_engine<
_RandomNumberEngine, __k > & __rhs) [friend]`

Compares two `shuffle_order_engine` random number generator objects of the same type for equality.

Parameters

`__lhs` A `shuffle_order_engine` random number generator object.

`__rhs` Another `shuffle_order_engine` random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1394 of file `random.h`.

5.650.5.3 `template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1,
typename _CharT, typename _Traits > std::basic_istream<_CharT,
_Traits>& operator>> (std::basic_istream<_CharT, _Traits > & ,
std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &)
[friend]`

Extracts the current state of a `% subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `shuffle_order_engine` random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.651 std::slice Class Reference

Class defining one-dimensional subset of an array.

Public Member Functions

- [slice](#) ()
- [slice](#) (size_t, size_t, size_t)
- size_t [size](#) () const
- size_t [start](#) () const
- size_t [stride](#) () const

5.651.1 Detailed Description

Class defining one-dimensional subset of an array. The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 58 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice_array.h](#)

5.652 `std::slice_array< _Tp >` Class Template Reference

Reference to one-dimensional subset of an array.

Public Types

- typedef `_Tp` **value_type**

Public Member Functions

- [slice_array](#) (const [slice_array](#) &)
- template<class `_Dom` >
void **operator%=** (const `_Expr`< `_Dom`, `_Tp` > &) const
- void [operator%=](#) (const [valarray](#)< `_Tp` > &) const
- void [operator&=](#) (const [valarray](#)< `_Tp` > &) const

- `template<class _Dom >`
`void operator&= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void operator*= (const _Expr< _Dom, _Tp > &) const`
- `void operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator+= (const _Expr< _Dom, _Tp > &) const`
- `void operator+= (const valarray< _Tp > &) const`
- `void operator-= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator-= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void operator/= (const _Expr< _Dom, _Tp > &) const`
- `void operator/= (const valarray< _Tp > &) const`
- `void operator<<= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator<<= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const _Tp &) const`
- `slice_array & operator= (const slice_array &)`
- `template<class _Dom >`
`void operator= (const _Expr< _Dom, _Tp > &) const`
- `void operator= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator^= (const _Expr< _Dom, _Tp > &) const`
- `void operator^= (const valarray< _Tp > &) const`
- `void operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void operator|= (const _Expr< _Dom, _Tp > &) const`

Friends

- `class valarray< _Tp >`

5.652.1 Detailed Description

`template<typename _Tp> class std::slice_array< _Tp >`

Reference to one-dimensional subset of an array. A [slice_array](#) is a reference to the actual elements of an array specified by a slice. The way to get a [slice_array](#) is to call

operator[](slice) on a valarray. The returned [slice_array](#) then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this [slice_array](#) refers to.

Parameters

Tp Element type.

Definition at line 122 of file slice_array.h.

5.652.2 Member Function Documentation

5.652.2.1 `template<typename _Tp> void std::slice_array< _Tp >::operator%=(const valarray< _Tp > &) const`

Modulo slice elements by corresponding elements of *v*.

5.652.2.2 `template<typename _Tp> void std::slice_array< _Tp >::operator&=(const valarray< _Tp > &) const`

Logical and slice elements with corresponding elements of *v*.

5.652.2.3 `template<typename _Tp> void std::slice_array< _Tp >::operator*=(const valarray< _Tp > &) const`

Multiply slice elements by corresponding elements of *v*.

5.652.2.4 `template<typename _Tp> void std::slice_array< _Tp >::operator+=(const valarray< _Tp > &) const`

Add corresponding elements of *v* to slice elements.

5.652.2.5 `template<typename _Tp> void std::slice_array< _Tp >::operator-=(const valarray< _Tp > &) const`

Subtract corresponding elements of *v* from slice elements.

5.652.2.6 `template<typename _Tp> void std::slice_array< _Tp >::operator/=(const valarray< _Tp > &) const`

Divide slice elements by corresponding elements of *v*.

5.652.2.7 `template<typename _Tp> void std::slice_array< _Tp
>::operator<<= (const valarray< _Tp > &) const`

Left shift slice elements by corresponding elements of *v*.

5.652.2.8 `template<typename _Tp> void std::slice_array< _Tp
>::operator>>= (const valarray< _Tp > &) const`

Right shift slice elements by corresponding elements of *v*.

5.652.2.9 `template<typename _Tp> void std::slice_array< _Tp >::operator^=
(const valarray< _Tp > &) const`

Logical xor slice elements with corresponding elements of *v*.

5.652.2.10 `template<typename _Tp> void std::slice_array< _Tp
>::operator|= (const valarray< _Tp > &) const`

Logical or slice elements with corresponding elements of *v*.

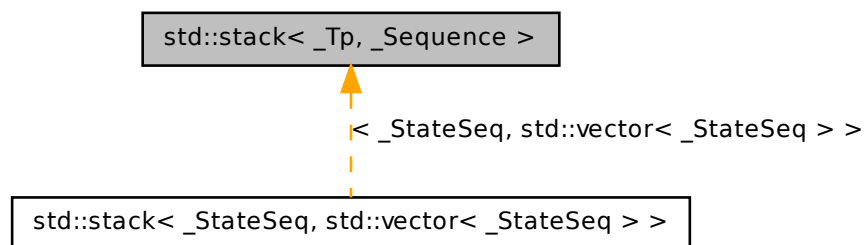
The documentation for this class was generated from the following file:

- [slice_array.h](#)

5.653 std::stack< _Tp, _Sequence > Class Template Reference

A standard container giving FILO behavior.

Inheritance diagram for std::stack< _Tp, _Sequence >:



Public Types

- typedef `_Sequence::const_reference` **const_reference**
- typedef `_Sequence` **container_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size_type**
- typedef `_Sequence::value_type` **value_type**

Public Member Functions

- `stack` (`const _Sequence &__c`)
- `stack` (`_Sequence &&__c=_Sequence()`)
- `template<typename... _Args>`
void **emplace** (`_Args &&...__args`)
- bool `empty` () const
- void `pop` ()
- void **push** (`value_type &&__x`)
- void `push` (`const value_type &__x`)
- `size_type` `size` () const
- void **swap** (`stack &__s`)
- `const_reference` `top` () const
- `reference` `top` ()

Protected Attributes

- `_Sequence c`

Friends

- `template<typename _Tp1, typename _Seq1 >
bool operator< (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)`
- `template<typename _Tp1, typename _Seq1 >
bool operator== (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)`

5.653.1 Detailed Description

`template<typename _Tp, typename _Sequence = deque<_Tp>> class std::stack< _Tp, _Sequence >`

A standard container giving FILO behavior. Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to [std::deque](#), but it can be any type that supports `back`, `push_back`, and `pop_front`, such as [std::list](#), [std::vector](#), or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 92 of file `stl_stack.h`.

5.653.2 Constructor & Destructor Documentation

5.653.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
std::stack< _Tp, _Sequence >::stack (const _Sequence & __c)
[inline, explicit]`

Default constructor creates no elements.

Definition at line 130 of file `stl_stack.h`.

5.653.3 Member Function Documentation

5.653.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::stack< _Tp, _Sequence >::empty () const [inline]`

Returns true if the stack is empty.

Definition at line 142 of file stl_stack.h.

5.653.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::pop () [inline]`

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before [pop\(\)](#) is called.

Definition at line 208 of file stl_stack.h.

5.653.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::push (const value_type & __x)
[inline]`

Add data to the top of the stack.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 182 of file stl_stack.h.

5.653.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::stack< _Tp, _Sequence >::size () const [inline]`

Returns the number of elements in the stack.

Definition at line 147 of file stl_stack.h.

5.653.3.5 `template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::stack<_Tp, _Sequence>::top () [inline]`

Returns a read/write reference to the data at the first element of the stack.

Definition at line 155 of file `stl_stack.h`.

5.653.3.6 `template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::stack<_Tp, _Sequence>::top () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 166 of file `stl_stack.h`.

The documentation for this class was generated from the following file:

- [stl_stack.h](#)

5.654 `std::student_t_distribution<_RealType>` Class Template Reference

A [student_t_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- `student_t_distribution` (`_RealType __n=_RealType(1)`)
- `student_t_distribution` (`const param_type &__p`)
- `result_type max` () const
- `result_type min` () const
- `_RealType n` () const
- `template<typename _UniformRandomNumberGenerator >
result_type operator()` (`_UniformRandomNumberGenerator &__urng`)

- template<typename _UniformRandomNumberGenerator >
 [result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const
 [param_type](#) &__p)
- void [param](#) (const [param_type](#) &__param)
- [param_type](#) [param](#) () const
- void [reset](#) ()

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
 [std::basic_ostream](#)< _CharT, _Traits > & [operator<<](#) ([std::basic_ostream](#)< _
 CharT, _Traits > &, const [std::student_t_distribution](#)< _RealType1 > &)
- template<typename _RealType1 >
 bool [operator==](#) (const [std::student_t_distribution](#)< _RealType1 > &__d1,
 const [std::student_t_distribution](#)< _RealType1 > &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
 [std::basic_istream](#)< _CharT, _Traits > & [operator>>](#) ([std::basic_istream](#)< _
 CharT, _Traits > &, [std::student_t_distribution](#)< _RealType1 > &)

5.654.1 Detailed Description

**template<typename _RealType = double> class std::student_t_distribution< _
RealType >**

A [student_t_distribution](#) random number distribution. The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3064 of file random.h.

5.654.2 Member Typedef Documentation

**5.654.2.1 template<typename _RealType = double> typedef _RealType
std::student_t_distribution< _RealType >::result_type**

The type of the range of the distribution.

Definition at line 3071 of file random.h.

5.654.3 Member Function Documentation

5.654.3.1 `template<typename _RealType = double> result_type
std::student_t_distribution< _RealType >::max () const
[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3147 of file random.h.

5.654.3.2 `template<typename _RealType = double> result_type
std::student_t_distribution< _RealType >::min () const
[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3140 of file random.h.

5.654.3.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::student_t_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3155 of file random.h.

References `std::sqrt()`.

5.654.3.4 `template<typename _RealType = double> param_type
std::student_t_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 3125 of file random.h.

5.654.3.5 `template<typename _RealType = double> void
std::student_t_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

__param The new parameter set of the distribution.

5.654 std::student_t_distribution< _RealType > Class Template Reference 3029

Definition at line 3133 of file random.h.

5.654.3.6 `template<typename _RealType = double> void
std::student_t_distribution< _RealType >::reset() [inline]`

Resets the distribution state.

Definition at line 3108 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`, and `std::normal_distribution< _RealType >::reset()`.

5.654.4 Friends And Related Function Documentation

5.654.4.1 `template<typename _RealType = double> template<typename
_RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<<
(std::basic_ostream< _CharT, _Traits > & , const
std::student_t_distribution< _RealType1 > &) [friend]`

Inserts a student_t_distribution random number distribution `__x` into the output stream `__os`.

Parameters

`__os` An output stream.

`__x` A student_t_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

5.654.4.2 `template<typename _RealType = double> template<typename
_RealType1 > bool operator==(const std::student_t_distribution<
_RealType1 > & __d1, const std::student_t_distribution<
_RealType1 > & __d2) [friend]`

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3177 of file random.h.

```

5.654.4.3 template<typename _RealType = double> template<typename
          _RealType1 , typename _CharT , typename _Traits >
          std::basic_istream<_CharT, _Traits>& operator>>
          ( std::basic_istream< _CharT, _Traits > & ,
            std::student_t_distribution< _RealType1 > & ) [friend]

```

Extracts a `student_t_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

`__is` An input stream.

`__x` A `student_t_distribution` random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

5.655 std::student_t_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef [student_t_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __n=_RealType(1))
- **_RealType n** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.655.1 Detailed Description

template<typename _RealType = double> struct std::student_t_distribution< _RealType >::param_type

Parameter type.

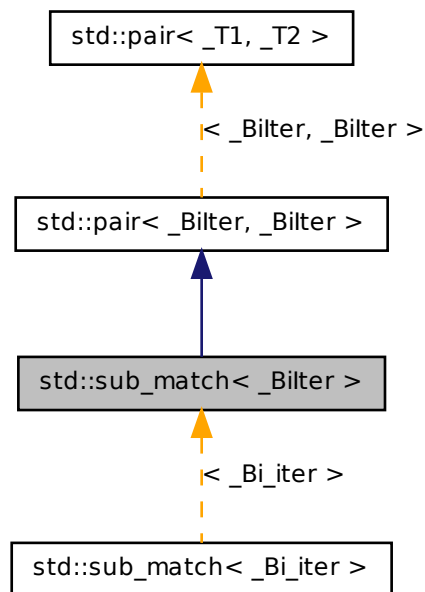
Definition at line 3073 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.656 std::sub_match< _Biliter > Class Template Reference

Inheritance diagram for std::sub_match< _Biliter >:



Public Types

- typedef iterator_traits< _Bilter >::difference_type **difference_type**
- typedef _Bilter [first_type](#)
- typedef _Bilter **iterator**
- typedef _Bilter [second_type](#)
- typedef [std::basic_string](#)< value_type > **string_type**
- typedef iterator_traits< _Bilter >::value_type **value_type**

Public Member Functions

- int [compare](#) (const [sub_match](#) &__s) const
- int [compare](#) (const [string_type](#) &__s) const
- int [compare](#) (const value_type *__s) const
- difference_type [length](#) () const
- [operator string_type](#) () const
- [string_type](#) str () const
- void [swap](#) ([pair](#) &__p)

Public Attributes

- _Bilter [first](#)
- bool **matched**
- _Bilter [second](#)

5.656.1 Detailed Description

template<typename _Bilter> class std::sub_match< _Bilter >

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with [std::basic_string](#) objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 756 of file regex.h.

5.656.2 Member Typedef Documentation

5.656.2.1 `typedef _Biliter std::pair< _Biliter , _Biliter >::first_type` [`inherited`]

`first_type` is the first bound type

Definition at line 86 of file `stl_pair.h`.

5.656.2.2 `typedef _Biliter std::pair< _Biliter , _Biliter >::second_type` [`inherited`]

`second_type` is the second bound type

Definition at line 87 of file `stl_pair.h`.

5.656.3 Member Function Documentation

5.656.3.1 `template<typename _Biliter> int std::sub_match< _Biliter >::compare (const sub_match< _Biliter > & __s) const` [`inline`]

Compares this and another matched sequence.

Parameters

`s` Another matched sequence to compare to this one.

Return values

`<0` this matched sequence will collate before `s`.

`=0` this matched sequence is equivalent to `s`.

`>0` this matched sequence will collate after `s`.

Definition at line 815 of file `regex.h`.

Referenced by `std::operator!=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

5.656.3.2 `template<typename _Biliter> int std::sub_match< _Biliter >::compare (const string_type & __s) const` [`inline`]

Compares this `sub_match` to a string.

Parameters

s A string to compare to this [sub_match](#).

Return values

<0 this matched sequence will collate before *s*.

$=0$ this matched sequence is equivalent to *s*.

<0 this matched sequence will collate after *s*.

Definition at line 828 of file regex.h.

5.656.3.3 `template<typename _Biter> int std::sub_match< _Biter
>::compare (const value_type * __s) const [inline]`

Compares this [sub_match](#) to a C-style string.

Parameters

s A C-style string to compare to this [sub_match](#).

Return values

<0 this matched sequence will collate before *s*.

$=0$ this matched sequence is equivalent to *s*.

<0 this matched sequence will collate after *s*.

Definition at line 841 of file regex.h.

5.656.3.4 `template<typename _Biter> difference_type std::sub_match<
_Biter >::length () const [inline]`

Gets the length of the matching sequence.

Definition at line 772 of file regex.h.

5.656.3.5 `template<typename _Biter> std::sub_match< _Biter >::operator
string_type () const [inline]`

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the [str\(\)](#) member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 785 of file regex.h.

5.656.3.6 `template<typename _BiIter> string_type std::sub_match<_BiIter>::str() const [inline]`

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

Definition at line 798 of file regex.h.

Referenced by `std::sub_match<_BiIter>::compare()`, `std::operator!=()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

5.656.4 Member Data Documentation

5.656.4.1 `_BiIter std::pair<_BiIter, _BiIter>::first [inherited]`

`first` is a copy of the first object

Definition at line 89 of file stl_pair.h.

Referenced by `std::sub_match<_BiIter>::length()`, `std::sub_match<_BiIter>::operator string_type()`, and `std::sub_match<_BiIter>::str()`.

5.656.4.2 `_BiIter std::pair<_BiIter, _BiIter>::second [inherited]`

`second` is a copy of the second object

Definition at line 90 of file stl_pair.h.

Referenced by `std::sub_match<_BiIter>::length()`, `std::sub_match<_BiIter>::operator string_type()`, and `std::sub_match<_BiIter>::str()`.

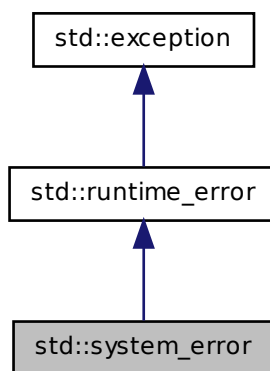
The documentation for this class was generated from the following file:

- [regex.h](#)

5.657 std::system_error Class Reference

Thrown to indicate error code of underlying system.

Inheritance diagram for std::system_error:



Public Member Functions

- `system_error` ([error_code](#) __ec=[error_code](#)())
- `system_error` ([error_code](#) __ec, const [string](#) &__what)
- `system_error` (int __v, const [error_category](#) &__ecat, const [string](#) &__what)
- `system_error` (int __v, const [error_category](#) &__ecat)
- const [error_code](#) & `code` () const throw ()
- virtual const char * `what` () const throw ()

5.657.1 Detailed Description

Thrown to indicate error code of underlying system.

Definition at line 307 of file `system_error`.

5.657.2 Member Function Documentation

5.657.2.1 virtual const char* std::runtime_error::what () const throw () [virtual, inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [system_error](#)

5.658 std::thread Class Reference

thread

Classes

- class [id](#)
[thread::id](#)

Public Types

- typedef [shared_ptr](#)< _Impl_base > [__shared_base_type](#)
- typedef __thread_t [native_handle_type](#)

Public Member Functions

- **thread** ([thread](#) &)
- **thread** ([thread](#) &&__t)
- template<typename _Callable , typename... _Args>
thread (_Callable &&__f, _Args &&...__args)
- **thread** (const [thread](#) &)
- void **detach** ()
- [thread::id](#) **get_id** () const
- void **join** ()
- bool **joinable** () const
- native_handle_type [native_handle](#) ()
- [thread](#) & **operator=** (const [thread](#) &)

- [thread](#) & **operator=** ([thread](#) &&__t)
- void **swap** ([thread](#) &__t)

Static Public Member Functions

- static unsigned int **hardware_concurrency** ()

5.658.1 Detailed Description

[thread](#)

Definition at line 60 of file [thread](#).

5.658.2 Member Function Documentation

5.658.2.1 [native_handle_type](#) [std::thread::native_handle](#) () [[inline](#)]

Precondition

[thread](#) is joinable

Definition at line 175 of file [thread](#).

The documentation for this class was generated from the following file:

- [thread](#)

5.659 [std::thread::id](#) Class Reference

[thread::id](#)

Public Member Functions

- **id** ([native_handle_type](#) __id)

Friends

- class **hash**< [thread::id](#) >
- bool **operator**< ([thread::id](#) __x, [thread::id](#) __y)
- template<class _CharT, class _Traits >
[basic_ostream](#)< _CharT, _Traits > & **operator**<< ([basic_ostream](#)< _CharT, _Traits > &__out, [thread::id](#) __id)

- bool **operator==** ([thread::id](#) __x, [thread::id](#) __y)
- class **thread**

5.659.1 Detailed Description

[thread::id](#)

Definition at line 68 of file thread.

The documentation for this class was generated from the following file:

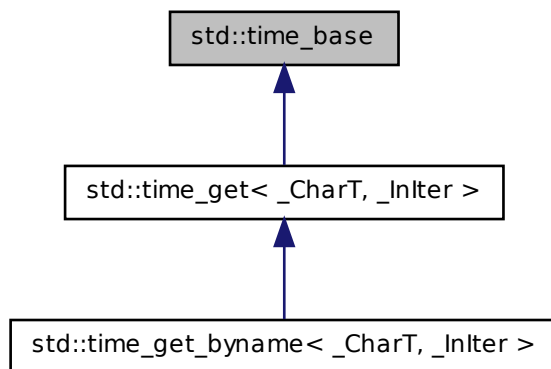
- [thread](#)

5.660 std::time_base Class Reference

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Inheritance diagram for std::time_base:



Public Types

- enum **dateorder** {

```
no_order, dmy, mdy, ymd,
ydm }
```

5.660.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

Definition at line 50 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

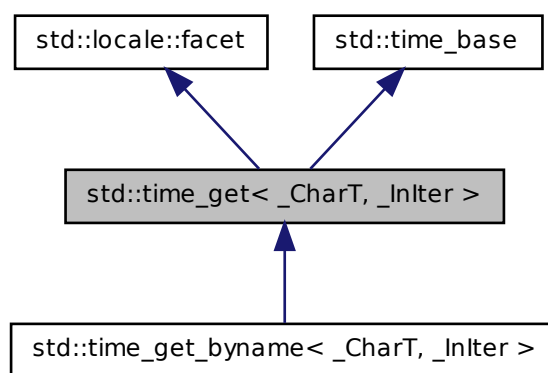
- [locale_facets_nonio.h](#)

5.661 std::time_get< _CharT, _InIter > Class Template Reference

Primary class template [time_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

Inheritance diagram for std::time_get< _CharT, _InIter >:



Public Types

- typedef [basic_string](#)< _CharT > [__string_type](#)
- enum [dateorder](#) {
[no_order](#), [dmy](#), [mdy](#), [ymd](#),
[ydm](#) }
- typedef _CharT [char_type](#)
- typedef _InIter [iter_type](#)

Public Member Functions

- [time_get](#) (size_t __refs=0)
- dateorder [date_order](#) () const
- [iter_type get_date](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- [iter_type get_monthname](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- [iter_type get_time](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- [iter_type get_weekday](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- [iter_type get_year](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~time_get](#) ()
- [iter_type _M_extract_name](#) ([iter_type](#) __beg, [iter_type](#) __end, int &__member, const _CharT **__names, size_t __indexlen, [ios_base](#) &__io, [ios_base::iostate](#) &__err) const
- [iter_type _M_extract_num](#) ([iter_type](#) __beg, [iter_type](#) __end, int &__member, int __min, int __max, size_t __len, [ios_base](#) &__io, [ios_base::iostate](#) &__err) const
- [iter_type _M_extract_via_format](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm, const _CharT *__format) const
- [iter_type _M_extract_wday_or_month](#) ([iter_type](#) __beg, [iter_type](#) __end, int &__member, const _CharT **__names, size_t __indexlen, [ios_base](#) &__io, [ios_base::iostate](#) &__err) const

- virtual dateorder [do_date_order](#) () const
- virtual [iter_type](#) [do_get_date](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- virtual [iter_type](#) [do_get_monthname](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- virtual [iter_type](#) [do_get_time](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- virtual [iter_type](#) [do_get_weekday](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const
- virtual [iter_type](#) [do_get_year](#) ([iter_type](#) __beg, [iter_type](#) __end, [ios_base](#) &__io, [ios_base::iostate](#) &__err, tm *__tm) const

Static Protected Member Functions

- static [__c_locale](#) [_S_clone_c_locale](#) ([__c_locale](#) &__cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) &__cloc, const char *__s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) &__cloc)
- static [__c_locale](#) [_S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static [__c_locale](#) [_S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const char *__s)

Friends

- class [locale::Impl](#)

5.661.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::time_get< _CharT, _InIter >`

Primary class template [time_get](#).

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators. The [time_get](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [time_get](#) facet.

Definition at line 363 of file [locale_facets_nonio.h](#).

5.661.2 Member Typedef Documentation

5.661.2.1 `template<typename _CharT, typename _InIter > typedef _CharT std::time_get< _CharT, _InIter >::char_type`

Public typedefs.

Reimplemented in [std::time_get_byname< _CharT, _InIter >](#).

Definition at line 369 of file locale_facets_nonio.h.

5.661.2.2 `template<typename _CharT, typename _InIter > typedef _InIter std::time_get< _CharT, _InIter >::iter_type`

Public typedefs.

Reimplemented in [std::time_get_byname< _CharT, _InIter >](#).

Definition at line 370 of file locale_facets_nonio.h.

5.661.3 Constructor & Destructor Documentation

5.661.3.1 `template<typename _CharT, typename _InIter > std::time_get< _CharT, _InIter >::time_get (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 385 of file locale_facets_nonio.h.

5.661.3.2 `template<typename _CharT, typename _InIter > virtual std::time_get< _CharT, _InIter >::~~time_get () [inline, protected, virtual]`

Destructor.

Definition at line 541 of file locale_facets_nonio.h.

5.661.4 Member Function Documentation

5.661.4.1 `template<typename _CharT, typename _InIter > dateorder
std::time_get< _CharT, _InIter >::date_order () const [inline]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format *x* given to `time_put::put()` only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

Returns

A member of `timebase::dateorder`.

Definition at line 402 of file `locale_facets_nonio.h`.

5.661.4.2 `template<typename _CharT, typename _InIter >
_GLIBCXX_END_LDBL_NAMESPACE time_base::dateorder
std::time_get< _CharT, _InIter >::do_date_order () const
[protected, virtual]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format *x* given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of `timebase::dateorder`.

Definition at line 618 of file `locale_facets_nonio.tcc`.

5.661.4.3 `template<typename _CharT, typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_date (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [protected, virtual]`

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

[get_date\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 1045 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

```
5.661.4.4 template<typename _CharT , typename _InIter > _InIter  
std::time_get< _CharT, _InIter >::do_get_monthname ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm.
This function is a hook for derived classes to change the value returned.

See also

[get_monthname\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 1090 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

5.661.4.5 `template<typename _CharT, typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_time (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [protected, virtual]`

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

See also

[get_time\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct *tm* to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 1028 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

5.661.4.6 `template<typename _CharT, typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_weekday (iter_type
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
tm * __tm) const [protected, virtual]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

See also

[get_weekday\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 1062 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

```
5.661.4.7 template<typename _CharT , typename _InIter > _InIter  
std::time_get< _CharT, _InIter >::do_get_year ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [protected, virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_year\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 1118 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

5.661.4.8 `template<typename _CharT, typename _InIter> iter_type
std::time_get<_CharT, _InIter>::get_date (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [inline]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling [time_get::do_get_date\(\)](#).

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, *err* |= [ios_base::failbit](#). If parsing reads all the characters, *err* |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct *tm* to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 451 of file `locale_facets_nonio.h`.

5.661.4.9 `template<typename _CharT, typename _InIter> iter_type
std::time_get<_CharT, _InIter>::get_monthname (iter_type
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
tm * __tm) const [inline]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling [time_get::do_get_monthname\(\)](#).

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, *err* |= [ios_base::failbit](#). If parsing reads all the characters, *err* |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 508 of file locale_facets_nonio.h.

```
5.661.4.10 template<typename _CharT , typename _InIter > iter_type  
std::time_get< _CharT, _InIter >::get_time ( iter_type __beg,  
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *  
__tm ) const [inline]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_time\(\)](#).

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 426 of file locale_facets_nonio.h.

5.661.4.11 `template<typename _CharT, typename _InIter> iter_type
std::time_get<_CharT, _InIter>::get_weekday (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [inline]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= `ios_base::failbit`. If parsing reads all the characters, err |= `ios_base::eofbit`.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 479 of file locale_facets_nonio.h.

5.661.4.12 `template<typename _CharT, typename _InIter> iter_type
std::time_get<_CharT, _InIter>::get_year (iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [inline]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= `ios_base::failbit`. If parsing reads all the characters, err |= `ios_base::eofbit`.

5.662 std::time_get_byname<_CharT, _InIter > Class Template Reference 3051

Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 534 of file locale_facets_nonio.h.

5.661.5 Member Data Documentation

5.661.5.1 template<typename _CharT, typename _InIter > locale::id std::time_get<_CharT, _InIter >::id [static]

Numpunct facet id.

Definition at line 375 of file locale_facets_nonio.h.

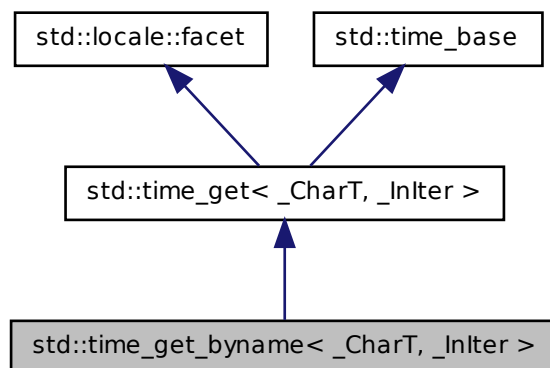
The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [locale_facets_nonio.tcc](#)

5.662 std::time_get_byname<_CharT, _InIter > Class Template Reference

class [time_get_byname](#) [22.2.5.2].

Inheritance diagram for `std::time_get_byname<_CharT, _InIter>`:



Public Types

- typedef `basic_string<_CharT>` `__string_type`
- typedef `_CharT` `char_type`
- enum `dateorder` {
 `no_order`, `dmy`, `mdy`, `ymd`,
 `ydm` }
- typedef `_InIter` `iter_type`

Public Member Functions

- `time_get_byname` (`const char *`, `size_t __refs=0`)
- `dateorder date_order` () const
- `iter_type get_date` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- `iter_type get_monthname` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- `iter_type get_time` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const
- `iter_type get_weekday` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) const

- `iter_type get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`

Static Public Attributes

- static `locale::id id`

Protected Member Functions

- `iter_type _M_extract_name (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type _M_extract_num (iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type _M_extract_via_format (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const`
- `iter_type _M_extract_wday_or_month (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- virtual `dateorder do_date_order () const`
- virtual `iter_type do_get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static `void _S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static `void _S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static `_GLIBCXX_CONST const char * _S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Friends

- class `locale::_Impl`

5.662.1 Detailed Description

`template<typename _CharT, typename _InIter> class std::time_get_byname<_CharT, _InIter >`

class [time_get_byname](#) [22.2.5.2].

Definition at line 681 of file `locale_facets_nonio.h`.

5.662.2 Member Typedef Documentation

5.662.2.1 `template<typename _CharT , typename _InIter > typedef _CharT std::time_get_byname< _CharT, _InIter >::char_type`

Public typedefs.

Reimplemented from [std::time_get< _CharT, _InIter >](#).

Definition at line 685 of file `locale_facets_nonio.h`.

5.662.2.2 `template<typename _CharT , typename _InIter > typedef _InIter std::time_get_byname< _CharT, _InIter >::iter_type`

Public typedefs.

Reimplemented from [std::time_get< _CharT, _InIter >](#).

Definition at line 686 of file `locale_facets_nonio.h`.

5.662.3 Member Function Documentation

5.662.3.1 `template<typename _CharT , typename _InIter > dateorder std::time_get< _CharT, _InIter >::date_order () const [inline, inherited]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format *x* given to [time_put::put\(\)](#) only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

5.662 `std::time_get_byname<_CharT, _InIter>` Class Template Reference 3055

NOTE: The library always returns noorder at the moment.

Returns

A member of `timebase::dateorder`.

Definition at line 402 of file `locale_facets_nonio.h`.

```
5.662.3.2  template<typename _CharT , typename _InIter >
           _GLIBCXX_END_LDBL_NAMESPACE time_base::dateorder
           std::time_get< _CharT, _InIter >::do_date_order ( ) const
           [protected, virtual, inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to [time_put::put\(\)](#) only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of `timebase::dateorder`.

Definition at line 618 of file `locale_facets_nonio.tcc`.

```
5.662.3.3  template<typename _CharT , typename _InIter > _InIter
           std::time_get< _CharT, _InIter >::do_get_date ( iter_type __beg,
           iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
           __tm ) const [protected, virtual, inherited]
```

Parse input date string.

This function parses a date according to the format `X` and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

[get_date\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 1045 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

```
5.662.3.4 template<typename _CharT , typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_monthname ( iter_type
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
tm * __tm ) const [protected, virtual, inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

[get_monthname\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 1090 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

```
5.662.3.5 template<typename _CharT , typename _InIter > _InIter
std::time_get< _CharT, _InIter >::do_get_time ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm ) const [protected, virtual, inherited]
```

Parse input time string.

5.662 std::time_get_byname<_CharT, _InIter> Class Template Reference 3057

This function parses a time according to the format *x* and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

See also

[get_time\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct *tm* to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 1028 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

```
5.662.3.6 template<typename _CharT, typename _InIter> _InIter  
std::time_get<_CharT, _InIter>::do_get_weekday ( iter_type  
__beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,  
tm * __tm ) const [protected, virtual, inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. This function is a hook for derived classes to change the value returned.

See also

[get_weekday\(\)](#) for details.

Parameters

beg Start of string to parse.
end End of string to parse.
io Source of the locale.
err Error flags to set.
tm Pointer to struct *tm* to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 1062 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

5.662.3.7 `template<typename _CharT, typename _InIter> _InIter
std::time_get<_CharT, _InIter>::do_get_year(iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [protected, virtual, inherited]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

[get_year\(\)](#) for details.

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct `tm` to fill in.

Returns

Iterator to first char beyond year.

Definition at line 1118 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

5.662.3.8 `template<typename _CharT, typename _InIter> iter_type
std::time_get<_CharT, _InIter>::get_date(iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm) const [inline, inherited]`

Parse input date string.

5.662 std::time_get_byname<_CharT, _InIter> Class Template Reference 3059

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling [time_get::do_get_date\(\)](#).

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct *tm* to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 451 of file locale_facets_nonio.h.

```
5.662.3.9  template<typename _CharT, typename _InIter> iter_type
           std::time_get<_CharT, _InIter>::get_monthname ( iter_type
           __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err,
           tm * __tm ) const [inline, inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling [time_get::do_get_monthname\(\)](#).

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 508 of file locale_facets_nonio.h.

```
5.662.3.10  template<typename _CharT , typename _InIter > iter_type
              std::time_get< _CharT, _InIter >::get_time ( iter_type __beg,
              iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
              __tm ) const [inline, inherited]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_time\(\)](#).

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

beg Start of string to parse.

end End of string to parse.

io Source of the locale.

err Error flags to set.

tm Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 426 of file locale_facets_nonio.h.

```
5.662.3.11  template<typename _CharT , typename _InIter > iter_type
              std::time_get< _CharT, _InIter >::get_weekday ( iter_type __beg,
              iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
              __tm ) const [inline, inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_weekday\(\)](#).

5.662 std::time_get_byname<_CharT, _InIter> Class Template Reference 3061

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 479 of file locale_facets_nonio.h.

```
5.662.3.12  template<typename _CharT, typename _InIter> iter_type
std::time_get<_CharT, _InIter>::get_year ( iter_type __beg,
iter_type __end, ios_base & __io, ios_base::iostate & __err, tm *
__tm ) const [inline, inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling [time_get::do_get_year\(\)](#).

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= [ios_base::failbit](#). If parsing reads all the characters, err |= [ios_base::eofbit](#).

Parameters

- beg* Start of string to parse.
- end* End of string to parse.
- io* Source of the locale.
- err* Error flags to set.
- tm* Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 534 of file `locale_facets_nonio.h`.

5.662.4 Member Data Documentation

5.662.4.1 `template<typename _CharT, typename _InIter > locale::id
std::time_get<_CharT, _InIter >::id [static, inherited]`

Numpunct facet id.

Definition at line 375 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

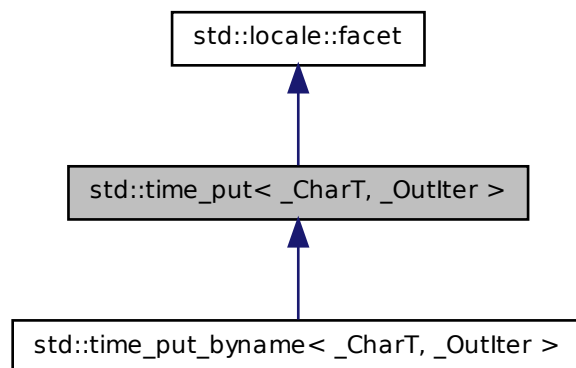
- [locale_facets_nonio.h](#)

5.663 `std::time_put< _CharT, _OutIter >` Class Template Reference

Primary class template [time_put](#).

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

Inheritance diagram for std::time_put< _CharT, _OutIter >:



Public Types

- typedef `_CharT` `char_type`
- typedef `_OutIter` `iter_type`

Public Member Functions

- `time_put` (`size_t __refs=0`)
- `iter_type put` (`iter_type __s`, `ios_base &__io`, `char_type __fill`, `const tm *__tm`, `char __format`, `char __mod=0`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__io`, `char_type __fill`, `const tm *__tm`, `const _CharT *__beg`, `const _CharT *__end`) `const`

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual `~time_put` ()

- virtual [iter_type](#) [do_put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static [__c_locale](#) [_S_clone_c_locale](#) ([__c_locale](#) &__cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) &__cloc, const char *__s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) &__cloc)
- static [__c_locale](#) [_S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static [__c_locale](#) [_S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.663.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::time_put< _CharT, _OutIter >`

Primary class template [time_put](#).

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`. The [time_put](#) template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the [time_put](#) facet.

Definition at line 710 of file `locale_facets_nonio.h`.

5.663.2 Member Typedef Documentation

5.663.2.1 `template<typename _CharT , typename _OutIter > typedef _CharT std::time_put< _CharT, _OutIter >::char_type`

Public typedefs.

Reimplemented in [std::time_put_byname< _CharT, _OutIter >](#).

Definition at line 716 of file `locale_facets_nonio.h`.

5.663.2.2 `template<typename _CharT, typename _OutIter> typedef _OutIter std::time_put<_CharT, _OutIter>::iter_type`

Public typedefs.

Reimplemented in [std::time_put_byname<_CharT, _OutIter>](#).

Definition at line 717 of file locale_facets_nonio.h.

5.663.3 Constructor & Destructor Documentation

5.663.3.1 `template<typename _CharT, typename _OutIter> std::time_put< _CharT, _OutIter>::time_put (size_t __refs = 0) [inline, explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

refs Passed to the base facet class.

Definition at line 731 of file locale_facets_nonio.h.

5.663.3.2 `template<typename _CharT, typename _OutIter> virtual std::time_put<_CharT, _OutIter>::~~time_put () [inline, protected, virtual]`

Destructor.

Definition at line 777 of file locale_facets_nonio.h.

5.663.4 Member Function Documentation

5.663.4.1 `template<typename _CharT, typename _OutIter> _OutIter std::time_put<_CharT, _OutIter>::do_put (iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod) const [protected, virtual]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#) for more details.

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
format Format char.
mod Optional modifier char.

Returns

Iterator after writing.

Definition at line 1176 of file locale_facets_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

Referenced by `std::time_put< _CharT, _OutIter >::put()`.

5.663.4.2 `template<typename _CharT, typename _OutIter> _OutIter
 std::time_put< _CharT, _OutIter >::put (iter_type __s, ios_base &
 __io, char_type __fill, const tm * __tm, const _CharT * __beg,
 const _CharT * __end) const`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
beg Start of format string.
end End of format string.

Returns

Iterator after writing.

Definition at line 1141 of file locale_facets_nonio.tcc.

References std::ios_base::M_getloc(), std::time_put< _CharT, _OutIter >::do_put(), and std::__ctype_abstract_base< _CharT >::narrow().

```
5.663.4.3 template<typename _CharT , typename _OutIter > iter_type  
std::time_put< _CharT, _OutIter >::put ( iter_type __s, ios_base  
& __io, char_type __fill, const tm * __tm, char __format, char  
__mod = 0 ) const [inline]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning [time_put::do_put\(\)](#).

Parameters

- s* The stream to write to.
- io* Source of locale.
- fill* char_type to use for padding.
- tm* Struct tm with date and time info to format.
- format* Format char.
- mod* Optional modifier char.

Returns

Iterator after writing.

Definition at line 770 of file locale_facets_nonio.h.

References std::time_put< _CharT, _OutIter >::do_put().

5.663.5 Member Data Documentation

```
5.663.5.1 template<typename _CharT , typename _OutIter > locale::id  
std::time_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

Definition at line 721 of file locale_facets_nonio.h.

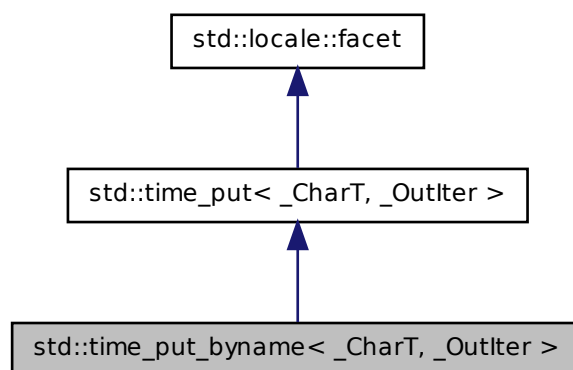
The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [locale_facets_nonio.tcc](#)

5.664 `std::time_put_byname< _CharT, _OutIter >` Class Template Reference

class `time_put_byname` [22.2.5.4].

Inheritance diagram for `std::time_put_byname< _CharT, _OutIter >`:



Public Types

- typedef `_CharT` `char_type`
- typedef `_OutIter` `iter_type`

Public Member Functions

- **`time_put_byname`** (`const char *`, `size_t __refs=0`)
- `iter_type put` (`iter_type __s`, `ios_base &__io`, `char_type __fill`, `const tm *__tm`, `char __format`, `char __mod=0`) `const`
- `iter_type put` (`iter_type __s`, `ios_base &__io`, `char_type __fill`, `const tm *__tm`, `const _CharT *__beg`, `const _CharT *__end`) `const`

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual [iter_type do_put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static [__c_locale _S_clone_c_locale](#) ([__c_locale](#) &__cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) &__cloc, const char *__s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) &__cloc)
- static [__c_locale _S_get_c_locale](#) ()
- static [_GLIBCXX_CONST](#) const char * [_S_get_c_name](#) () throw ()
- static [__c_locale _S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const char *__s)

Friends

- class [locale::_Impl](#)

5.664.1 Detailed Description

`template<typename _CharT, typename _OutIter> class std::time_put_byname< _CharT, _OutIter >`

class [time_put_byname](#) [22.2.5.4].

Definition at line 806 of file `locale_facets_nonio.h`.

5.664.2 Member Typedef Documentation

5.664.2.1 `template<typename _CharT , typename _OutIter > typedef _CharT std::time_put_byname< _CharT, _OutIter >::char_type`

Public typedefs.

Reimplemented from [std::time_put< _CharT, _OutIter >](#).

Definition at line 810 of file `locale_facets_nonio.h`.

5.664.2.2 `template<typename _CharT , typename _OutIter > typedef _OutIter std::time_put_byname< _CharT, _OutIter >::iter_type`

Public typedefs.

Reimplemented from [std::time_put<_CharT, _OutIter>](#).

Definition at line 811 of file locale_facets_nonio.h.

5.664.3 Member Function Documentation

5.664.3.1 `template<typename _CharT, typename _OutIter> _OutIter
std::time_put<_CharT, _OutIter>::do_put (iter_type __s,
ios_base & __io, char_type __fill, const tm * __tm, char __format,
char __mod) const [protected, virtual, inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

[put\(\)](#) for more details.

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
format Format char.
mod Optional modifier char.

Returns

Iterator after writing.

Definition at line 1176 of file locale_facets_nonio.tcc.

References [std::ios_base::_M_getloc\(\)](#), and [std::__ctype_abstract_base<_CharT>::widen\(\)](#).

Referenced by [std::time_put<_CharT, _OutIter>::put\(\)](#).

5.664.3.2 `template<typename _CharT, typename _OutIter> _OutIter
std::time_put<_CharT, _OutIter>::put (iter_type __s, ios_base &
__io, char_type __fill, const tm * __tm, const _CharT * __beg,
const _CharT * __end) const [inherited]`

Format and output a time or date.

5.664 `std::time_put_byname<_CharT, _OutIter>` Class Template Reference 3071

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
beg Start of format string.
end End of format string.

Returns

Iterator after writing.

Definition at line 1141 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, `std::time_put<_CharT, _OutIter>::do_put()`, and `std::__ctype_abstract_base<_CharT>::narrow()`.

5.664.3.3 `template<typename _CharT, typename _OutIter> iter_type
std::time_put<_CharT, _OutIter>::put (iter_type __s, ios_base
& __io, char_type __fill, const tm * __tm, char __format, char
__mod = 0) const [inline, inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning [time_put::do_put\(\)](#).

Parameters

s The stream to write to.
io Source of locale.
fill char_type to use for padding.
tm Struct tm with date and time info to format.
format Format char.
mod Optional modifier char.

Returns

Iterator after writing.

Definition at line 770 of file `locale_facets_nonio.h`.

References `std::time_put<_CharT, _OutIter>::do_put()`.

5.664.4 Member Data Documentation

5.664.4.1 `template<typename _CharT, typename _OutIter > locale::id
std::time_put< _CharT, _OutIter >::id` `[static, inherited]`

Numpunct facet id.

Definition at line 721 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

5.665 std::timed_mutex Class Reference

[timed_mutex](#)

Public Types

- typedef `__native_type * native_handle_type`

Public Member Functions

- **timed_mutex** (const [timed_mutex](#) &)
- void **lock** ()
- native_handle_type **native_handle** ()
- [timed_mutex](#) & **operator=** (const [timed_mutex](#) &)
- bool **try_lock** ()
- template<class _Rep, class _Period >
bool **try_lock_for** (const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<class _Clock, class _Duration >
bool **try_lock_until** (const [chrono::time_point](#)< _Clock, _Duration > &__-
atime)
- void **unlock** ()

5.665.1 Detailed Description

[timed_mutex](#)

Definition at line 166 of file `mutex`.

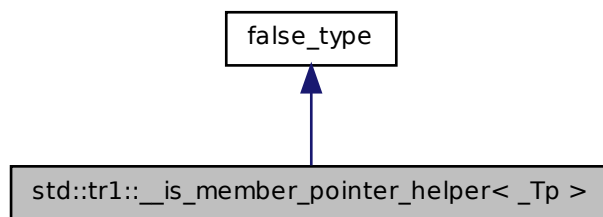
The documentation for this class was generated from the following file:

- [mutex](#)

5.666 `std::tr1::__is_member_pointer_helper< _Tp >` Struct Template Reference

`is_member_pointer`

Inheritance diagram for `std::tr1::__is_member_pointer_helper< _Tp >`:



Public Types

- typedef [integral_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value_type**

Static Public Attributes

- static const `_Tp` **value**

5.666.1 Detailed Description

template<typename `_Tp`> struct `std::tr1::__is_member_pointer_helper< _Tp >`

`is_member_pointer`

Definition at line 295 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.667 `std::tr1::add_const< _Tp >` Struct Template Reference

[add_const](#)

Public Types

- `typedef _Tp const type`

5.667.1 Detailed Description

```
template<typename _Tp> struct std::tr1::add_const< _Tp >
```

[add_const](#)

Definition at line 426 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.668 `std::tr1::add_cv< _Tp >` Struct Template Reference

[add_cv](#)

Public Types

- `typedef add_const< typename add_volatile< _Tp >::type >::type type`

5.668.1 Detailed Description

```
template<typename _Tp> struct std::tr1::add_cv< _Tp >
```

[add_cv](#)

Definition at line 436 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.669 `std::tr1::add_pointer< _Tp >` Struct Template Reference

[add_pointer](#)

Public Types

- typedef [remove_reference< _Tp >::type](#) * **type**

5.669.1 Detailed Description

```
template<typename _Tp> struct std::tr1::add_pointer< _Tp >
```

[add_pointer](#)

Definition at line 491 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.670 `std::tr1::add_volatile< _Tp >` Struct Template Reference

[add_volatile](#)

Public Types

- typedef `_Tp volatile` **type**

5.670.1 Detailed Description

```
template<typename _Tp> struct std::tr1::add_volatile< _Tp >
```

[add_volatile](#)

Definition at line 431 of file `tr1_impl/type_traits`.

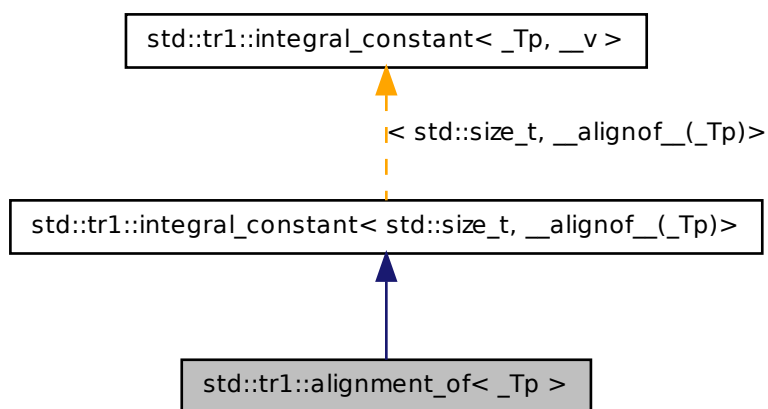
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.671 `std::tr1::alignment_of< _Tp >` Struct Template Reference

[alignment_of](#)

Inheritance diagram for `std::tr1::alignment_of< _Tp >`:



Public Types

- typedef [integral_constant](#)< `std::size_t`, `__v` > **type**
- typedef `std::size_t` **value_type**

Static Public Attributes

- static const `std::size_t` **value**

5.671.1 Detailed Description

```
template<typename _Tp> struct std::tr1::alignment_of< _Tp >
```

[alignment_of](#)

Definition at line 350 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.672 `std::tr1::array< _Tp, _Nm >` Struct Template Reference

A standard container for storing a fixed size sequence of elements.

Public Types

- `typedef const value_type * const_iterator`
- `typedef const _Tp * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse_iterator< const_iterator > const_reverse_iterator`
- `typedef std::ptrdiff_t difference_type`
- `typedef value_type * iterator`
- `typedef _Tp * pointer`
- `typedef value_type & reference`
- `typedef std::reverse_iterator< iterator > reverse_iterator`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- reference `at (size_type __n)`
- const_reference `at (size_type __n) const`
- reference `back ()`
- const_reference `back () const`
- iterator `begin ()`
- const_iterator `begin () const`
- const_iterator `cbegin () const`
- const_iterator `cend () const`
- [const_reverse_iterator](#) `crbegin () const`
- [const_reverse_iterator](#) `crend () const`
- const _Tp * `data () const`
- _Tp * `data ()`
- bool `empty () const`
- iterator `end ()`

- `const_iterator` **end** () const
- `void` **fill** (const `value_type` &__u)
- `reference` **front** ()
- `const_reference` **front** () const
- `size_type` **max_size** () const
- `reference` **operator[]** (size_type __n)
- `const_reference` **operator[]** (size_type __n) const
- `const_reverse_iterator` **rbegin** () const
- `reverse_iterator` **rbegin** ()
- `reverse_iterator` **rend** ()
- `const_reverse_iterator` **rend** () const
- `size_type` **size** () const
- `void` **swap** (`array` &__other)

Public Attributes

- `value_type` **_M_instance** [_Nm?_Nm:1]

5.672.1 Detailed Description

`template<typename _Tp, std::size_t _Nm> struct std::tr1::array< _Tp, _Nm >`

A standard container for storing a fixed size sequence of elements. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.

Parameters

Tp Type of element. Required to be a complete type.

N Number of elements.

Definition at line 49 of file `tr1_impl/array`.

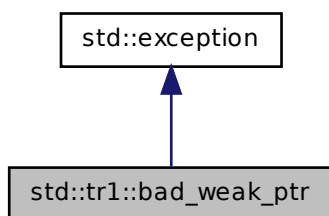
The documentation for this struct was generated from the following file:

- [tr1_impl/array](#)

5.673 std::tr1::bad_weak_ptr Class Reference

Exception possibly thrown by [shared_ptr](#).

Inheritance diagram for std::tr1::bad_weak_ptr:



Public Member Functions

- virtual char const * [what](#) () const throw ()

5.673.1 Detailed Description

Exception possibly thrown by [shared_ptr](#).

Definition at line 58 of file boost_sp_counted_base.h.

5.673.2 Member Function Documentation

5.673.2.1 virtual char const* std::tr1::bad_weak_ptr::what () const throw () [inline, virtual]

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

Definition at line 62 of file boost_sp_counted_base.h.

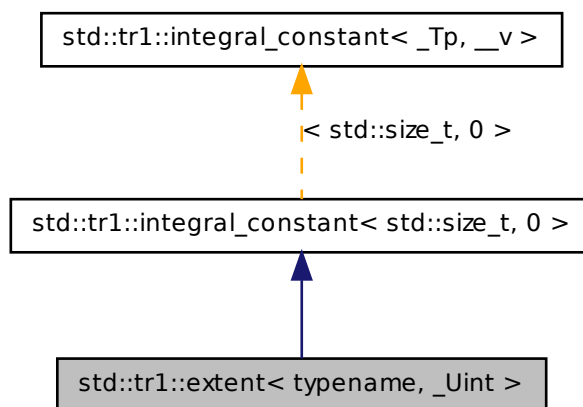
The documentation for this class was generated from the following file:

- [boost_sp_counted_base.h](#)

5.674 `std::tr1::extent< typename, _Uint >` Struct Template Reference

extent

Inheritance diagram for `std::tr1::extent< typename, _Uint >`:



Public Types

- typedef `integral_constant< std::size_t, __v >` **type**
- typedef `std::size_t` **value_type**

Static Public Attributes

- static const `std::size_t` **value**

5.674.1 Detailed Description

```
template<typename, unsigned _Uint = 0> struct std::tr1::extent< typename, _
_Uint >
```

extent

5.675 `std::tr1::has_virtual_destructor< _Tp >` Struct Template Reference 3081

Definition at line 368 of file `tr1_impl/type_traits`.

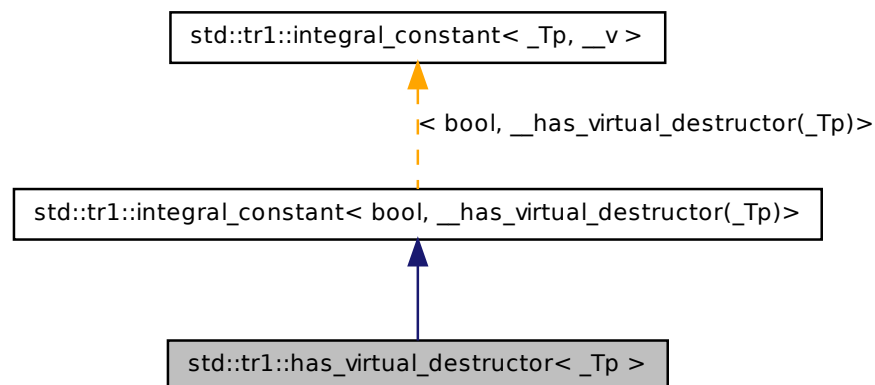
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.675 `std::tr1::has_virtual_destructor< _Tp >` Struct Template Reference

[has_virtual_destructor](#)

Inheritance diagram for `std::tr1::has_virtual_destructor< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.675.1 Detailed Description

template<typename _Tp> struct std::tr1::has_virtual_destructor< _Tp >

[has_virtual_destructor](#)

Definition at line 344 of file tr1_impl/type_traits.

The documentation for this struct was generated from the following file:

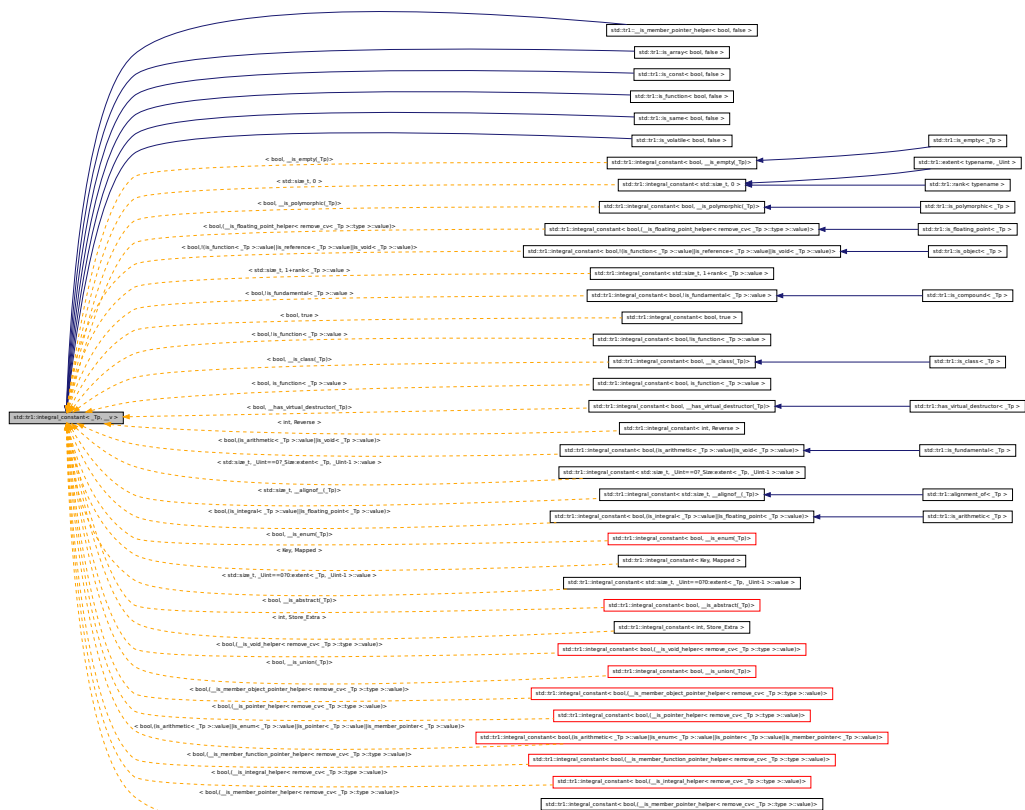
- [tr1_impl/type_traits](#)

5.676 **std::tr1::integral_constant< _Tp, __v > Struct** Template Reference

[integral_constant](#)

5.676 std::tr1::integral_constant<_Tp, __v> Struct Template Reference 3083

Inheritance diagram for `std::tr1::integral_constant<_Tp, __v>`:



Public Types

- typedef **integral_constant**< _Tp, __v > **type**
- typedef _Tp **value_type**

Static Public Attributes

- static const `_Tp` value

5.676.1 Detailed Description

```
template<typename _Tp, _Tp __v> struct std::tr1::integral_constant< _Tp, __v
>
```

[integral_constant](#)

Definition at line 67 of file tr1_impl/type_traits.

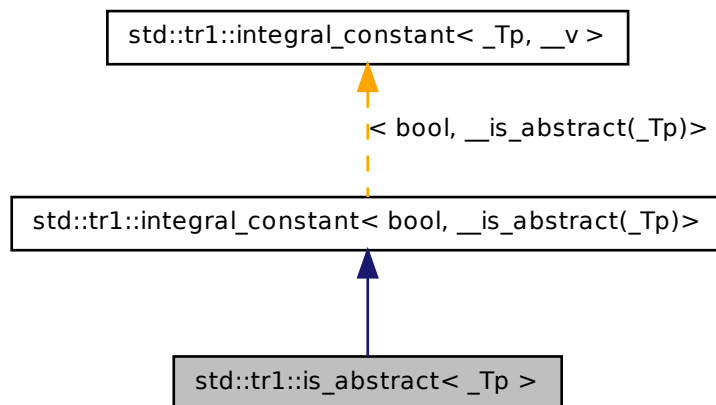
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.677 std::tr1::is_abstract< _Tp > Struct Template Reference

[is_abstract](#)

Inheritance diagram for std::tr1::is_abstract< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**

- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.677.1 Detailed Description

template<typename _Tp> struct std::tr1::is_abstract< _Tp >

[is_abstract](#)

Definition at line 338 of file tr1_impl/type_traits.

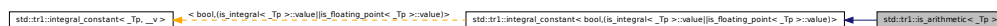
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.678 std::tr1::is_arithmetic< _Tp > Struct Template Reference

[is_arithmetic](#)

Inheritance diagram for std::tr1::is_arithmetic< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.678.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_arithmetic< _Tp >`

[is_arithmetic](#)

Definition at line 255 of file `tr1_impl/type_traits`.

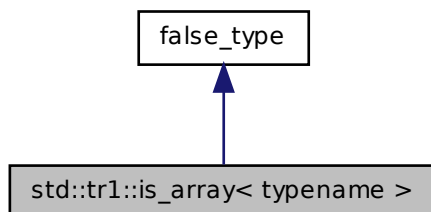
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.679 `std::tr1::is_array< typename >` Struct Template Reference

[is_array](#)

Inheritance diagram for `std::tr1::is_array< typename >`:



Public Types

- typedef [integral_constant](#)< _Tp, __v > **type**
- typedef _Tp **value_type**

Static Public Attributes

- static const _Tp **value**

5.679.1 Detailed Description

`template<typename> struct std::tr1::is_array< typename >`

[is_array](#)

Definition at line 147 of file tr1_impl/type_traits.

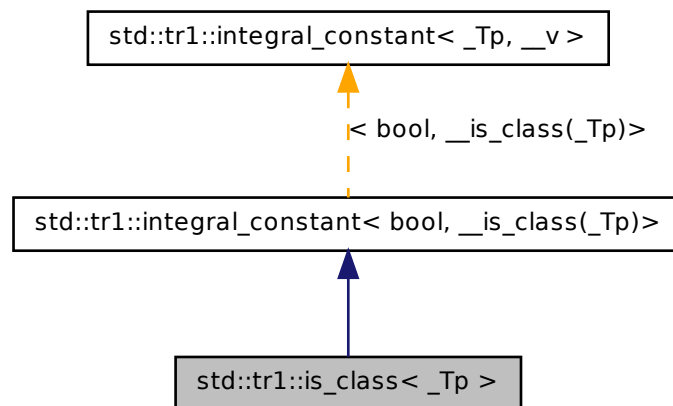
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.680 std::tr1::is_class< _Tp > Struct Template Reference

[is_class](#)

Inheritance diagram for std::tr1::is_class< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.680.1 Detailed Description

template<typename _Tp> struct std::tr1::is_class< _Tp >

[is_class](#)

Definition at line 218 of file tr1_impl/type_traits.

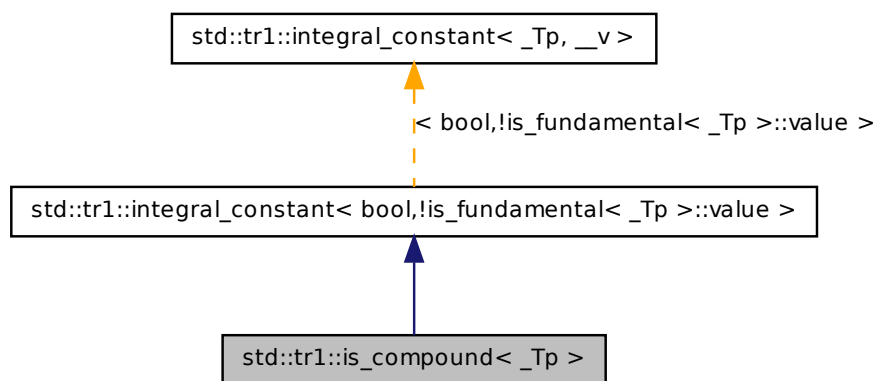
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.681 std::tr1::is_compound< _Tp > Struct Template Reference

[is_compound](#)

Inheritance diagram for std::tr1::is_compound< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.681.1 Detailed Description

template<typename _Tp> struct std::tr1::is_compound< _Tp >

[is_compound](#)

Definition at line 290 of file tr1_impl/type_traits.

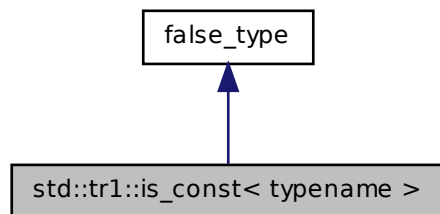
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.682 std::tr1::is_const< typename > Struct Template Reference

[is_const](#)

Inheritance diagram for std::tr1::is_const< typename >:



Public Types

- typedef [integral_constant](#)< _Tp, __v > **type**
- typedef _Tp **value_type**

Static Public Attributes

- static const _Tp **value**

5.682.1 Detailed Description

template<typename> struct std::tr1::is_const< typename >

[is_const](#)

Definition at line 308 of file tr1_impl/type_traits.

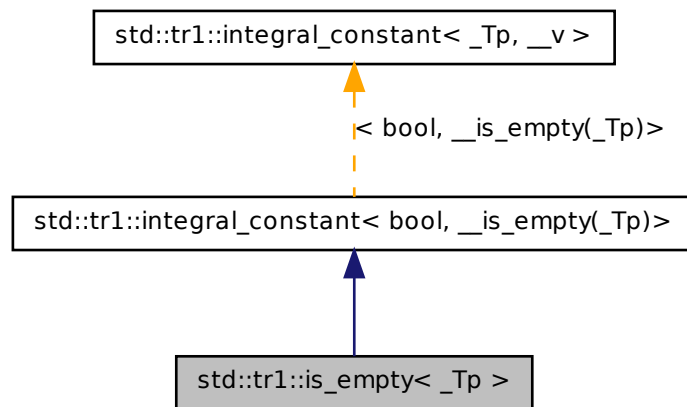
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.683 **std::tr1::is_empty< _Tp > Struct Template Reference**

[is_empty](#)

Inheritance diagram for std::tr1::is_empty< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.683.1 Detailed Description

template<typename _Tp> struct std::tr1::is_empty< _Tp >

[is_empty](#)

Definition at line 326 of file `tr1_impl/type_traits`.

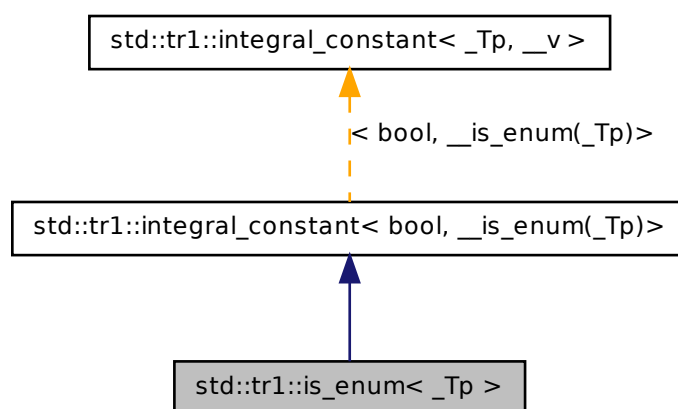
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.684 std::tr1::is_enum< _Tp > Struct Template Reference

[is_enum](#)

Inheritance diagram for std::tr1::is_enum< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.684.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_enum< _Tp >
```

[is_enum](#)

Definition at line 206 of file tr1_impl/type_traits.

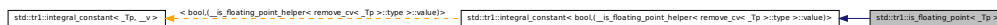
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.685 std::tr1::is_floating_point< _Tp > Struct Template Reference

[is_floating_point](#)

Inheritance diagram for std::tr1::is_floating_point< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.685.1 Detailed Description

```
template<typename _Tp> struct std::tr1::is_floating_point< _Tp >
```

[is_floating_point](#)

Definition at line 140 of file tr1_impl/type_traits.

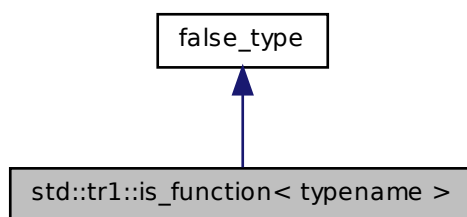
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.686 `std::tr1::is_function< typename >` Struct Template Reference

[is_function](#)

Inheritance diagram for `std::tr1::is_function< typename >`:



Public Types

- typedef [integral_constant](#)< `_Tp`, `__v` > `type`
- typedef `_Tp` `value_type`

Static Public Attributes

- static const `_Tp` `value`

5.686.1 Detailed Description

`template<typename> struct std::tr1::is_function< typename >`

[is_function](#)

Definition at line 224 of file `tr1_impl/type_traits`.

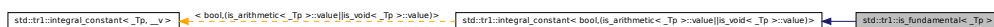
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.687 std::tr1::is_fundamental< _Tp > Struct Template Reference

[is_fundamental](#)

Inheritance diagram for std::tr1::is_fundamental< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.687.1 Detailed Description

template<typename _Tp> struct std::tr1::is_fundamental< _Tp >

[is_fundamental](#)

Definition at line 262 of file tr1_impl/type_traits.

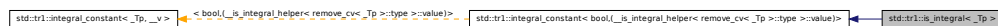
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.688 std::tr1::is_integral< _Tp > Struct Template Reference

[is_integral](#)

Inheritance diagram for `std::tr1::is_integral< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.688.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_integral< _Tp >`

[is_integral](#)

Definition at line 126 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.689 std::tr1::is_member_function_pointer< _Tp > Struct Template Reference

[is_member_function_pointer](#)

Inheritance diagram for `std::tr1::is_member_function_pointer< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**

5.690 `std::tr1::is_member_object_pointer< _Tp >` Struct Template Reference

- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.689.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_member_function_pointer< _Tp >`

[is_member_function_pointer](#)

Definition at line 199 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.690 `std::tr1::is_member_object_pointer< _Tp >` Struct Template Reference

[is_member_object_pointer](#)

Inheritance diagram for `std::tr1::is_member_object_pointer< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.690.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_member_object_pointer< _Tp >`

[is_member_object_pointer](#)

Definition at line 186 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.691 std::tr1::is_object< _Tp > Struct Template Reference

[is_object](#)

Inheritance diagram for `std::tr1::is_object< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.691.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_object< _Tp >`

[is_object](#)

Definition at line 269 of file `tr1_impl/type_traits`.

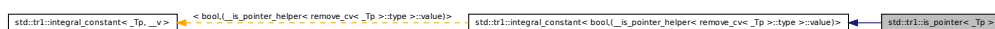
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.692 std::tr1::is_pointer< _Tp > Struct Template Reference

[is_pointer](#)

Inheritance diagram for std::tr1::is_pointer< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.692.1 Detailed Description

template<typename _Tp> struct std::tr1::is_pointer< _Tp >

[is_pointer](#)

Definition at line 165 of file tr1_impl/type_traits.

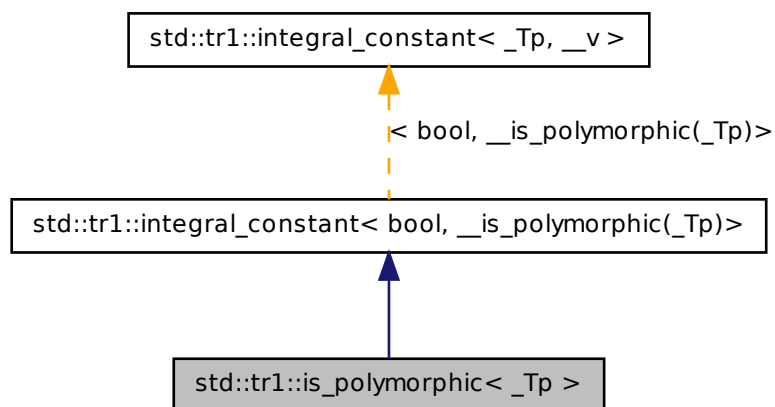
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.693 std::tr1::is_polymorphic< _Tp > Struct Template Reference

[is_polymorphic](#)

Inheritance diagram for `std::tr1::is_polymorphic< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.693.1 Detailed Description

template<typename _Tp> struct `std::tr1::is_polymorphic< _Tp >`

[is_polymorphic](#)

Definition at line 332 of file `tr1_impl/type_traits`.

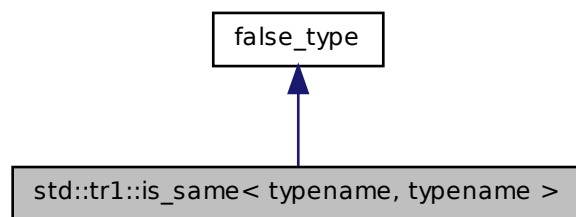
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.694 std::tr1::is_same< typename, typename > Struct Template Reference

[is_same](#)

Inheritance diagram for std::tr1::is_same< typename, typename >:



Public Types

- typedef [integral_constant](#)< _Tp, __v > **type**
- typedef _Tp **value_type**

Static Public Attributes

- static const _Tp **value**

5.694.1 Detailed Description

```
template<typename, typename> struct std::tr1::is_same< typename, typename  
>
```

[is_same](#)

Definition at line 389 of file tr1_impl/type_traits.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.695 `std::tr1::is_scalar< _Tp >` Struct Template Reference

[is_scalar](#)

Inheritance diagram for `std::tr1::is_scalar< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.695.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_scalar< _Tp >`

[is_scalar](#)

Definition at line 281 of file `tr1_impl/type_traits`.

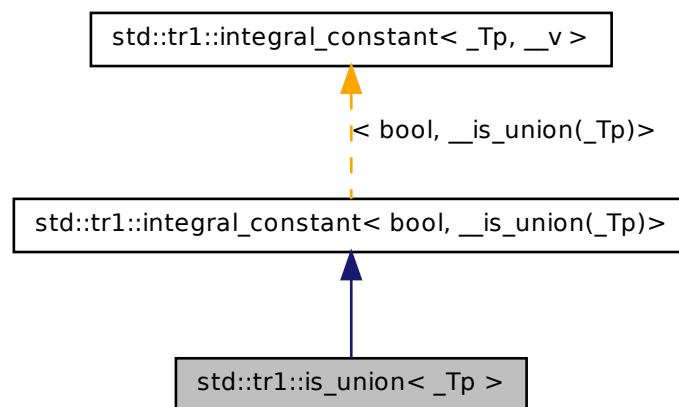
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.696 `std::tr1::is_union< _Tp >` Struct Template Reference

[is_union](#)

Inheritance diagram for std::tr1::is_union< _Tp >:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.696.1 Detailed Description

template<typename _Tp> struct std::tr1::is_union< _Tp >

[is_union](#)

Definition at line 212 of file `tr1_impl/type_traits`.

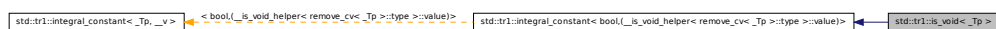
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.697 `std::tr1::is_void< _Tp >` Struct Template Reference

[is_void](#)

Inheritance diagram for `std::tr1::is_void< _Tp >`:



Public Types

- typedef [integral_constant](#)< bool, __v > **type**
- typedef bool **value_type**

Static Public Attributes

- static const bool **value**

5.697.1 Detailed Description

`template<typename _Tp> struct std::tr1::is_void< _Tp >`

[is_void](#)

Definition at line 96 of file `tr1_impl/type_traits`.

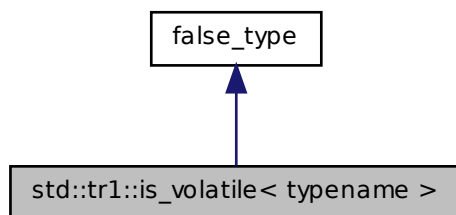
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.698 `std::tr1::is_volatile< typename >` Struct Template Reference

[is_volatile](#)

Inheritance diagram for `std::tr1::is_volatile< typename >`:



Public Types

- typedef [integral_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value_type**

Static Public Attributes

- static const `_Tp` **value**

5.698.1 Detailed Description

`template<typename> struct std::tr1::is_volatile< typename >`

[is_volatile](#)

Definition at line 317 of file `tr1_impl/type_traits`.

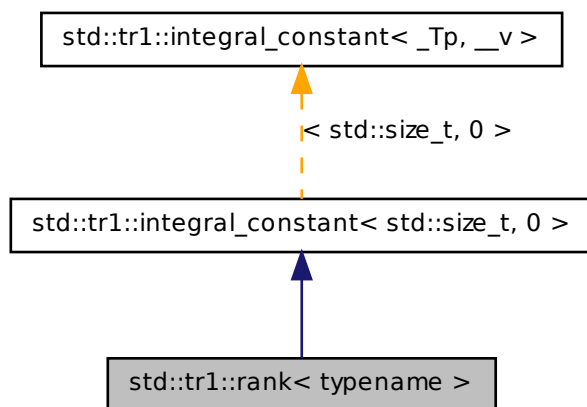
The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.699 `std::tr1::rank< typename >` Struct Template Reference

`rank`

Inheritance diagram for `std::tr1::rank< typename >`:



Public Types

- typedef [integral_constant](#)< `std::size_t`, `__v` > **type**
- typedef `std::size_t` **value_type**

Static Public Attributes

- static const `std::size_t` **value**

5.699.1 Detailed Description

template<typename> struct `std::tr1::rank< typename >`

`rank`

Definition at line 355 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.700 `std::tr1::remove_all_extents< _Tp >` Struct Template Reference

[remove_all_extents](#)

Public Types

- `typedef _Tp type`

5.700.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_all_extents< _Tp >`

[remove_all_extents](#)

Definition at line 459 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.701 `std::tr1::remove_const< _Tp >` Struct Template Reference

[remove_const](#)

Public Types

- `typedef _Tp type`

5.701.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_const< _Tp >`

[remove_const](#)

Definition at line 400 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.702 `std::tr1::remove_cv< _Tp >` Struct Template Reference

[remove_cv](#)

Public Types

- typedef [remove_const](#)< typename [remove_volatile](#)< _Tp >::type >::type **type**

5.702.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_cv< _Tp >`

[remove_cv](#)

Definition at line 418 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.703 `std::tr1::remove_extent< _Tp >` Struct Template Reference

[remove_extent](#)

Public Types

- typedef `_Tp` **type**

5.703.1 Detailed Description

`template<typename _Tp> struct std::tr1::remove_extent< _Tp >`

[remove_extent](#)

Definition at line 446 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.704 `std::tr1::remove_pointer< _Tp >` Struct Template Reference

[remove_pointer](#)

Inherits `std::tr1::__remove_pointer_helper< _Tp, remove_cv< _Tp >::type >`.

Public Types

- `typedef _Tp type`

5.704.1 Detailed Description

```
template<typename _Tp> struct std::tr1::remove_pointer< _Tp >
```

[remove_pointer](#)

Definition at line 482 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.705 `std::tr1::remove_volatile< _Tp >` Struct Template Reference

[remove_volatile](#)

Public Types

- `typedef _Tp type`

5.705.1 Detailed Description

```
template<typename _Tp> struct std::tr1::remove_volatile< _Tp >
```

[remove_volatile](#)

Definition at line 409 of file `tr1_impl/type_traits`.

The documentation for this struct was generated from the following file:

- [tr1_impl/type_traits](#)

5.706 std::try_to_lock_t Struct Reference

Try to acquire ownership of the mutex without blocking.

5.706.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

Definition at line 378 of file mutex.

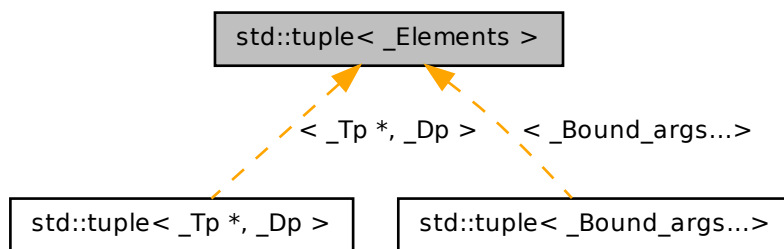
The documentation for this struct was generated from the following file:

- [mutex](#)

5.707 std::tuple< _Elements > Class Template Reference

tuple

Inheritance diagram for std::tuple< _Elements >:



Public Member Functions

- **tuple** (const _Elements &...__elements)
- **tuple** (const [tuple](#) &)
- **tuple** ([tuple](#) &&__in)

- template<typename... _UElements, typename = typename std::enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
tuple (_UElements &&...__elements)
- template<typename... _UElements, typename = typename std::enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
tuple (const [tuple](#)< _UElements...> &__in)
- template<typename... _UElements, typename = typename std::enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
tuple ([tuple](#)< _UElements...> &&__in)
- template<typename... _UElements, typename = typename std::enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
[tuple](#) & **operator=** (const [tuple](#)< _UElements...> &__in)
- template<typename... _UElements, typename = typename std::enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
[tuple](#) & **operator=** ([tuple](#)< _UElements...> &&__in)
- [tuple](#) & **operator=** (const [tuple](#) &__in)
- [tuple](#) & **operator=** ([tuple](#) &&__in)
- void **swap** ([tuple](#) &__in)

5.707.1 Detailed Description

template<typename... _Elements> class std::tuple< _Elements >

[tuple](#)

Definition at line 223 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

5.708 std::tuple< _T1 > Class Template Reference

[tuple](#) (1-element).

Inherits [_Tuple_impl](#)< 0, _T1 >.

Public Member Functions

- **tuple** (const _T1 &__a1)
- **tuple** (const [tuple](#) &)
- **tuple** ([tuple](#) &&__in)

- `template<typename _U1 , typename = typename std::enable_if<std::is_convertible<_U1, _T1>::value>::type>`
`tuple (_U1 &&__a1)`
- `template<typename _U1 >`
`tuple (const tuple< _U1 > &__in)`
- `template<typename _U1 >`
`tuple (tuple< _U1 > &&__in)`
- `template<typename _U1 >`
`tuple & operator= (const tuple< _U1 > &__in)`
- `template<typename _U1 >`
`tuple & operator= (tuple< _U1 > &&__in)`
- `tuple & operator= (const tuple &__in)`
- `tuple & operator= (tuple &&__in)`
- `void swap (tuple &__in)`

5.708.1 Detailed Description

`template<typename _T1> class std::tuple< _T1 >`

tuple (1-element).

Definition at line 406 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

5.709 std::tuple< _T1, _T2 > Class Template Reference

tuple (2-element), with construction and assignment from a pair.

Inherits `_Tuple_impl< 0, _T1, _T2 >`.

Public Member Functions

- `tuple (const _T1 &__a1, const _T2 &__a2)`
- `tuple (const tuple &)`
- `template<typename _U1, typename _U2 >`
`tuple (const pair< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >`
`tuple (pair< _U1, _U2 > &&__in)`
- `tuple (tuple &&__in)`

- `template<typename _U1, typename _U2 >
tuple (_U1 &&__a1, _U2 &&__a2)`
- `template<typename _U1, typename _U2 >
tuple (const tuple< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >
tuple (tuple< _U1, _U2 > &&__in)`
- `template<typename _U1, typename _U2 >
tuple & operator= (const tuple< _U1, _U2 > &__in)`
- `tuple & operator= (const tuple &__in)`
- `tuple & operator= (tuple &&__in)`
- `template<typename _U1, typename _U2 >
tuple & operator= (tuple< _U1, _U2 > &&__in)`
- `template<typename _U1, typename _U2 >
tuple & operator= (const pair< _U1, _U2 > &__in)`
- `template<typename _U1, typename _U2 >
tuple & operator= (pair< _U1, _U2 > &&__in)`
- `void swap (tuple &__in)`

5.709.1 Detailed Description

`template<typename _T1, typename _T2> class std::tuple< _T1, _T2 >`

tuple (2-element), with construction and assignment from a pair.

Definition at line 308 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

5.710 std::tuple_element< 0, tuple< _Head, _Tail...> > Struct Template Reference

Public Types

- `typedef _Head type`

5.710.1 Detailed Description

`template<typename _Head, typename... _Tail> struct std::tuple_element< 0, tuple< _Head, _Tail...> >`

Basis case for tuple_element: The first element is the one we're seeking.

Definition at line 489 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.711 `std::tuple_element< __i, tuple< _Head, _Tail...>>` Struct Template Reference

Inherits `tuple_element< __i-1, tuple< _Tail...>>`.

5.711.1 Detailed Description

```
template<std::size_t __i, typename _Head, typename... _Tail> struct std::tuple_
element< __i, tuple< _Head, _Tail...>> >
```

Recursive case for `tuple_element`: strip off the first element in the tuple and retrieve the (i-1)th element of the remaining tuple.

Definition at line 482 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.712 `std::tuple_size< tuple< _Elements...>>` Struct Template Reference

```
class tuple_size
```

Static Public Attributes

- static const `std::size_t` **value**

5.712.1 Detailed Description

```
template<typename... _Elements> struct std::tuple_size< tuple< _Elements...>>
>
```

```
class tuple_size
```

Definition at line 500 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

5.713 std::type_info Class Reference

Part of RTTI.

Inherited by `__cxxabiv1::__array_type_info`, `__cxxabiv1::__class_type_info`, `__cxxabiv1::__enum_type_info`, `__cxxabiv1::__function_type_info`, `__cxxabiv1::__fundamental_type_info`, and `__cxxabiv1::__pbase_type_info`.

Public Member Functions

- virtual `~type_info()`
- virtual bool `__do_catch` (const `type_info` *`__thr_type`, void **`__thr_obj`, unsigned `__outer`) const
- virtual bool `__do_upcast` (const `__cxxabiv1::__class_type_info` *`__target`, void **`__obj_ptr`) const
- virtual bool `__is_function_p` () const
- virtual bool `__is_pointer_p` () const
- bool `before` (const `type_info` &`__arg`) const
- size_t `hash_code` () const throw ()
- const char * `name` () const
- bool `operator!=` (const `type_info` &`__arg`) const
- bool `operator==` (const `type_info` &`__arg`) const

Protected Member Functions

- `type_info` (const char *`__n`)

Protected Attributes

- const char * `__name`

5.713.1 Detailed Description

Part of RTTI. The `type_info` class describes type information generated by an implementation.

Definition at line 92 of file typeinfo.

5.713.2 Constructor & Destructor Documentation

5.713.2.1 virtual std::type_info::~~type_info () [virtual]

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated [type_info](#) structures in the new-abi.

5.713.3 Member Function Documentation

5.713.3.1 const char* std::type_info::name () const [inline]

Returns an *implementation-defined* byte string; this is not portable between compilers!

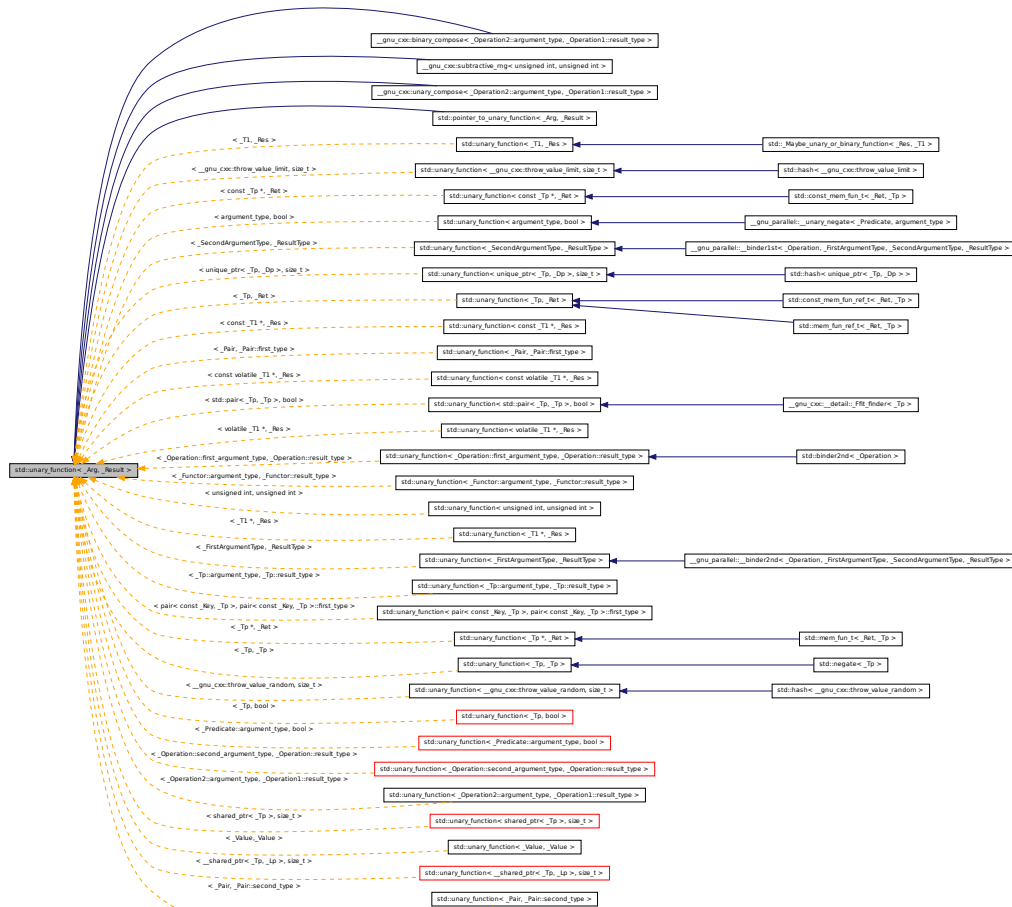
Definition at line 103 of file typeinfo.

The documentation for this class was generated from the following file:

- [typeinfo](#)

5.714 std::unary_function< _Arg, _Result > Struct Template Reference

Inheritance diagram for std::unary_function< _Arg, _Result >:



Public Types

- typedef `_Arg` `argument_type`
- typedef `_Result` `result_type`

5.714.1 Detailed Description

template<typename _Arg, typename _Result> struct std::unary_function< _Arg, _Result >

This is one of the [functor base classes](#).

Definition at line 100 of file stl_function.h.

5.714.2 Member Typedef Documentation

**5.714.2.1 template<typename _Arg, typename _Result> typedef _Arg
std::unary_function< _Arg, _Result >::argument_type**

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file stl_function.h.

**5.714.2.2 template<typename _Arg, typename _Result> typedef _Result
std::unary_function< _Arg, _Result >::result_type**

`result_type` is the return type

Definition at line 105 of file stl_function.h.

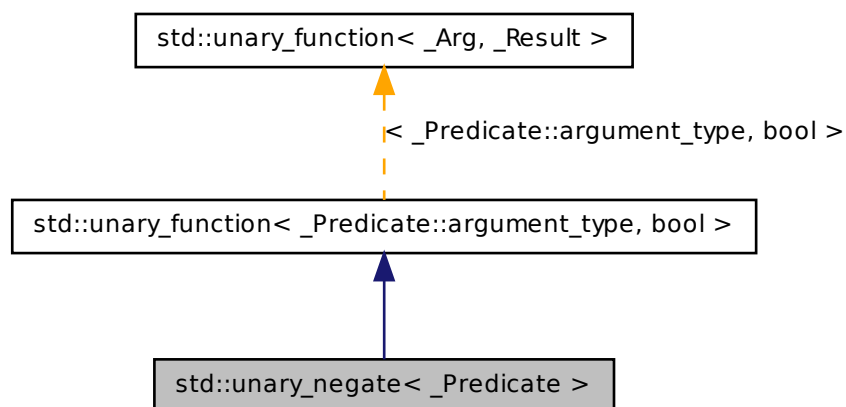
The documentation for this struct was generated from the following file:

- [stl_function.h](#)

5.715 **std::unary_negate< _Predicate > Class Template Reference**

One of the [negation functors](#).

Inheritance diagram for std::unary_negate< _Predicate >:



Public Types

- typedef `_Predicate::argument_type` [argument_type](#)
- typedef `bool` [result_type](#)

Public Member Functions

- **unary_negate** (const `_Predicate` &__x)
- **bool operator()** (const typename `_Predicate::argument_type` &__x) const

Protected Attributes

- `_Predicate` **_M_pred**

5.715.1 Detailed Description

`template<typename _Predicate> class std::unary_negate< _Predicate >`

One of the [negation functors](#).

Definition at line 346 of file `stl_function.h`.

5.715.2 Member Typedef Documentation

5.715.2.1 `typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type , bool >::argument_type [inherited]`

`argument_type` is the type of the /// argument (no surprises here)

Definition at line 102 of file `stl_function.h`.

5.715.2.2 `typedef bool std::unary_function< _Predicate::argument_type , bool >::result_type [inherited]`

`result_type` is the return type

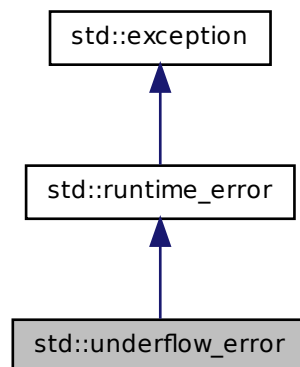
Definition at line 105 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

5.716 `std::underflow_error` Class Reference

Inheritance diagram for `std::underflow_error`:



Public Member Functions

- **`underflow_error`** (const [string](#) &__arg)
- virtual const char * [what](#) () const throw ()

5.716.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 140 of file `stdexcept`.

5.716.2 Member Function Documentation

5.716.2.1 `virtual const char* std::runtime_error::what () const throw ()` [[virtual](#), [inherited](#)]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

5.717 `std::uniform_int_distribution< _IntType >` Class Template Reference

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- [uniform_int_distribution](#) (`_IntType __a=0, _IntType __b=std::numeric_limits<_IntType >::max()`)
- [uniform_int_distribution](#) (const [param_type](#) &__p)
- [result_type a](#) () const
- [result_type b](#) () const
- [result_type max](#) () const
- [result_type min](#) () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >
[result_type operator\(\)](#) (`_UniformRandomNumberGenerator &__urng`)
- void [param](#) (const [param_type](#) &__param)
- [param_type param](#) () const
- void [reset](#) ()

Public Attributes

- [param_type _M_param](#)

5.717.1 Detailed Description

template<typename _IntType = int> class std::uniform_int_distribution< _IntType >

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

Definition at line 1631 of file random.h.

5.717.2 Member Typedef Documentation

5.717.2.1 `template<typename _IntType = int> typedef _IntType
std::uniform_int_distribution<_IntType>::result_type`

The type of the range of the distribution.

Definition at line 1638 of file random.h.

5.717.3 Constructor & Destructor Documentation

5.717.3.1 `template<typename _IntType = int> std::uniform_int_distribution<
_IntType>::uniform_int_distribution (_IntType __a = 0, _IntType
__b = std::numeric_limits<_IntType>::max()) [inline,
explicit]`

Constructs a uniform distribution object.

Definition at line 1674 of file random.h.

5.717.4 Member Function Documentation

5.717.4.1 `template<typename _IntType = int> result_type
std::uniform_int_distribution<_IntType>::max () const
[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1726 of file random.h.

5.717.4.2 `template<typename _IntType = int> result_type
std::uniform_int_distribution< _IntType >::min () const
[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1719 of file random.h.

5.717.4.3 `template<typename _IntType = int> template<typename
_UniformRandomNumberGenerator > result_type
std::uniform_int_distribution< _IntType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 1734 of file random.h.

References `std::uniform_int_distribution< _IntType >::operator()()`, and `std::uniform_int_distribution< _IntType >::param()`.

Referenced by `std::uniform_int_distribution< _IntType >::operator()()`.

5.717.4.4 `template<typename _IntType = int> void std::uniform_int_
distribution< _IntType >::param (const param_type & __param)
[inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 1712 of file random.h.

5.717.4.5 `template<typename _IntType = int> param_type
std::uniform_int_distribution< _IntType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 1704 of file random.h.

Referenced by `std::uniform_int_distribution< _IntType >::operator()()`, `std::operator==()`, and `std::operator>>()`.

5.718 `std::uniform_int_distribution<_IntType>::param_type` Struct Reference 3125

5.717.4.6 `template<typename _IntType = int> void
std::uniform_int_distribution<_IntType>::reset () [inline]`

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 1690 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.718 `std::uniform_int_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `uniform_int_distribution<_IntType>` `distribution_type`

Public Member Functions

- `param_type` (`_IntType __a=0, _IntType __b=std::numeric_limits<_IntType>::max()`)
- `result_type a` () const
- `result_type b` () const

Friends

- bool `operator==` (const `param_type` &__p1, const `param_type` &__p2)

5.718.1 Detailed Description

`template<typename _IntType = int> struct std::uniform_int_distribution<_IntType>::param_type`

Parameter type.

Definition at line 1640 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.719 `std::uniform_real_distribution< _RealType >` Class Template Reference

Uniform continuous distribution for random numbers.

Classes

- struct `param_type`

Public Types

- typedef `_RealType` `result_type`

Public Member Functions

- `uniform_real_distribution` (`_RealType __a=_RealType(0), _RealType __b=-RealType(1)`)
- `uniform_real_distribution` (`const param_type &__p`)
- `result_type a` () const
- `result_type b` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator > result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `void param` (`const param_type &__param`)
- `param_type param` () const
- `void reset` ()

5.719.1 Detailed Description

```
template<typename _RealType = double> class std::uniform_real_distribution<
_RealType >
```

Uniform continuous distribution for random numbers. A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1803 of file random.h.

5.719.2 Member Typedef Documentation

5.719.2.1 `template<typename _RealType = double> typedef _RealType
std::uniform_real_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 1810 of file `random.h`.

5.719.3 Constructor & Destructor Documentation

5.719.3.1 `template<typename _RealType = double> std::uniform_real_
distribution<_RealType>::uniform_real_distribution (_RealType
__a = _RealType(0), _RealType __b = _RealType(1))
[inline, explicit]`

Constructs a [uniform_real_distribution](#) object.

Parameters

`__min` [IN] The lower bound of the distribution.

`__max` [IN] The upper bound of the distribution.

Definition at line 1849 of file `random.h`.

5.719.4 Member Function Documentation

5.719.4.1 `template<typename _RealType = double> result_type
std::uniform_real_distribution<_RealType>::max () const
[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1901 of file `random.h`.

5.719.4.2 `template<typename _RealType = double> result_type
std::uniform_real_distribution<_RealType>::min () const
[inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1894 of file `random.h`.

5.719.4.3 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::uniform_real_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 1909 of file random.h.

References `std::uniform_real_distribution< _RealType >::operator()()`, and
`std::uniform_real_distribution< _RealType >::param()`.

Referenced by `std::uniform_real_distribution< _RealType >::operator()()`.

5.719.4.4 `template<typename _RealType = double> void
std::uniform_real_distribution< _RealType >::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 1887 of file random.h.

5.719.4.5 `template<typename _RealType = double> param_type
std::uniform_real_distribution< _RealType >::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 1879 of file random.h.

Referenced by `std::uniform_real_distribution< _RealType >::operator()()`,
`std::operator==()`, and `std::operator>>()`.

5.719.4.6 `template<typename _RealType = double> void
std::uniform_real_distribution< _RealType >::reset ()
[inline]`

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1865 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

5.720 `std::uniform_real_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `uniform_real_distribution<_RealType>` `distribution_type`

Public Member Functions

- `param_type` (`_RealType` __a=`_RealType`(0), `_RealType` __b=`_RealType`(1))
- `result_type` a () const
- `result_type` b () const

Friends

- bool `operator==` (const `param_type` &__p1, const `param_type` &__p2)

5.720.1 Detailed Description

`template<typename _RealType = double> struct std::uniform_real_distribution<_RealType>::param_type`

Parameter type.

Definition at line 1812 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

5.721 `std::unique_lock<_Mutex>` Class Template Reference

[unique_lock](#)

Public Types

- typedef `_Mutex` `mutex_type`

Public Member Functions

- **unique_lock** (mutex_type &__m)
- **unique_lock** (mutex_type &__m, [try_to_lock_t](#))
- **unique_lock** (const [unique_lock](#) &)
- **unique_lock** (mutex_type &__m, [adopt_lock_t](#))
- **unique_lock** ([unique_lock](#) &&__u)
- **unique_lock** (mutex_type &__m, [defer_lock_t](#))
- template<typename _Clock, typename _Duration >
unique_lock (mutex_type &__m, const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- template<typename _Rep, typename _Period >
unique_lock (mutex_type &__m, const [chrono::duration](#)< _Rep, _Period > &__rtime)
- void **lock** ()
- mutex_type * **mutex** () const
- **operator bool** () const
- [unique_lock](#) & **operator=** (const [unique_lock](#) &)
- [unique_lock](#) & **operator=** ([unique_lock](#) &&__u)
- bool **owns_lock** () const
- mutex_type * **release** ()
- void **swap** ([unique_lock](#) &__u)
- bool **try_lock** ()
- template<typename _Rep, typename _Period >
bool **try_lock_for** (const [chrono::duration](#)< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration >
bool **try_lock_until** (const [chrono::time_point](#)< _Clock, _Duration > &__atime)
- void **unlock** ()

5.721.1 Detailed Description

template<typename _Mutex> class std::unique_lock< _Mutex >

[unique_lock](#)

Definition at line 415 of file mutex.

The documentation for this class was generated from the following file:

- [mutex](#)

5.722 `std::unique_ptr< _Tp, _Dp >` Class Template Reference

20.7.12.2 `unique_ptr` for single objects.

Public Types

- `typedef _Dp deleter_type`
- `typedef _Tp element_type`
- `typedef _Pointer::type pointer`

Public Member Functions

- `unique_ptr` (pointer __p)
- `unique_ptr` (pointer __p, typename `std::remove_reference< deleter_type >::type &&__d`)
- `template<typename _Up , typename = typename std::enable_if<std::is_convertible<_Up*, _Tp*>::value && std::is_same<_Dp, default_delete<_Tp>>::value>::type>`
`unique_ptr` (`auto_ptr< _Up > &&__u`)
- `unique_ptr` (nullptr_t)
- `unique_ptr` (const `unique_ptr` &)
- `unique_ptr` (pointer __p, typename `std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d`)
- `unique_ptr` (`unique_ptr` &&__u)
- `template<typename _Up , typename _Ep , typename = typename std::enable_if <std::is_convertible<typename unique_ptr<_Up, _Ep>::pointer, pointer>::value && !std::is_array<_Up>::value && ((std::is_reference<_Dp>::value && std::is_same<_Ep, _Dp>::value) || (!std::is_reference<_Dp>::value && std::is_convertible<_Ep, _Dp>::value))>::type>`
`unique_ptr` (`unique_ptr< _Up, _Ep > &&__u`)
- `pointer get () const`
- `deleter_type & get_deleter ()`
- `const deleter_type & get_deleter () const`
- `operator bool () const`
- `std::add_lvalue_reference< element_type >::type operator* () const`
- `pointer operator-> () const`
- `unique_ptr & operator=` (nullptr_t)
- `template<typename _Up , typename _Ep , typename = typename std::enable_if <std::is_convertible<typename unique_ptr<_Up, _Ep>::pointer, pointer>::value && !std::is_array<_Up>::value>::type>`
`unique_ptr & operator=` (`unique_ptr< _Up, _Ep > &&__u`)
- `unique_ptr & operator=` (`unique_ptr` &&__u)
- `unique_ptr & operator=` (const `unique_ptr` &)

- pointer **release** ()
- void **reset** (pointer __p=pointer())
- void **swap** ([unique_ptr](#) &__u)

5.722.1 Detailed Description

template<typename _Tp, typename _Dp = default_delete<_Tp>> class std::unique_ptr< _Tp, _Dp >

20.7.12.2 [unique_ptr](#) for single objects.

Definition at line 82 of file [unique_ptr.h](#).

The documentation for this class was generated from the following file:

- [unique_ptr.h](#)

5.723 std::unique_ptr< _Tp[], _Dp > Class Template Reference

20.7.12.3 [unique_ptr](#) for array objects with a runtime length

Public Types

- typedef _Dp **deleter_type**
- typedef _Tp **element_type**
- typedef _Tp * **pointer**

Public Member Functions

- **unique_ptr** (pointer __p)
- template<typename _Up >
unique_ptr (_Up *, typename [std::remove_reference](#)< deleter_type >::type &&, typename [std::enable_if](#)< [std::is_convertible](#)< _Up *, pointer >::value >::type *==0)
- **unique_ptr** (pointer __p, typename [std::remove_reference](#)< deleter_type >::type &&__d)
- **unique_ptr** (nullptr_t)
- **unique_ptr** (const [unique_ptr](#) &)
- **unique_ptr** (pointer __p, typename [std::conditional](#)< [std::is_reference](#)< deleter_type >::value, deleter_type, const deleter_type & >::type __d)

- `unique_ptr` (`unique_ptr` &&__u)
- `template<typename _Up >`
`unique_ptr` (`_Up` *, `typename std::conditional< std::is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type, typename std::enable_if< std::is_convertible< _Up *, pointer >::value >::type * = 0`)
- `template<typename _Up >`
`unique_ptr` (`_Up` *, `typename std::enable_if< std::is_convertible< _Up *, pointer >::value >::type * = 0`)
- `template<typename _Up, typename _Ep >`
`unique_ptr` (`unique_ptr`< `_Up`, `_Ep` > &&__u)
- `pointer get () const`
- `deleter_type & get_deleter ()`
- `const deleter_type & get_deleter () const`
- `operator bool () const`
- `unique_ptr & operator= (nullptr_t)`
- `template<typename _Up, typename _Ep >`
`unique_ptr & operator= (unique_ptr`< `_Up`, `_Ep` > &&__u)
- `unique_ptr & operator= (unique_ptr &&__u)`
- `unique_ptr & operator= (const unique_ptr &)`
- `std::add_lvalue_reference< element_type >::type operator[] (size_t __i) const`
- `pointer release ()`
- `void reset (pointer __p=pointer())`
- `void reset (nullptr_t)`
- `template<typename _Up >`
`void reset (_Up)`
- `void swap (unique_ptr &__u)`

5.723.1 Detailed Description

`template<typename _Tp, typename _Dp> class std::unique_ptr<_Tp[],_Dp>`

20.7.12.3 `unique_ptr` for array objects with a runtime length

Definition at line 260 of file `unique_ptr.h`.

The documentation for this class was generated from the following file:

- `unique_ptr.h`

5.724 `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Inherits `std::__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::__Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::__Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Base::hasher hasher`
- `typedef __detail::__Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Key key_type`
- `typedef __detail::__Node_iterator< value_type, __constant_iterators, __cache_hash_code > local_iterator`
- `typedef _Allocator::pointer pointer`
- `typedef _Allocator::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

Public Member Functions

- **unordered_map** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- `template<typename _InputIterator >`
unordered_map (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** ([initializer_list](#)< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- `const _RehashPolicy & __rehash_policy () const`
- `void __rehash_policy (const _RehashPolicy &)`

- `_Value_allocator_type` **M_get_Value_allocator** () const
- `local_iterator` **begin** (size_type __n)
- `const_local_iterator` **begin** (size_type __n) const
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- size_type **bucket** (const key_type &__k) const
- size_type **bucket_count** () const
- size_type **bucket_size** (size_type __n) const
- `const_iterator` **cbegin** () const
- `const_local_iterator` **cbegin** (size_type __n) const
- `const_iterator` **cend** () const
- `const_local_iterator` **cend** (size_type) const
- void **clear** ()
- size_type **count** (const key_type &__k) const
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `local_iterator` **end** (size_type)
- `const_local_iterator` **end** (size_type) const
- [std::pair](#)< iterator, iterator > **equal_range** (const key_type &__k)
- [std::pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- size_type **erase** (const key_type &)
- `iterator` **erase** (const_iterator)
- `iterator` **erase** (const_iterator, const_iterator)
- `const_iterator` **find** (const key_type &__k) const
- `iterator` **find** (const key_type &__k)
- `allocator_type` **get_allocator** () const
- [_Insert_Return_Type](#) **insert** (const value_type &__v)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** ([initializer_list](#)< value_type > __l)
- `iterator` **insert** (const_iterator, const value_type &__v)
- key_equal **key_eq** () const
- float **load_factor** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- `unordered_map` & **operator=** ([initializer_list](#)< value_type > __l)
- void **rehash** (size_type __n)
- size_type **size** () const
- void **swap** (_Hashtable &)

Friends

- struct `__detail::_Map_base`

5.724.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred =
std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>
>> class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys. Meets the requirements of a [container](#), and [unordered associative container](#)

Parameters

- Key* Type of key objects.
- Tp* Type of mapped objects.
- Hash* Hashing function object type, defaults to `hash<Value>`.
- Pred* Predicate function object type, defaults to `equal_to<Value>`.
- Alloc* Allocator type, defaults to `allocator<Key>`.

The resulting value type of the container is `std::pair<const Key, Tp>`.

Definition at line 254 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered_map.h](#)

5.725 `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Inherits `std::__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`.

Public Types

- typedef `_Base::allocator_type` **allocator_type**
- typedef `__detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code >` **const_iterator**

- typedef __detail::__Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > **const_local_iterator**
- typedef _Allocator::const_pointer **const_pointer**
- typedef _Allocator::const_reference **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Base::hasher **hasher**
- typedef __detail::__Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Key **key_type**
- typedef __detail::__Node_iterator< value_type, __constant_iterators, __cache_hash_code > **local_iterator**
- typedef _Allocator::pointer **pointer**
- typedef _Allocator::reference **reference**
- typedef _Base::size_type **size_type**
- typedef [_Base::value_type](#) **value_type**

Public Member Functions

- **unordered_multimap** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_multimap (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** ([initializer_list](#)< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- const _RehashPolicy & **__rehash_policy** () const
- void **__rehash_policy** (const _RehashPolicy &)
- _Value_allocator_type **_M_get_Value_allocator** () const
- local_iterator **begin** (size_type __n)
- const_local_iterator **begin** (size_type __n) const
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket** (const key_type &__k) const
- size_type **bucket_count** () const
- size_type **bucket_size** (size_type __n) const
- const_iterator **cbegin** () const
- const_local_iterator **cbegin** (size_type __n) const
- const_iterator **cend** () const
- const_local_iterator **cend** (size_type) const
- void **clear** ()

- size_type **count** (const key_type &__k) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- local_iterator **end** (size_type)
- const_local_iterator **end** (size_type) const
- [std::pair](#)< iterator, iterator > **equal_range** (const key_type &__k)
- [std::pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- size_type **erase** (const key_type &)
- iterator **erase** (const_iterator)
- iterator **erase** (const_iterator, const_iterator)
- const_iterator **find** (const key_type &__k) const
- iterator **find** (const key_type &__k)
- allocator_type **get_allocator** () const
- [_Insert_Return_Type](#) **insert** (const value_type &__v)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** ([initializer_list](#)< value_type > __l)
- iterator **insert** (const_iterator, const value_type &__v)
- key_equal **key_eq** () const
- float **load_factor** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- [unordered_multimap](#) & **operator=** ([initializer_list](#)< value_type > __l)
- void **rehash** (size_type __n)
- size_type **size** () const
- void **swap** (_Hashtable &)

Friends

- struct [__detail::_Map_base](#)

5.725.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred =
std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>
>> class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys. Meets the requirements of a [container](#), and [unordered associative container](#)

5.726 `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference 3139

Parameters

Key Type of key objects.

Tp Type of mapped objects.

Hash Hashing function object type, defaults to `hash<Value>`.

Pred Predicate function object type, defaults to `equal_to<Value>`.

Alloc Allocator type, defaults to `allocator<Key>`.

The resulting value type of the container is `std::pair<const Key, Tp>`.

Definition at line 322 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered_map.h](#)

5.726 `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Inherits `std::__unordered_multiset< _Value, _Hash, _Pred, _Alloc >`.

Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef __detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_iterator`
- `typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > const_local_iterator`
- `typedef _Allocator::const_pointer const_pointer`
- `typedef _Allocator::const_reference const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Base::hasher hasher`
- `typedef __detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > iterator`
- `typedef _Base::key_equal key_equal`
- `typedef _Key key_type`
- `typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > local_iterator`
- `typedef _Allocator::pointer pointer`
- `typedef _Allocator::reference reference`
- `typedef _Base::size_type size_type`
- `typedef _Base::value_type value_type`

Public Member Functions

- **unordered_multiset** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_multiset (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** ([initializer_list](#)< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- const _RehashPolicy & **__rehash_policy** () const
- void **__rehash_policy** (const _RehashPolicy &)
- _Value_allocator_type **_M_get_Value_allocator** () const
- local_iterator **begin** (size_type __n)
- const_local_iterator **begin** (size_type __n) const
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket** (const key_type &__k) const
- size_type **bucket_count** () const
- size_type **bucket_size** (size_type __n) const
- const_iterator **cbegin** () const
- const_local_iterator **cbegin** (size_type __n) const
- const_iterator **cend** () const
- const_local_iterator **cend** (size_type) const
- void **clear** ()
- size_type **count** (const key_type &__k) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- local_iterator **end** (size_type)
- const_local_iterator **end** (size_type) const
- [std::pair](#)< iterator, iterator > **equal_range** (const key_type &__k)
- [std::pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- size_type **erase** (const key_type &)
- iterator **erase** (const_iterator)
- iterator **erase** (const_iterator, const_iterator)
- const_iterator **find** (const key_type &__k) const
- iterator **find** (const key_type &__k)
- allocator_type **get_allocator** () const
- [_Insert_Return_Type](#) **insert** (const value_type &__v)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)

- void **insert** ([initializer_list](#)< value_type > __l)
- iterator **insert** (const_iterator, const value_type &__v)
- key_equal **key_eq** () const
- float **load_factor** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- [unordered_multiset](#) & **operator=** ([initializer_list](#)< value_type > __l)
- void **rehash** (size_type __n)
- size_type **size** () const
- void **swap** (_Hashtable &)

Friends

- struct **__detail::_Map_base**

5.726.1 Detailed Description

template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves. Meets the requirements of a [container](#), and [unordered associative container](#)

Parameters

Value Type of key objects.

Hash Hashing function object type, defaults to hash<Value>.

Pred Predicate function object type, defaults to equal_to<Value>.

Alloc Allocator type, defaults to allocator<Key>.

Definition at line 312 of file unordered_set.h.

The documentation for this class was generated from the following file:

- [unordered_set.h](#)

5.727 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Inherits std::__unordered_set< _Value, _Hash, _Pred, _Alloc >.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __detail::_Hashtable_const_iterator< value_type, __constant_iterators, __cache_hash_code > **const_iterator**
- typedef __detail::_Node_const_iterator< value_type, __constant_iterators, __cache_hash_code > **const_local_iterator**
- typedef _Allocator::const_pointer **const_pointer**
- typedef _Allocator::const_reference **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Base::hasher **hasher**
- typedef __detail::_Hashtable_iterator< value_type, __constant_iterators, __cache_hash_code > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Key **key_type**
- typedef __detail::_Node_iterator< value_type, __constant_iterators, __cache_hash_code > **local_iterator**
- typedef _Allocator::pointer **pointer**
- typedef _Allocator::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_set** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
unordered_set (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** ([initializer_list](#)< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eq=key_equal(), const allocator_type &__a=allocator_type())
- const _RehashPolicy & **__rehash_policy** () const
- void **__rehash_policy** (const _RehashPolicy &)
- _Value_allocator_type **_M_get_Value_allocator** () const
- local_iterator **begin** (size_type __n)
- const_local_iterator **begin** (size_type __n) const
- iterator **begin** ()
- const_iterator **begin** () const

- size_type **bucket** (const key_type &__k) const
- size_type **bucket_count** () const
- size_type **bucket_size** (size_type __n) const
- const_iterator **cbegin** () const
- const_local_iterator **cbegin** (size_type __n) const
- const_iterator **cend** () const
- const_local_iterator **cend** (size_type) const
- void **clear** ()
- size_type **count** (const key_type &__k) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- local_iterator **end** (size_type)
- const_local_iterator **end** (size_type) const
- [std::pair](#)< iterator, iterator > **equal_range** (const key_type &__k)
- [std::pair](#)< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- size_type **erase** (const key_type &)
- iterator **erase** (const_iterator)
- iterator **erase** (const_iterator, const_iterator)
- const_iterator **find** (const key_type &__k) const
- iterator **find** (const key_type &__k)
- allocator_type **get_allocator** () const
- [_Insert_Return_Type](#) **insert** (const value_type &__v)
- template<typename _InputIterator >
void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** ([initializer_list](#)< value_type > __l)
- iterator **insert** (const_iterator, const value_type &__v)
- key_equal **key_eq** () const
- float **load_factor** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- **unordered_set** & **operator=** ([initializer_list](#)< value_type > __l)
- void **rehash** (size_type __n)
- size_type **size** () const
- void **swap** (_Hashtable &)

Friends

- struct [__detail::_Map_base](#)

5.727.1 Detailed Description

```
template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_
to<_Value>, class _Alloc = std::allocator<_Value>> class std::unordered_set<
_Value, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves. Meets the requirements of a [container](#), and [unordered associative container](#)

Parameters

Value Type of key objects.

Hash Hashing function object type, defaults to `hash<Value>`.

Pred Predicate function object type, defaults to `equal_to<Value>`.

Alloc Allocator type, defaults to `allocator<Key>`.

Definition at line 247 of file `unordered_set.h`.

The documentation for this class was generated from the following file:

- [unordered_set.h](#)

5.728 std::uses_allocator< _Tp, _Alloc > Struct Template Reference

[allocator.uses.trait]

Inherits `integral_constant< bool, __uses_allocator_helper< _Tp, _Alloc >::value >`.

5.728.1 Detailed Description

```
template<typename _Tp, typename _Alloc> struct std::uses_allocator< _Tp, _
Alloc >
```

[allocator.uses.trait]

Definition at line 228 of file `allocator.h`.

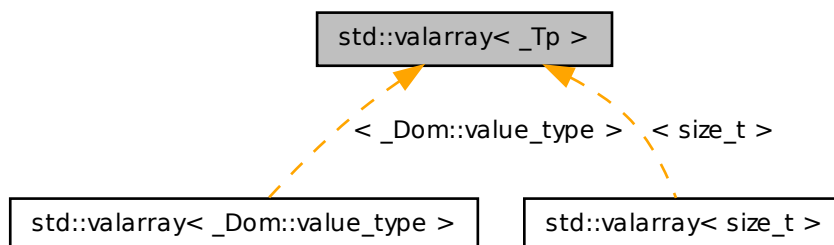
The documentation for this struct was generated from the following file:

- [allocator.h](#)

5.729 `std::valarray<_Tp>` Class Template Reference

Smart array designed to support numeric processing.

Inheritance diagram for `std::valarray<_Tp>`:



Public Types

- `typedef _Tp value_type`

Public Member Functions

- `valarray()`
- `valarray(size_t)`
- `valarray(const _Tp *__restrict__, size_t)`
- `valarray(const mask_array<_Tp> &)`
- `valarray(const indirect_array<_Tp> &)`
- `template<typename _Tp>`
`valarray(const _Tp *__restrict__ __p, size_t __n)`
- `valarray(const valarray &)`
- `valarray(initializer_list<_Tp>)`
- `template<class _Dom>`
`valarray(const _Expr<_Dom, _Tp> &__e)`
- `valarray(const _Tp &, size_t)`
- `valarray(const slice_array<_Tp> &)`
- `valarray(const gslice_array<_Tp> &)`
- `_Expr<_ValFunClos<_ValArray, _Tp>, _Tp> apply(_Tp func(_Tp)) const`

- `_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > apply` (`_Tp func(const _Tp &)`) const
- `valarray< _Tp > cshift` (int) const
- `_Tp max` () const
- `_Tp min` () const
- `_UnaryOp< __logical_not >::_Rt operator!` () const
- `valarray< _Tp > & operator%=` (const `_Tp` &)
- `valarray< _Tp > & operator%=` (const `valarray< _Tp >` &)
- `template<class _Dom >`
`valarray< _Tp > & operator%=` (const `_Expr< _Dom, _Tp >` &)
- `valarray< _Tp > & operator&=` (const `_Tp` &)
- `valarray< _Tp > & operator&=` (const `valarray< _Tp >` &)
- `template<class _Dom >`
`valarray< _Tp > & operator&=` (const `_Expr< _Dom, _Tp >` &)
- `valarray< _Tp > & operator*=` (const `_Tp` &)
- `valarray< _Tp > & operator*=` (const `valarray< _Tp >` &)
- `template<class _Dom >`
`valarray< _Tp > & operator*=` (const `_Expr< _Dom, _Tp >` &)
- `_UnaryOp< __unary_plus >::_Rt operator+` () const
- `valarray< _Tp > & operator+=` (const `_Tp` &)
- `valarray< _Tp > & operator+=` (const `valarray< _Tp >` &)
- `template<class _Dom >`
`valarray< _Tp > & operator+=` (const `_Expr< _Dom, _Tp >` &)
- `_UnaryOp< __negate >::_Rt operator-` () const
- `valarray< _Tp > & operator-=` (const `_Tp` &)
- `valarray< _Tp > & operator-=` (const `valarray< _Tp >` &)
- `template<class _Dom >`
`valarray< _Tp > & operator-=` (const `_Expr< _Dom, _Tp >` &)
- `valarray< _Tp > & operator/=` (const `_Tp` &)
- `valarray< _Tp > & operator/=` (const `valarray< _Tp >` &)
- `template<class _Dom >`
`valarray< _Tp > & operator/=` (const `_Expr< _Dom, _Tp >` &)
- `valarray< _Tp > & operator<=<=` (const `_Tp` &)
- `valarray< _Tp > & operator<=<=` (const `valarray< _Tp >` &)
- `template<class _Dom >`
`valarray< _Tp > & operator<=<=` (const `_Expr< _Dom, _Tp >` &)
- `valarray< _Tp > & operator=` (const `_Tp` &)
- `valarray< _Tp > & operator=` (const `gslice_array< _Tp >` &)
- `valarray< _Tp > & operator=` (const `slice_array< _Tp >` &)
- `valarray< _Tp > & operator=` (const `indirect_array< _Tp >` &)
- `valarray & operator=` (`initializer_list< _Tp >`)
- `template<class _Dom >`
`valarray< _Tp > & operator=` (const `_Expr< _Dom, _Tp >` &)

- `valarray< _Tp > & operator=` (const `mask_array< _Tp > &`)
- `valarray< _Tp > & operator=` (const `valarray< _Tp > &`)
- `valarray< _Tp > & operator>>=` (const `valarray< _Tp > &`)
- `template<class _Dom >`
`valarray< _Tp > & operator>>=` (const `_Expr< _Dom, _Tp > &`)
- `valarray< _Tp > & operator>>=` (const `_Tp &`)
- `gslice_array< _Tp > operator[]` (const `gslice &`)
- `_Tp & operator[]` (size_t)
- `_Expr< _SClos< _ValArray, _Tp >, _Tp > operator[]` (slice) const
- `valarray< _Tp > operator[]` (const `valarray< bool > &`) const
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > operator[]` (const `valarray< size_t > &`) const
- `slice_array< _Tp > operator[]` (slice)
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > operator[]` (const `gslice &`) const
- `indirect_array< _Tp > operator[]` (const `valarray< size_t > &`)
- `const _Tp & operator[]` (size_t) const
- `mask_array< _Tp > operator[]` (const `valarray< bool > &`)
- `valarray< _Tp > & operator^=` (const `valarray< _Tp > &`)
- `template<class _Dom >`
`valarray< _Tp > & operator^=` (const `_Expr< _Dom, _Tp > &`)
- `valarray< _Tp > & operator^=` (const `_Tp &`)
- `template<class _Dom >`
`valarray< _Tp > & operator|=` (const `_Expr< _Dom, _Tp > &`)
- `valarray< _Tp > & operator|=` (const `_Tp &`)
- `valarray< _Tp > & operator|=` (const `valarray< _Tp > &`)
- `_UnaryOp< __bitwise_not >::_Rt operator~` () const
- `void resize` (size_t __size, _Tp __c=_Tp())
- `valarray< _Tp > shift` (int) const
- `size_t size` () const
- `_Tp sum` () const

Friends

- `class _Array< _Tp >`

5.729.1 Detailed Description

`template<class _Tp> class std::valarray< _Tp >`

Smart array designed to support numeric processing. A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

Parameters

Tp Type of object in the array.

Definition at line 111 of file valarray.

5.729.2 Constructor & Destructor Documentation

5.729.2.1 `template<class _Tp> std::valarray< _Tp >::valarray (const _Tp * __restrict__, size_t)`

Construct an array initialized to the first n elements of t .

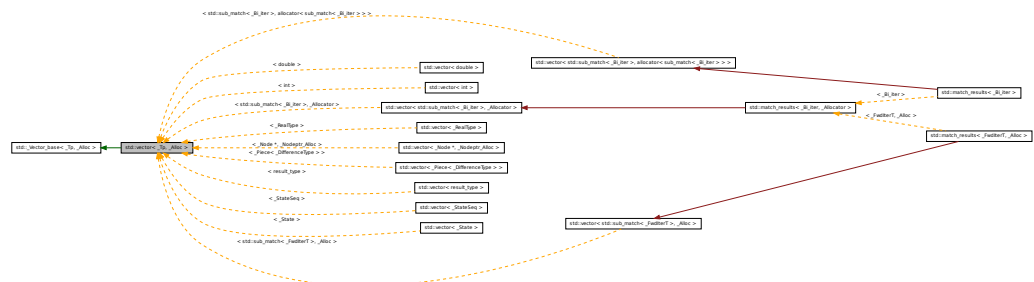
The documentation for this class was generated from the following file:

- [valarray](#)

5.730 `std::vector< _Tp, _Alloc >` Class Template Reference

A standard container which offers fixed time access to individual elements in any order.

Inheritance diagram for `std::vector< _Tp, _Alloc >`:



Public Types

- `typedef _Alloc allocator_type`
- `typedef __gnu_cxx::__normal_iterator< const_pointer, vector > const_iterator`
- `typedef _Tp_alloc_type::const_pointer const_pointer`
- `typedef _Tp_alloc_type::const_reference const_reference`

- typedef `std::reverse_iterator`< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, `vector` > **iterator**
- typedef _Tp_alloc_type::pointer **pointer**
- typedef _Tp_alloc_type::reference **reference**
- typedef `std::reverse_iterator`< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- `vector` ()
- `vector` (const allocator_type &__a)
- `vector` (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
`vector` (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- `vector` (const `vector` &__x)
- `vector` (size_type __n)
- `vector` (`vector` &&__x)
- `vector` (`initializer_list`< value_type > __l, const allocator_type &__a=allocator_type())
- `~vector` ()
- void `assign` (size_type __n, const value_type &__val)
- template<typename _InputIterator >
void `assign` (_InputIterator __first, _InputIterator __last)
- void `assign` (`initializer_list`< value_type > __l)
- reference `at` (size_type __n)
- const_reference `at` (size_type __n) const
- reference `back` ()
- const_reference `back` () const
- iterator `begin` ()
- const_iterator `begin` () const
- size_type `capacity` () const
- const_iterator `cbegin` () const
- const_iterator `cend` () const
- void `clear` ()
- `const_reverse_iterator` `crbegin` () const
- `const_reverse_iterator` `crend` () const
- _Tp * `data` ()
- const _Tp * `data` () const

- `template<typename... _Args>`
`iterator emplace (iterator __position, _Args &&...__args)`
- `template<typename... _Args>`
`void emplace_back (_Args &&...__args)`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase (iterator __position)`
- `const_reference front () const`
- `reference front ()`
- `iterator insert (iterator __position, value_type &&__x)`
- `template<typename _InputIterator >`
`void insert (iterator __position, _InputIterator __first, _InputIterator __last)`
- `void insert (iterator __position, initializer_list< value_type > __l)`
- `void insert (iterator __position, size_type __n, const value_type &__x)`
- `iterator insert (iterator __position, const value_type &__x)`
- `size_type max_size () const`
- `vector & operator= (initializer_list< value_type > __l)`
- `vector & operator= (vector &&__x)`
- `vector & operator= (const vector &__x)`
- `reference operator\[\] (size_type __n)`
- `const_reference operator\[\] (size_type __n) const`
- `void pop_back ()`
- `void push_back (value_type &&__x)`
- `void push_back (const value_type &__x)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `const_reverse_iterator rend () const`
- `reverse_iterator rend ()`
- `void reserve (size_type __n)`
- `void resize (size_type __new_size)`
- `void resize (size_type __new_size, const value_type &__x)`
- `void shrink_to_fit ()`
- `size_type size () const`
- `void swap (vector &__x)`

Protected Member Functions

- `_Tp_alloc_type::pointer` **`_M_allocate`** (`size_t __n`)
- `template<typename _ForwardIterator>`
`pointer` **`_M_allocate_and_copy`** (`size_type __n`, `_ForwardIterator __first`, `_ForwardIterator __last`)
- `template<typename _InputIterator>`
`void` **`_M_assign_aux`** (`_InputIterator __first`, `_InputIterator __last`, `std::input_iterator_tag`)
- `template<typename _ForwardIterator>`
`void` **`_M_assign_aux`** (`_ForwardIterator __first`, `_ForwardIterator __last`, `std::forward_iterator_tag`)
- `template<typename _Integer>`
`void` **`_M_assign_dispatch`** (`_Integer __n`, `_Integer __val`, `__true_type`)
- `template<typename _InputIterator>`
`void` **`_M_assign_dispatch`** (`_InputIterator __first`, `_InputIterator __last`, `__false_type`)
- `size_type` **`_M_check_len`** (`size_type __n`, `const char *__s`) `const`
- `void` **`_M_deallocate`** (`typename _Tp_alloc_type::pointer __p`, `size_t __n`)
- `void` **`_M_default_append`** (`size_type __n`)
- `void` **`_M_default_initialize`** (`size_type __n`)
- `void` **`_M_erase_at_end`** (`pointer __pos`)
- `void` **`_M_fill_assign`** (`size_type __n`, `const value_type &__val`)
- `void` **`_M_fill_initialize`** (`size_type __n`, `const value_type &__value`)
- `void` **`_M_fill_insert`** (`iterator __pos`, `size_type __n`, `const value_type &__x`)
- `_Tp_alloc_type &` **`_M_get_Tp_allocator`** ()
- `const _Tp_alloc_type &` **`_M_get_Tp_allocator`** () `const`
- `template<typename _InputIterator>`
`void` **`_M_initialize_dispatch`** (`_InputIterator __first`, `_InputIterator __last`, `__false_type`)
- `template<typename _Integer>`
`void` **`_M_initialize_dispatch`** (`_Integer __n`, `_Integer __value`, `__true_type`)
- `template<typename... _Args>`
`void` **`_M_insert_aux`** (`iterator __position`, `_Args &&...__args`)
- `template<typename _InputIterator>`
`void` **`_M_insert_dispatch`** (`iterator __pos`, `_InputIterator __first`, `_InputIterator __last`, `__false_type`)
- `template<typename _Integer>`
`void` **`_M_insert_dispatch`** (`iterator __pos`, `_Integer __n`, `_Integer __val`, `__true_type`)
- `void` **`_M_range_check`** (`size_type __n`) `const`
- `template<typename _ForwardIterator>`
`void` **`_M_range_initialize`** (`_ForwardIterator __first`, `_ForwardIterator __last`, `std::forward_iterator_tag`)

- `template<typename _InputIterator >`
`void _M_range_initialize (_InputIterator __first, _InputIterator __last,`
`std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_range_insert (iterator __pos, _ForwardIterator __first, _-`
`ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator >`
`void _M_range_insert (iterator __pos, _InputIterator __first, _InputIterator __-`
`last, std::input_iterator_tag)`
- `allocator_type get_allocator () const`

Protected Attributes

- `_Vector_impl _M_impl`

5.730.1 Detailed Description

`template<typename _Tp, typename _Alloc = std::allocator<_Tp>> class`
`std::vector< _Tp, _Alloc >`

A standard container which offers fixed time access to individual elements in any order. Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting (`[]`) access is also provided as with C-style arrays.

Definition at line 178 of file `stl_vector.h`.

5.730.2 Constructor & Destructor Documentation

5.730.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>`
`std::vector< _Tp, _Alloc >::vector () [inline]`

Default constructor creates no elements.

Definition at line 215 of file `stl_vector.h`.

5.730.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (const allocator_type & __a)
[inline, explicit]`

Creates a vector with no elements.

Parameters

a An allocator object.

Definition at line 223 of file stl_vector.h.

5.730.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (size_type __n) [inline,
explicit]`

Creates a vector with default constructed elements.

Parameters

n The number of elements to initially create.

This constructor fills the vector with *n* default constructed elements.

Definition at line 235 of file stl_vector.h.

5.730.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (size_type __n, const value_type
& __value, const allocator_type & __a = allocator_type())
[inline]`

Creates a vector with copies of an exemplar element.

Parameters

n The number of elements to initially create.

value An element to copy.

a An allocator.

This constructor fills the vector with *n* copies of *value*.

Definition at line 247 of file stl_vector.h.

5.730.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (const vector<_Tp, _Alloc> &
__x) [inline]`

Vector copy constructor.

Parameters

x A vector of identical element and allocator types.

The newly-created vector uses a copy of the allocation object used by *x*. All the elements of *x* are copied, but any extra memory in *x* (for fast expansion) will not be copied.

Definition at line 276 of file `stl_vector.h`.

5.730.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (vector<_Tp, _Alloc> && __x
) [inline]`

Vector move constructor.

Parameters

x A vector of identical element and allocator types.

The newly-created vector contains the exact contents of *x*. The contents of *x* are a valid, but unspecified vector.

Definition at line 292 of file `stl_vector.h`.

5.730.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (initializer_list< value_type
> __l, const allocator_type & __a = allocator_type())
[inline]`

Builds a vector from an initializer list.

Parameters

l An [initializer_list](#).

a An allocator.

Create a vector consisting of copies of the elements in the [initializer_list](#) *l*.

This will call the element type's copy constructor *N* times (where *N* is *l.size()*) and do no memory reallocation.

Definition at line 306 of file `stl_vector.h`.

```

5.730.2.8  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             template<typename _InputIterator > std::vector< _Tp, _Alloc
             >::vector ( _InputIterator __first, _InputIterator __last, const
             allocator_type & __a = allocator_type() ) [inline]

```

Builds a vector from a range.

Parameters

first An input iterator.

last An input iterator.

a An allocator.

Create a vector consisting of copies of the elements from [first,last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor *N* times (where *N* is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2*N* calls to the copy constructor, and log*N* memory reallocations.

Definition at line 332 of file stl_vector.h.

```

5.730.2.9  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             std::vector< _Tp, _Alloc >::~~vector ( ) [inline]

```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 347 of file stl_vector.h.

5.730.3 Member Function Documentation

```

5.730.3.1  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             template<typename _ForwardIterator > pointer std::vector< _Tp,
             _Alloc >::_M_allocate_and_copy ( size_type __n, _ForwardIterator
             __first, _ForwardIterator __last ) [inline, protected]

```

Memory expansion handler. Uses the member allocation function to obtain *n* bytes of memory, and then copies [first,last) into it.

Definition at line 1047 of file stl_vector.h.

Referenced by std::vector< _Tp, _Alloc >::operator=().

5.730.3.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::M_range_check (size_type __n)
const [inline, protected]`

Safety check used only from [at\(\)](#).

Definition at line 714 of file `stl_vector.h`.

5.730.3.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::vector<_Tp,
_Alloc>::assign (_InputIterator __first, _InputIterator __last)
[inline]`

Assigns a range to a vector.

Parameters

first An input iterator.

last An input iterator.

This function fills a vector with copies of the elements in the range `[first,last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 427 of file `stl_vector.h`.

5.730.3.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::assign (size_type __n, const
value_type & __val) [inline]`

Assigns a given value to a vector.

Parameters

n Number of elements to be assigned.

val Value to be assigned.

This function fills a vector with *n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 410 of file `stl_vector.h`.

5.730.3.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::assign (initializer_list< value_type
> __l) [inline]`

Assigns an initializer list to a vector.

Parameters

l An [initializer_list](#).

This function fills a vector with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 447 of file `std_vector.h`.

Referenced by `std::vector< std::sub_match< _Bi_iter >, _Allocator >::assign()`.

5.730.3.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::at (size_type __n)
[inline]`

Provides access to the data contained in the vector.

Parameters

n The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

[std::out_of_range](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws [out_of_range](#) if the check fails.

Definition at line 733 of file `std_vector.h`.

5.730.3.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::at (size_type __n)
const [inline]`

Provides access to the data contained in the vector.

Parameters

n The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

[*std::out_of_range*](#) If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws [*out_of_range*](#) if the check fails.

Definition at line 751 of file `stl_vector.h`.

5.730.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::back () const
[inline]`

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 786 of file `stl_vector.h`.

5.730.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc>::back () [inline]`

Returns a read/write reference to the data at the last element of the vector.

Definition at line 778 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::max()`, and `std::piecewise_constant_distribution<_RealType>::max()`.

5.730.3.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::begin () const
[inline]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results<_Bi_iter, _Allocator>`, `std::match_results<_Bi_iter>`, and `std::match_results<_FwdIterT, _Alloc>`.

Definition at line 470 of file `stl_vector.h`.

5.730.3.11 **template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector< _Tp, _Alloc >::begin () [inline]**

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 461 of file stl_vector.h.

Referenced by `std::vector< _Tp, _Alloc >::emplace()`, `std::vector< _Tp, _Alloc >::insert()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::vector< _Tp, _Alloc >::operator==()`, and `std::vector< std::sub_match< _Bi_iter >, _Allocator >::vector()`.

5.730.3.12 **template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector< _Tp, _Alloc >::capacity () const [inline]**

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 648 of file stl_vector.h.

Referenced by `std::vector< _Tp, _Alloc >::operator=()`.

5.730.3.13 **template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector< _Tp, _Alloc >::cbegin () const [inline]**

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results< _Bi_iter, _Allocator >`, `std::match_results< _Bi_iter >`, and `std::match_results< _FwdIterT, _Alloc >`.

Definition at line 534 of file stl_vector.h.

5.730.3.14 **template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector< _Tp, _Alloc >::cend () const [inline]**

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Reimplemented in `std::match_results< _Bi_iter, _Allocator >`, `std::match_results< _Bi_iter >`, and `std::match_results< _FwdIterT, _Alloc >`.

Definition at line 543 of file stl_vector.h.

5.730.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::clear () [inline]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1037 of file `stl_vector.h`.

5.730.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin ()
const [inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 552 of file `stl_vector.h`.

5.730.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector<_Tp, _Alloc>::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 561 of file `stl_vector.h`.

5.730.3.18 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Tp* std::vector<_Tp, _Alloc>::data () [inline]`

Returns a pointer such that [`data()`, `data() + size()`) is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 801 of file `stl_vector.h`.

5.730.3.19 `template<typename _Tp, typename _Alloc> template<typename...
_Args> vector<_Tp, _Alloc>::iterator std::vector<_Tp, _Alloc>
>::emplace (iterator __position, _Args &&... __args)`

Inserts an object in vector before specified iterator.

Parameters

position An iterator into the vector.

args Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.`

Definition at line 272 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::begin()`, and `std::vector< _Tp, _Alloc >::end()`.

5.730.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::vector< _Tp, _Alloc >::empty () const [inline]`

Returns true if the vector is empty. (Thus [begin\(\)](#) would equal [end\(\)](#).)

Reimplemented in [std::match_results< _Bi_iter, _Allocator >](#), [std::match_results< _-Bi_iter >](#), and [std::match_results< _FwdIterT, _Alloc >](#).

Definition at line 657 of file `stl_vector.h`.

5.730.3.21 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector< _Tp, _Alloc >::end () [inline]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 479 of file `stl_vector.h`.

Referenced by `std::vector< _Tp, _Alloc >::emplace()`, `std::vector< _Tp, _-Alloc >::erase()`, `std::vector< _Tp, _Alloc >::insert()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::vector< _Tp, _Alloc >::operator=()`, `std::operator==()`, and `std::vector< std::sub_match< _Bi_iter >, _Allocator >::vector()`.

5.730.3.22 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector< _Tp, _Alloc >::end () const [inline]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Reimplemented in [std::match_results< _Bi_iter, _Allocator >](#), [std::match_results< _-Bi_iter >](#), and [std::match_results< _FwdIterT, _Alloc >](#).

Definition at line 488 of file `stl_vector.h`.

5.730.3.23 `template<typename _Tp , typename _Alloc > vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc >::erase (iterator __position)`

Remove element at given position.

Parameters

position Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 134 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::end()`.

5.730.3.24 `template<typename _Tp , typename _Alloc > vector< _Tp, _Alloc >::iterator std::vector< _Tp, _Alloc >::erase (iterator __first, iterator __last)`

Remove a range of elements.

Parameters

first Iterator pointing to the first element to be erased.

last Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 146 of file `vector.tcc`.

References `std::vector< _Tp, _Alloc >::end()`.

5.730.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::front () [inline]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 762 of file stl_vector.h.

Referenced by `std::piecewise_linear_distribution< _RealType >::min()`, and `std::piecewise_constant_distribution< _RealType >::min()`.

5.730.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::front () const
[inline]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 770 of file stl_vector.h.

5.730.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator > void std::vector< _Tp,
_Alloc >::insert (iterator __position, _InputIterator __first,
_InputIterator __last) [inline]`

Inserts a range into the vector.

Parameters

position An iterator into the vector.

first An input iterator.

last An input iterator.

This function will insert copies of the data in the range [first,last) into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 960 of file stl_vector.h.

5.730.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (iterator __position,
value_type && __x) [inline]`

Inserts given rvalue into vector before specified iterator.

Parameters

position An iterator into the vector.
x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 906 of file `stl_vector.h`.

```
5.730.3.29  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::vector< _Tp, _Alloc >::insert ( iterator __position,
             initializer_list< value_type > __l ) [inline]
```

Inserts an [initializer_list](#) into the vector.

Parameters

position An iterator into the vector.
l An [initializer_list](#).

This function will insert copies of the data in the [initializer_list](#) *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 923 of file `stl_vector.h`.

Referenced by `std::vector< std::sub_match< _Bi_iter >, _Allocator >::insert()`.

```
5.730.3.30  template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
             void std::vector< _Tp, _Alloc >::insert ( iterator __position,
             size_type __n, const value_type & __x ) [inline]
```

Inserts a number of copies of given data into the vector.

Parameters

position An iterator into the vector.
n Number of elements to be inserted.
x Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 941 of file stl_vector.h.

5.730.3.31 `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc
>::iterator std::vector< _Tp, _Alloc >::insert (iterator __position,
const value_type & __x)`

Inserts given value into vector before specified iterator.

Parameters

position An iterator into the vector.

x Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using [std::list](#).

Definition at line 107 of file vector.tcc.

References `std::vector< _Tp, _Alloc >::begin()`, and `std::vector< _Tp, _Alloc >::end()`.

5.730.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::max_size () const
[inline]`

Returns the [size\(\)](#) of the largest possible vector.

Reimplemented in [std::match_results< _Bi_iter, _Allocator >](#), [std::match_results< _Bi_iter >](#), and [std::match_results< _FwdIterT, _Alloc >](#).

Definition at line 573 of file stl_vector.h.

5.730.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector& std::vector< _Tp, _Alloc >::operator= (vector< _Tp,
_Alloc > && __x) [inline]`

Vector move assignment operator.

Parameters

x A vector of identical element and allocator types.

The contents of *x* are moved into this vector (without copying). *x* is a valid, but unspecified vector.

Definition at line 371 of file `stl_vector.h`.

5.730.3.34 `template<typename _Tp, typename _Alloc > vector< _Tp, _Alloc > & std::vector< _Tp, _Alloc >::operator= (const vector< _Tp, _Alloc > & __x)`

Vector assignment operator.

Parameters

x A vector of identical element and allocator types.

All the elements of *x* are copied, but any extra memory in *x* (for fast expansion) will not be copied. Unlike the copy constructor, the allocator object is not copied.

Definition at line 157 of file `vector.tcc`.

References `std::_Destroy()`, `std::vector< _Tp, _Alloc >::_M_allocate_and_copy()`, `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::capacity()`, `std::vector< _Tp, _Alloc >::end()`, and `std::vector< _Tp, _Alloc >::size()`.

5.730.3.35 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector< _Tp, _Alloc >::operator= (initializer_list< value_type > __l) [inline]`

Vector list assignment operator.

Parameters

l An [initializer_list](#).

This function fills a vector with copies of the elements in the initializer list *l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 392 of file `stl_vector.h`.

5.730.3.36 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::operator[] (size_type
__n) const [inline]`

Subscript access to the data contained in the vector.

Parameters

n The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 708 of file `std_vector.h`.

5.730.3.37 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::operator[] (size_type __n)
[inline]`

Subscript access to the data contained in the vector.

Parameters

n The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and [out_of_range](#) lookups are not defined. (For checked lookups see [at\(\)](#).)

Definition at line 693 of file `std_vector.h`.

5.730.3.38 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::pop_back () [inline]`

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before [pop_back\(\)](#) is called.

Definition at line 855 of file stl_vector.h.

5.730.3.39 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::push_back (const value_type &
__x) [inline]`

Add data to the end of the vector.

Parameters

x Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 824 of file stl_vector.h.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

5.730.3.40 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc>::rbegin ()
[inline]`

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 497 of file stl_vector.h.

5.730.3.41 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin () const
[inline]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 506 of file stl_vector.h.

5.730.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc>::rend () [inline]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 515 of file stl_vector.h.

5.730.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rend () const
[inline]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 524 of file stl_vector.h.

5.730.3.44 `template<typename _Tp , typename _Alloc > void std::vector<
_Tp, _Alloc >::reserve (size_type __n)`

Attempt to preallocate enough memory for specified number of elements.

Parameters

n Number of elements required.

Exceptions

std::length_error If *n* exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

Definition at line 65 of file vector.tcc.

References `std::_Destroy()`.

5.730.3.45 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (size_type __new_size,
const value_type & __x) [inline]`

Resizes the vector to the specified number of elements.

Parameters

new_size Number of elements the vector should contain.

x Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 607 of file `stl_vector.h`.

5.730.3.46 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::resize (size_type __new_size)
[inline]`

Resizes the vector to the specified number of elements.

Parameters

new_size Number of elements the vector should contain.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 587 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink_and_double()`, `__gnu_parallel::multiway_merge_exact_splitting()`, and `__gnu_parallel::parallel_sort_mwms()`.

5.730.3.47 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::shrink_to_fit () [inline]`

A non-binding request to reduce [capacity\(\)](#) to [size\(\)](#).

Definition at line 639 of file `stl_vector.h`.

5.730.3.48 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector<_Tp, _Alloc>::size () const [inline]`

Returns the number of elements in the vector.

Reimplemented in [std::match_results<_Bi_iter, _Allocator>](#), [std::match_results<_Bi_iter>](#), and [std::match_results<_FwdIterT, _Alloc>](#).

Definition at line 568 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `__gnu_parallel::list_partition()`, `std::discrete_distribution<_IntType>::max()`, `std::vector<_Tp, _Alloc>::operator=()`, and `std::operator==()`.

5.730.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::swap (vector< _Tp, _Alloc > &
__x) [inline]`

Swaps data with another vector.

Parameters

x A vector of the same element and allocator types.

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global [std::swap\(\)](#) function is specialized such that `std::swap(v1,v2)` will feed to this function.

Definition at line 1017 of file `stl_vector.h`.

Referenced by `std::swap()`.

The documentation for this class was generated from the following files:

- [stl_vector.h](#)
- [vector.tcc](#)

5.731 `std::vector< bool, _Alloc >` Class Template Reference

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Inherits `std::_Bvector_base< _Alloc >`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Bit_const_iterator` **const_iterator**
- typedef `const bool *` **const_pointer**
- typedef `bool` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `bool` **value_type**

Public Member Functions

- **vector** (size_type __n, const bool &__value=bool(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
vector (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- **vector** (const [vector](#) &__x)
- **vector** (const allocator_type &__a)
- **vector** ([vector](#) &&__x)
- **vector** ([initializer_list](#)< bool > __l, const allocator_type &__a=allocator_type())
- void **assign** (size_type __n, const bool &__x)
- template<typename _InputIterator >
void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** ([initializer_list](#)< bool > __l)
- reference **at** (size_type __n)
- const_reference **at** (size_type __n) const
- reference **back** ()
- const_reference **back** () const
- const_iterator **begin** () const
- iterator **begin** ()
- size_type **capacity** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **clear** ()
- [const_reverse_iterator](#) **crbegin** () const
- [const_reverse_iterator](#) **crend** () const
- void **data** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __first, iterator __last)
- void **flip** ()
- reference **front** ()
- const_reference **front** () const
- allocator_type **get_allocator** () const
- void **insert** (iterator __position, size_type __n, const bool &__x)
- iterator **insert** (iterator __position, const bool &__x=bool())
- template<typename _InputIterator >
void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- void **insert** (iterator __p, [initializer_list](#)< bool > __l)

- `size_type max_size ()` const
- `vector & operator= (vector &&__x)`
- `vector & operator= (const vector &__x)`
- `vector & operator= (initializer_list< bool > __l)`
- reference `operator[]` (`size_type __n`)
- const_reference `operator[]` (`size_type __n`) const
- void `pop_back ()`
- void `push_back` (`bool __x`)
- `const_reverse_iterator rbegin ()` const
- `reverse_iterator rbegin ()`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend ()` const
- void `reserve` (`size_type __n`)
- void `resize` (`size_type __new_size`, `bool __x=bool()`)
- void `shrink_to_fit ()`
- `size_type size ()` const
- void `swap` (`vector &__x`)

Static Public Member Functions

- static void `swap` (reference `__x`, reference `__y`)

Protected Types

- typedef `_Alloc::template rebind< _Bit_type >::other _Bit_alloc_type`

Protected Member Functions

- `_Bit_type * _M_allocate` (`size_t __n`)
- `template<typename _InputIterator >`
void `_M_assign_aux` (`_InputIterator __first`, `_InputIterator __last`, `std::input_iterator_tag`)
- `template<typename _ForwardIterator >`
void `_M_assign_aux` (`_ForwardIterator __first`, `_ForwardIterator __last`, `std::forward_iterator_tag`)
- `template<typename _Integer >`
void `_M_assign_dispatch` (`_Integer __n`, `_Integer __val`, `__true_type`)
- `template<class _InputIterator >`
void `_M_assign_dispatch` (`_InputIterator __first`, `_InputIterator __last`, `__false_type`)
- `size_type _M_check_len` (`size_type __n`, `const char *__s`) const

- iterator **_M_copy_aligned** (const_iterator __first, const_iterator __last, iterator __result)
- void **_M_deallocate** ()
- void **_M_erase_at_end** (iterator __pos)
- void **_M_fill_assign** (size_t __n, bool __x)
- void **_M_fill_insert** (iterator __position, size_type __n, bool __x)
- _Bit_alloc_type & **_M_get_Bit_allocator** ()
- const _Bit_alloc_type & **_M_get_Bit_allocator** () const
- void **_M_initialize** (size_type __n)
- template<typename _InputIterator >
void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer >
void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
void **_M_initialize_range** (_InputIterator __first, _InputIterator __last, [std::input_iterator_tag](#))
- template<typename _ForwardIterator >
void **_M_initialize_range** (_ForwardIterator __first, _ForwardIterator __last, [std::forward_iterator_tag](#))
- void **_M_insert_aux** (iterator __position, bool __x)
- template<typename _Integer >
void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- template<typename _ForwardIterator >
void **_M_insert_range** (iterator __position, _ForwardIterator __first, _ForwardIterator __last, [std::forward_iterator_tag](#))
- template<typename _InputIterator >
void **_M_insert_range** (iterator __pos, _InputIterator __first, _InputIterator __last, [std::input_iterator_tag](#))
- void **_M_range_check** (size_type __n) const

Protected Attributes

- _Bvector_impl **_M_impl**

Friends

- class **hash**

5.731.1 Detailed Description

`template<typename _Alloc> class std::vector< bool, _Alloc >`

A specialization of vector for booleans which offers fixed time access to individual elements in any order. Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to bool. See DR96 for details.

See also

[vector](#) for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting (`[]`) access is also provided as with C-style arrays.

Definition at line 474 of file `stl_bvector.h`.

The documentation for this class was generated from the following files:

- [stl_bvector.h](#)
- [vector.tcc](#)

5.732 `std::weak_ptr<_Tp>` Class Template Reference

A smart pointer with weak semantics.

Inherits `std::__weak_ptr<_Tp>`.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `template<typename _Tp1 , typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type>`
`weak_ptr` (const [weak_ptr](#)<_Tp1> &__r)
- `template<typename _Tp1 , typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type>`
`weak_ptr` (const [shared_ptr](#)<_Tp1> &__r)
- `bool expired () const`

- [shared_ptr](#)< _Tp > **lock** () const
- template<typename _Tp1 >
[weak_ptr](#) & **operator=** (const [weak_ptr](#)< _Tp1 > &__r)
- template<typename _Tp1 >
[weak_ptr](#) & **operator=** (const [shared_ptr](#)< _Tp1 > &__r)
- template<typename _Tp1 >
bool **owner_before** (const __weak_ptr< _Tp1, _Lp > &__rhs) const
- template<typename _Tp1 >
bool **owner_before** (const __shared_ptr< _Tp1, _Lp > &__rhs) const
- void **reset** ()
- void **swap** (__weak_ptr &__s)
- long **use_count** () const

5.732.1 Detailed Description

template<typename _Tp> class std::weak_ptr< _Tp >

A smart pointer with weak semantics. With forwarding constructors and assignment operators.

Definition at line 387 of file shared_ptr.h.

The documentation for this class was generated from the following file:

- [shared_ptr.h](#)

5.733 std::weibull_distribution< _RealType > Class Template Reference

A [weibull_distribution](#) random number distribution.

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **weibull_distribution** (_RealType __a=_RealType(1), _RealType __b=_RealType(1))
- **weibull_distribution** (const [param_type](#) &__p)
- _RealType **a** () const
- _RealType **b** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- void **param** (const [param_type](#) &__param)
- [param_type](#) **param** () const
- void **reset** ()

5.733.1 Detailed Description

template<typename _RealType = double> class std::weibull_distribution< _RealType >

A [weibull_distribution](#) random number distribution. The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)$$

Definition at line 4338 of file random.h.

5.733.2 Member Typedef Documentation

5.733.2.1 template<typename _RealType = double> typedef _RealType std::weibull_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 4345 of file random.h.

5.733.3 Member Function Documentation

5.733.3.1 `template<typename _RealType = double> _RealType
std::weibull_distribution< _RealType >::a () const [inline]`

Return the a parameter of the distribution.

Definition at line 4396 of file random.h.

5.733.3.2 `template<typename _RealType = double> _RealType
std::weibull_distribution< _RealType >::b () const [inline]`

Return the b parameter of the distribution.

Definition at line 4403 of file random.h.

5.733.3.3 `template<typename _RealType = double> result_type
std::weibull_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4432 of file random.h.

5.733.3.4 `template<typename _RealType = double> result_type
std::weibull_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4425 of file random.h.

5.733.3.5 `template<typename _RealType = double> template<typename
_UniformRandomNumberGenerator > result_type
std::weibull_distribution< _RealType >::operator() (
_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 4440 of file random.h.

References `std::weibull_distribution< _RealType >::operator()()`, and `std::weibull_distribution< _RealType >::param()`.

Referenced by `std::weibull_distribution< _RealType >::operator()()`.

5.734 `std::weibull_distribution<_RealType>::param_type` Struct Reference

5.733.3.6 `template<typename _RealType = double> void
std::weibull_distribution<_RealType>::param (const param_type
& __param) [inline]`

Sets the parameter set of the distribution.

Parameters

`__param` The new parameter set of the distribution.

Definition at line 4418 of file `random.h`.

5.733.3.7 `template<typename _RealType = double> param_type
std::weibull_distribution<_RealType>::param () const
[inline]`

Returns the parameter set of the distribution.

Definition at line 4410 of file `random.h`.

Referenced by `std::weibull_distribution<_RealType>::operator()()`, `std::operator==()`, and `std::operator>>()`.

5.733.3.8 `template<typename _RealType = double> void
std::weibull_distribution<_RealType>::reset () [inline]`

Resets the distribution state.

Definition at line 4389 of file `random.h`.

The documentation for this class was generated from the following files:

- [random.h](#)
- [random.tcc](#)

5.734 `std::weibull_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `weibull_distribution<_RealType>` `distribution_type`

Public Member Functions

- **param_type** (_RealType __a=_RealType(1), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

5.734.1 Detailed Description

template<typename _RealType = double> struct std::weibull_distribution< _RealType >::param_type

Parameter type.

Definition at line 4347 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

Chapter 6

File Documentation

6.1 algo.h File Reference

Parallel STL function calls corresponding to the [stl_algo.h](#) header.

Classes

- struct [std::__parallel::__CRandNumber< _MustBeInt >](#)
Functor wrapper for std::rand().

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Functions

- template<typename _RAIter >
_RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _FIterator, typename _IteratorTag >
_FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _IteratorTag)
- template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >
_FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)

- `template<typename _RAIter, typename _BinaryPredicate >`
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter`
`__end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Predicate >`
`iterator_traits< _RAIter >::difference_type std::__parallel::__count_if -`
`switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random -`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu -`
`parallel::__parallel_unbalanced)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count_if_switch`
`(_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type std::__parallel::__count -`
`switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random -`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu -`
`parallel::__parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count_switch (-`
`_Iter __begin, _Iter __end, const _Tp &__value, _IteratorTag)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _`
`IteratorTag >`
`_RAIter std::__parallel::__find_first_of_switch (_RAIter __begin1, _RAIter`
`__end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp,`
`random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _`
`IteratorTag1, typename _IteratorTag2 >`
`_Iter std::__parallel::__find_first_of_switch (_Iter __begin1, _Iter __`
`end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _`
`IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2`
`>`
`_Iter std::__parallel::__find_first_of_switch (_Iter __begin1, _Iter __end1,`
`_FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`_Iter std::__parallel::__find_if_switch (_Iter __begin, _Iter __end, _`
`Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::__parallel::__find_if_switch (_RAIter __begin, _RAIter __end,`
`_Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Iter std::__parallel::__find_switch (_Iter __begin, _Iter __end, const _Tp`
`&__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter std::__parallel::__find_switch (_RAIter __begin, _RAIter __end,`
`const _Tp &__val, random_access_iterator_tag)`

- `template<typename _Iter , typename _Function , typename _IteratorTag >`
`_Function std::parallel::for_each_switch (_Iter __begin, _Iter __end, _-`
`Function __f, _IteratorTag)`
- `template<typename _RAIter , typename _Function >`
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter _-`
`__end, _Function __f, random_access_iterator_tag, __gnu_parallel::Parallelism _-`
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _OutputIterator , typename _Size , typename _Generator , typename _-`
`IteratorTag >`
`_OutputIterator std::parallel::generate_n_switch (_OutputIterator __-`
`begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter , typename _Size , typename _Generator >`
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n,`
`_Generator __gen, random_access_iterator_tag, __gnu_parallel::Parallelism _-`
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIterator , typename _Generator , typename _IteratorTag >`
`void std::parallel::generate_switch (_FIterator __begin, _FIterator __end,`
`_Generator __gen, _IteratorTag)`
- `template<typename _RAIter , typename _Generator >`
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _-`
`Generator __gen, random_access_iterator_tag, __gnu_parallel::Parallelism _-`
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`
`_FIterator std::parallel::max_element_switch (_FIterator __begin, _-`
`FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter`
`__end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::Parallelism _-`
`__parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Compare ,`
`typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >`
`_OutputIterator std::parallel::merge_switch (_Iter1 __begin1, _Iter1 _-`
`__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare`
`__comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Compare >`
`_OutputIterator std::parallel::merge_switch (_Iter1 __begin1, _Iter1 _-`
`__end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare`
`__comp, random_access_iterator_tag, random_access_iterator_tag, random_-`
`access_iterator_tag)`
- `template<typename _FIterator , typename _Compare , typename _IteratorTag >`
`_FIterator std::parallel::min_element_switch (_FIterator __begin, _-`
`FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter , typename _Compare >`
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter`

```

__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)

```

- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`
`_FIterator std::__parallel::__partition_switch (_FIterator __begin, _FIterator`
`__end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::__parallel::__partition_switch (_RAIter __begin, _RAIter __`
`end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`
`void std::__parallel::__replace_if_switch (_FIterator __begin, _FIterator __`
`end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`
`void std::__parallel::__replace_if_switch (_RAIter __begin, _RAIter __`
`end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_`
`tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_`
`balanced)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`
`void std::__parallel::__replace_switch (_FIterator __begin, _FIterator __end,`
`const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`void std::__parallel::__replace_switch (_RAIter __begin, _RAIter __end,`
`const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_`
`tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_`
`balanced)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate,`
`typename _IteratorTag >`
`_FIterator std::__parallel::__search_n_switch (_FIterator __begin, _FIterator`
`__end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`
`_IteratorTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RAIter std::__parallel::__search_n_switch (_RAIter __begin, _RAIter __`
`end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`
`random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 std::__parallel::__search_switch (_RAIter1 __begin1, _RAIter1 __`
`end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag,`
`random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename`
`_IteratorTag2 >`
`_FIterator1 std::__parallel::__search_switch (_FIterator1 __begin1, _`
`FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1,`
`_IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`
`_RAIter1 std::__parallel::__search_switch (_RAIter1 __begin1, _RAIter1`

- __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >
_FIterator1 **std::__parallel::search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)
 - template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **std::__parallel::set_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
 - template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >
_Output_RAIter **std::__parallel::set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
 - template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **std::__parallel::set_intersection_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
 - template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >
_Output_RAIter **std::__parallel::set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
 - template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **std::__parallel::set_symmetric_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
 - template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >
_Output_RAIter **std::__parallel::set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
 - template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >
_OutputIterator **std::__parallel::set_union_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)

- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter std::__parallel::__set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`
`_RAIter2 std::__parallel::__transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_RAIter2 std::__parallel::__transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`
`_RAIter3 std::__parallel::__transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`
`_OutputIterator std::__parallel::__transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator std::__parallel::__unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`
`RandomAccessOutputIterator std::__parallel::__unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::__adjacent_find (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator std::__parallel::__adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::__adjacent_find (_FIterator __begin, _FIterator __end)`

- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator std::__parallel::adjacent_find (_FIterator __begin, _FIterator __-`
`end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __-`
`begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __-`
`begin, _Iter __end, const _Tp &__value, __gnu_parallel::_Parallelism __-`
`parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __-`
`begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __-`
`begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __-`
`begin, _Iter __end, _Predicate __pred, __gnu_parallel::_Parallelism __-`
`parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __-`
`begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, __-`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`
`__begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`
`__begin2, _FIterator __end2)`
- `template<typename _Iter, typename _FIterator >`
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator`
`__begin2, _FIterator __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter std::__parallel::find_first_of (_Iter __begin1, _Iter __end1, _-`
`FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu-`
`parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred,`
`__gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _Predicate >`
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iterator, typename _Function >`
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`
`Function __f, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function`
`__f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`
`Function __f)`
- `template<typename _FIterator, typename _Generator >`
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`
`Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Generator >`
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`
`Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`
`void std::__parallel::generate (_FIterator __begin, _FIterator __end, _-`
`Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`
`_n, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`
`_n, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::__parallel::generate_n (_OutputIterator __begin, _Size _-`
`_n, _Generator __gen)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __-`
`end)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp, __gnu_parallel::sequential_tag)`

- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __-`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`
`Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp,`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`
`Iter2 __begin2, _Iter2 __end2, _OutputIterator __result)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__parallel::merge (_Iter1 __begin1, _Iter1 __end1, _-`
`Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end,`
`_Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end)`

- `template<typename _RAIter >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`
`__end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`
`RAIter __end, _Compare __comp)`
- `template<typename _FIterator, typename _Predicate >`
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred)`
- `template<typename _FIterator, typename _Predicate >`
`_FIterator std::__parallel::partition (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`
`RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`
`RandomNumberGenerator &&__rand)`
- `template<typename _FIterator, typename _Tp >`
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp`
`&__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Tp >`
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _-`
`Tp &__old_value, const _Tp &__new_value, __gnu_parallel::Parallelism _-`
`parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`
`void std::__parallel::replace (_FIterator __begin, _FIterator __end, const _Tp`
`&__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)`

- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred, const _Tp &__new_value, __gnu_parallel::Parallelism __-`
`parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_FIterator __begin, _FIterator __end, _-`
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 std::__parallel::search (_FIterator1 __begin1, _FIterator1 __end1,`
`_FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_`
`parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`
`Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_`
`parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`
`Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator std::__parallel::search_n (_FIterator __begin, _FIterator __end, _-`
`Integer __count, const _Tp &__val)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __-`
`pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_`
`parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_difference (_IIter1 __begin1, _IIter1 __-`
`end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __-`
`pred, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_difference (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __`
`pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __`
`pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_intersection (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _`
`Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _`
`Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_symmetric_difference (_Iter1 __begin1,`
`_Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, _Predicate __pred, __`
`gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __`
`end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::__parallel::set_union (_Iter1 __begin1, _Iter1 __end1,`
`_Iter2 __begin2, _Iter2 __end2, _OutputIterator __out)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`
`comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`
`comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::multiway_mergesort_exact_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`
`comp)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::multiway_mergesort_sampling_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`
`__comp, __gnu_parallel::sequential_tag)`

- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`
`__comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare`
`__comp)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_`
`parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __-`
`end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_`
`parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`
`BinaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter1 __begin1, _IIter1 __end1,`
`_IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __end, _`
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`
`BinaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter1 __begin1, _IIter1 __end1,`
`_IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __`
`gnu_parallel::sequential_tag)`
- `template<typename _IIter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_IIter __begin, _IIter __end,`
`_OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::`
`Parallelism __parallelism_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _`
`BinaryOperation >`

```

_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1,
      _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)
• template<typename _Iter, typename _OutputIterator >
  _OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,
      _OutputIterator __out)
• template<typename _Iter, typename _OutputIterator, typename _Predicate >
  _OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,
      _OutputIterator __out, _Predicate __pred)
• template<typename _Iter, typename _OutputIterator >
  _OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,
      _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter, typename _OutputIterator, typename _Predicate >
  _OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1,
      _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)

```

6.1.1 Detailed Description

Parallel STL function calls corresponding to the [stl_algo.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algo.h](#).

6.2 `algbase.h` File Reference

Parallel STL function calls corresponding to the [stl_algbase.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Functions

- template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >
 bool **std::__parallel::__lexicographical_compare_switch** (_Iter1 __begin1,

```

    _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _
    IteratorTag1, _IteratorTag2)
• template<typename _RAIter1, typename _RAIter2, typename _Predicate >
  bool std::__parallel::lexicographical_compare_switch (_RAIter1 __
  begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate
  __pred, random_access_iterator_tag, random_access_iterator_tag)
• template<typename _RAIter1, typename _RAIter2, typename _Predicate >
  pair< _RAIter1, _RAIter2 > std::__parallel::mismatch_switch (_RAIter1
  __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_
  access_iterator_tag, random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1,
  typename _IteratorTag2 >
  pair< _Iter1, _Iter2 > std::__parallel::mismatch_switch (_Iter1 __
  begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, _IteratorTag1, _
  IteratorTag2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2 >
  bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __
  begin2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  _Predicate __pred)
• template<typename _Iter1, typename _Iter2 >
  bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,
  \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __
  end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)
• template<typename _Iter1, typename _Iter2 >
  bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __
  end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2 >
  bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __
  end1, _Iter2 __begin2, _Iter2 __end2)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_
  parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate >
  pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1
  __end1, _Iter2 __begin2, _Predicate __pred)

```


- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`
`__end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1`
`__end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`

6.2.1 Detailed Description

Parallel STL function calls corresponding to the [stl_algobase.h](#) header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [algobase.h](#).

6.3 algorithm File Reference

6.3.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [algorithm](#).

6.4 algorithm File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair< _InputIterator, _OutputIterator > __gnu_cxx::__copy_n (_InputIterator`
`__first, _Size __count, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RAIterator, typename _Size, typename _OutputIterator >`
`pair< _RAIterator, _OutputIterator > __gnu_cxx::__copy_n (_RAIterator __`
`first, _Size __count, _OutputIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::__lexicographical_compare_3way (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`

- `int __gnu_cxx::__lexicographical_compare_3way` (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)
- `int __gnu_cxx::__lexicographical_compare_3way` (const char *__first1, const char *__last1, const char *__first2, const char *__last2)
- `template<typename _Tp, typename _Compare >`
`const _Tp & __gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- `template<typename _Tp >`
`const _Tp & __gnu_cxx::__median` (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`
`_RandomAccessIterator __gnu_cxx::__random_sample` (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`
`_RandomAccessIterator __gnu_cxx::__random_sample` (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`pair<_InputIterator, _OutputIterator > __gnu_cxx::__copy_n` (_InputIterator __first, _Size __count, _OutputIterator __result)
- `template<typename _InputIterator, typename _Tp, typename _Size >`
`void __gnu_cxx::__count` (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)
- `template<typename _InputIterator, typename _Predicate, typename _Size >`
`void __gnu_cxx::__count_if` (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)
- `template<typename _RandomAccessIterator >`
`bool __gnu_cxx::__is_heap` (_RandomAccessIterator __first, _RandomAccessIterator __last)
- `template<typename _RandomAccessIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::__is_heap` (_RandomAccessIterator __first, _RandomAccessIterator __last, _StrictWeakOrdering __comp)
- `template<typename _ForwardIterator, typename _StrictWeakOrdering >`
`bool __gnu_cxx::__is_sorted` (_ForwardIterator __first, _ForwardIterator __last, _StrictWeakOrdering __comp)
- `template<typename _ForwardIterator >`
`bool __gnu_cxx::__is_sorted` (_ForwardIterator __first, _ForwardIterator __last)
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::__lexicographical_compare_3way` (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`

```

_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __-
_first, _InputIterator __last, _RandomAccessIterator __out_first, __-
_RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
• template<typename _InputIterator, typename _RandomAccessIterator >
_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __-
_first, _InputIterator __last, _RandomAccessIterator __out_first, __-
_RandomAccessIterator __out_last)
• template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename
_RandomNumberGenerator >
_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, __-
_FowardIterator __last, _OutputIterator __out, const _Distance __n, __-
_RandomNumberGenerator &__rand)
• template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >
_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, __-
_FowardIterator __last, _OutputIterator __out, const _Distance __n)

```

6.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/algorithm](#).

6.5 algorithm File Reference

6.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [parallel/algorithm](#).

6.6 algorithmfwd.h File Reference

Namespaces

- namespace [std](#)

Functions

- template<typename _Filter >
_Filter **std::adjacent_find** (_Filter, _Filter)

- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`
`bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`
`bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`
`bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2 >`
`_BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter >`
`_OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::count (_Iter, _Iter, const _Tp &)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::count_if (_Iter, _Iter, _-`
`Predicate)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _-`
`BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _-`
`Compare)`
- `template<typename _Filter, typename _Tp >`
`void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp >`
`_OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::find (_Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`

- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::find_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct >`
`_Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator >`
`void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter >`
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter >`
`bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`_RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`
`bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _Filter >`
`bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`
`_Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2 >`
`void std::iter_swap (_Filter1, _Filter2)`

- `template<typename _Iter1, typename _Iter2 >`
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _-`
`Compare)`
- `template<typename _Filter, typename _Tp >`
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAIter >`
`void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`
`&__b)`

- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp`
`&__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _FIter >`
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter)`
- `template<typename _FIter, typename _Compare >`
`pair< _FIter, _FIter > std::minmax_element (_FIter, _FIter, _Compare)`
- `template<typename _IIter1, typename _IIter2 >`
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2)`
- `template<typename _IIter1, typename _IIter2, typename _BinaryPredicate >`
`pair< _IIter1, _IIter2 > std::mismatch (_IIter1, _IIter1, _IIter2, _-`
`BinaryPredicate)`
- `template<typename _BIter >`
`bool std::next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _IIter, typename _Predicate >`
`bool std::none_of (_IIter, _IIter, _Predicate)`
- `template<typename _RAIter >`
`void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _IIter, typename _RAIter, typename _Compare >`
`_RAIter std::partial_sort_copy (_IIter, _IIter, _RAIter, _RAIter, _Compare)`
- `template<typename _IIter, typename _RAIter >`
`_RAIter std::partial_sort_copy (_IIter, _IIter, _RAIter, _RAIter)`
- `template<typename _BIter, typename _Predicate >`
`_BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _IIter, typename _OIter1, typename _OIter2, typename _Predicate >`
`pair< _OIter1, _OIter2 > std::partition_copy (_IIter, _IIter, _OIter1, _OIter2,`
`_Predicate)`
- `template<typename _FIter, typename _Predicate >`
`_FIter std::partition_point (_FIter, _FIter, _Predicate)`
- `template<typename _RAIter >`
`void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::pop_heap (_RAIter, _RAIter, _Compare)`

- `template<typename _BIter >`
`bool std::prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool std::prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _RAIter >`
`void std::push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter, typename _Generator >`
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RAIter >`
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _FIter, typename _Tp >`
`_FIter std::remove (_FIter, _FIter, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Tp >`
`_OIter std::remove_copy (_IIter, _IIter, _OIter, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Predicate >`
`_OIter std::remove_copy_if (_IIter, _IIter, _OIter, _Predicate)`
- `template<typename _FIter, typename _Predicate >`
`_FIter std::remove_if (_FIter, _FIter, _Predicate)`
- `template<typename _FIter, typename _Tp >`
`void std::replace (_FIter, _FIter, const _Tp &, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Tp >`
`_OIter std::replace_copy (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _IIter, typename _OIter, typename _Predicate, typename _Tp >`
`_OIter std::replace_copy_if (_IIter, _IIter, _OIter, _Predicate, const _Tp &)`
- `template<typename _FIter, typename _Predicate, typename _Tp >`
`void std::replace_if (_FIter, _FIter, _Predicate, const _Tp &)`
- `template<typename _BIter >`
`void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter >`
`_OIter std::reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _FIter >`
`void std::rotate (_FIter, _FIter, _FIter)`
- `template<typename _FIter, typename _OIter >`
`_OIter std::rotate_copy (_FIter, _FIter, _FIter, _OIter)`
- `template<typename _FIter1, typename _FIter2 >`
`_FIter1 std::search (_FIter1, _FIter1, _FIter2, _FIter2)`
- `template<typename _FIter1, typename _FIter2, typename _BinaryPredicate >`
`_FIter1 std::search (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)`
- `template<typename _FIter, typename _Size, typename _Tp, typename _BinaryPredicate >`
`_FIter std::search_n (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _FIter, typename _Size, typename _Tp >`
`_FIter std::search_n (_FIter, _FIter, _Size, const _Tp &)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _RAIter, typename _UGenerator >`
`void std::shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RAIter, typename _Compare >`
`void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter >`
`void std::sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`
`_BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter, typename _Compare >`
`void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void std::stable_sort (_RAIter, _RAIter)`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b)`
- `template<typename _Tp, size_t _Nm >`
`void std::swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`
`_OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`

- `template<typename _Filter >`
`_Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >`
`_OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp >`
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

6.6.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [bits/algorithmfwd.h](#).

6.7 algorithmfwd.h File Reference

Namespaces

- namespace [std](#)
- namespace [std::__parallel](#)

Functions

- `template<typename _Filter, typename _IterTag >`
`_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _RAIter >`
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _RAIter, typename _BiPredicate >`
`_RAIter std::__parallel::__adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >`
`_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`

- `template<typename _RAIter, typename _Predicate >`
`iterator_traits< _RAIter >::difference_type std::parallel::count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`iterator_traits< _Iter >::difference_type std::parallel::count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type std::parallel::count_switch (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _FIter, typename _IterTag1, typename _IterTag2 >`
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _FIter, _FIter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag >`
`_RAIter std::parallel::find_first_of_switch (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`_Iter std::parallel::find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`_Iter std::parallel::find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`
`_Function std::parallel::for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _RAIter, typename _Function >`
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`
`_OIter std::parallel::generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n,`

- ```

 _Generator __gen, random_access_iterator_tag, __gnu_parallel::Parallelism _
 _parallelism_tag=__gnu_parallel::parallel_balanced)

```
- `template<typename _Filter, typename _Generator, typename _IterTag >`  
`void std::__parallel::generate_switch (_Filter, _Filter, _Generator, _IterTag)`
  - `template<typename _RAIter, typename _Generator >`  
`void std::__parallel::generate_switch (_RAIter __begin, _RAIter __end, _`  
`Generator __gen, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism _`  
`parallelism_tag=__gnu_parallel::parallel_balanced)`
  - `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`  
`bool std::__parallel::lexicographical_compare_switch (_Iter1, _Iter1, _`  
`Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare_switch (_RAIter1 __`  
`begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate`  
`__pred, random_access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::__parallel::max_element_switch (_Filter, _Filter, _Compare, _`  
`IterTag)`
  - `template<typename _RAIter, typename _Compare >`  
`_RAIter std::__parallel::max_element_switch (_RAIter __begin, _RAIter`  
`__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::`  
`Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
  - `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename`  
`_IterTag1, typename _IterTag2, typename _IterTag3 >`  
`_OIter std::__parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _`  
`OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
  - `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2,`  
`_OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag,`  
`random_access_iterator_tag)`
  - `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::__parallel::min_element_switch (_Filter, _Filter, _Compare, _`  
`IterTag)`
  - `template<typename _RAIter, typename _Compare >`  
`_RAIter std::__parallel::min_element_switch (_RAIter __begin, _RAIter`  
`__end, _Compare __comp, random_access_iterator_tag, \_\_gnu\_parallel::`  
`Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
  - `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::__parallel::mismatch_switch (_RAIter1`  
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random-`  
`access_iterator_tag, random_access_iterator_tag)`
  - `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`

- pair<\_Iter1, \_Iter2> **std::parallel::mismatch\_switch** (\_Iter1, \_Iter1, \_Iter2, \_Predicate, \_IterTag1, \_IterTag2)
- template<typename \_Filter, typename \_Predicate, typename \_IterTag>  
\_Filter **std::parallel::partition\_switch** (\_Filter, \_Filter, \_Predicate, \_IterTag)
  - template<typename \_RAIter, typename \_Predicate>  
\_RAIter **std::parallel::partition\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Predicate \_\_pred, random\_access\_iterator\_tag)
  - template<typename \_Filter, typename \_Predicate, typename \_Tp, typename \_IterTag>  
void **std::parallel::replace\_if\_switch** (\_Filter, \_Filter, \_Predicate, const \_Tp &, \_IterTag)
  - template<typename \_RAIter, typename \_Predicate, typename \_Tp>  
void **std::parallel::replace\_if\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Predicate \_\_pred, const \_Tp & \_\_new\_value, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag=\_\_gnu\_parallel::parallel\_balanced)
  - template<typename \_Filter, typename \_Tp, typename \_IterTag>  
void **std::parallel::replace\_switch** (\_Filter, \_Filter, const \_Tp &, const \_Tp &, \_IterTag)
  - template<typename \_RAIter, typename \_Tp>  
void **std::parallel::replace\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, const \_Tp & \_\_old\_value, const \_Tp & \_\_new\_value, random\_access\_iterator\_tag, [\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag=\_\_gnu\_parallel::parallel\_balanced)
  - template<typename \_RAIter, typename \_Integer, typename \_Tp, typename \_BiPredicate>  
\_RAIter **std::parallel::search\_n\_switch** (\_RAIter, \_RAIter, \_Integer, const \_Tp &, \_BiPredicate, random\_access\_iterator\_tag)
  - template<typename \_Filter, typename \_Integer, typename \_Tp, typename \_BiPredicate, typename \_IterTag>  
\_Filter **std::parallel::search\_n\_switch** (\_Filter, \_Filter, \_Integer, const \_Tp &, \_BiPredicate, \_IterTag)
  - template<typename \_RAIter1, typename \_RAIter2>  
\_RAIter1 **std::parallel::search\_switch** (\_RAIter1 \_\_begin1, \_RAIter1 \_\_end1, \_RAIter2 \_\_begin2, \_RAIter2 \_\_end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
  - template<typename \_Filter1, typename \_Filter2, typename \_IterTag1, typename \_IterTag2>  
\_Filter1 **std::parallel::search\_switch** (\_Filter1, \_Filter1, \_Filter2, \_Filter2, \_IterTag1, \_IterTag2)
  - template<typename \_RAIter1, typename \_RAIter2, typename \_BiPredicate>  
\_RAIter1 **std::parallel::search\_switch** (\_RAIter1, \_RAIter1, \_RAIter2, \_RAIter2, \_BiPredicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)
  - template<typename \_Filter1, typename \_Filter2, typename \_BiPredicate, typename \_IterTag1, typename \_IterTag2>

```

_Filter1 std::__parallel::__search_switch (_Filter1, _Filter1, _Filter2, _Filter2,
_BiPredicate, _IterTag1, _IterTag2)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::__parallel::__set_difference_switch (_Iter1, _Iter1, _Iter2, _-
Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
• template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename
_Predicate >
_Output_RAIter std::__parallel::__set_difference_switch (_RAIter1 _-
begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _-
Output_RAIter __result, _Predicate __pred, random_access_iterator_tag,
random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::__parallel::__set_intersection_switch (_Iter1, _Iter1, _Iter2, _-
Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
• template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _-
Predicate >
_Output_RAIter std::__parallel::__set_intersection_switch (_RAIter1 _-
begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_-
RAIter __result, _Predicate __pred, random_access_iterator_tag, random_-
access_iterator_tag, random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::__parallel::__set_symmetric_difference_switch (_Iter1, _Iter1,
_Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
• template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename
_Predicate >
_Output_RAIter std::__parallel::__set_symmetric_difference_switch (_-
RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2,
_Output_RAIter __result, _Predicate __pred, random_access_iterator_tag,
random_access_iterator_tag, random_access_iterator_tag)
• template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename
_IterTag1, typename _IterTag2, typename _IterTag3 >
_OIter std::__parallel::__set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2,
_OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
• template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _-
Predicate >
_Output_RAIter std::__parallel::__set_union_switch (_RAIter1 __begin1, _-
RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter
__result, _Predicate __pred, random_access_iterator_tag, random_access_-
iterator_tag, random_access_iterator_tag)
• template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1,
typename _IterTag2 >
_OIter std::__parallel::__transform1_switch (_Iter, _Iter, _OIter, _-
UnaryOperation, _IterTag1, _IterTag2)

```

- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`  
`_RAOIter std::__parallel::__transform1_switch (_RAIter, _RAIter,`  
`_RAOIter, _UnaryOperation, random_access_iterator_tag, random_`  
`access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism=\_\_gnu\_`  
`parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation`  
`>`  
`_RAIter3 std::__parallel::__transform2_switch (_RAIter1, _RAIter1, _`  
`RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_`  
`access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism`  
`__parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, type-`  
`name _Tag1, typename _Tag2, typename _Tag3 >`  
`_OIter std::__parallel::__transform2_switch (_Iter1, _Iter1, _Iter2, _OIter,`  
`_BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, type-`  
`name _IterTag2 >`  
`_OIter std::__parallel::__unique_copy_switch (_Iter, _Iter, _OIter, _`  
`Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`  
`_RandomAccess_OIter std::__parallel::__unique_copy_switch (_RAIter,`  
`_RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag,`  
`random_access_iterator_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::__parallel::__adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter >`  
`_Filter std::__parallel::adjacent_find (_Filter, _Filter, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::__parallel::adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::__parallel::adjacent_find (_Filter, _Filter)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __`  
`begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __`  
`begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __`  
`begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::\_\_Parallelism __`  
`parallelism_tag)`

- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __`  
`__begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::\_Parallelism __`  
`parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __`  
`begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __`  
`begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __`  
`begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, -`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _FIter, _FIter, _BiPredicate,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _FIter, _FIter, _BiPredicate)`
- `template<typename _Iter, typename _FIter >`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _FIter, _FIter)`
- `template<typename _Iter, typename _FIter >`  
`_Iter std::__parallel::find_first_of ( _Iter, _Iter, _FIter, _FIter, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each ( _Iter __begin, _Iter __end, _Function`  
`__f, \_\_gnu\_parallel::sequential\_tag)`



- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _-`  
`Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __-`  
`end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`

- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism \_\_parallelism\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`

- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter`  
`__end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _-`  
`RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate, \_\_gnu\_`  
`parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, \_\_`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &&__rand)`
- `template<typename _RAIter >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _-`  
`RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_`  
`gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`

- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, __-`  
`gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, __-`  
`gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, __-`  
`gnu_parallel::Parallelism)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _-`  
`BiPredicate)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _-`  
`BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu -`  
`parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu -`  
`parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _-`  
`BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _-`  
`BiPredicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter,`  
`_Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter,`  
`_Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`  
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`  
`OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`  
`OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`  
`OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _-`  
`OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`  
`Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`  
`Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`  
`Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _-`  
`Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`  
`Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _-`  
`Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_-`  
`gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __-`  
`comp)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`

- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`

### 6.7.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/algorithmfwd.h](#).

## 6.8 allocator.h File Reference

### Classes

- class [std::allocator<\\_Tp>](#)  
*The standard allocator, as per [20.4].*  
*Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html>.*
- class [std::allocator<void>](#)  
*[allocator<void>](#) specialization.*
- struct [std::allocator\\_arg\\_t](#)  
*[[allocator.tag](#)]*
- struct [std::uses\\_allocator<\\_Tp, \\_Alloc>](#)  
*[[allocator.uses.trait](#)]*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _T1, typename _T2 >  
bool std::operator!= (const allocator<_T1> &, const allocator<_T2> &)`
- `template<typename _Tp >  
bool std::operator!= (const allocator<_Tp> &, const allocator<_Tp> &)`
- `template<typename _T1, typename _T2 >  
bool std::operator== (const allocator<_T1> &, const allocator<_T2> &)`
- `template<typename _Tp >  
bool std::operator== (const allocator<_Tp> &, const allocator<_Tp> &)`

### Variables

- static const allocator\_arg\_t **std::allocator\_arg**

### 6.8.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [allocator.h](#).

## 6.9 array File Reference

### 6.9.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [array](#).

## 6.10 array File Reference

### Classes

- struct [std::tr1::array< \\_Tp, \\_Nm >](#)  
*A standard container for storing a fixed size sequence of elements.*

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
_Tp & std::tr1::get (array< _Tp, _Nm > &__arr)`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
const _Tp & std::tr1::get (const array< _Tp, _Nm > &__arr)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::tr1::operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>  
bool std::tr1::operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`



- `template<typename _Tp, std::size_t _Nm>`  
`bool std::tr1::operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::tr1::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::tr1::operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`bool std::tr1::operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`void std::tr1::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two)`

### 6.10.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/array](#).

## 6.11 array\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::array\\_allocator< \\_Tp, \\_Array >](#)  
*An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.*
- class [\\_\\_gnu\\_cxx::array\\_allocator\\_base< \\_Tp >](#)  
*Base class.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp, typename _Array >`  
`bool __gnu_cxx::operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp, typename _Array >`  
`bool __gnu_cxx::operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`

### 6.11.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [array\\_allocator.h](#).

## 6.12 assoc\_container.hpp File Reference

### Classes

- [class \\_\\_gnu\\_pbds::basic\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Resize\\_Policy, Store\\_Hash, Tag, Policy\\_TL, Allocator >](#)  
*An abstract basic hash-based associative container.*
- [class \\_\\_gnu\\_pbds::basic\\_tree< Key, Mapped, Tag, Node\\_Update, Policy\\_Tl, Allocator >](#)  
*An abstract basic tree-like (tree, trie) associative container.*
- [class \\_\\_gnu\\_pbds::cc\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Comb\\_Hash\\_Fn, Resize\\_Policy, Store\\_Hash, Allocator >](#)  
*A concrete collision-chaining hash-based associative container.*
- [class \\_\\_gnu\\_pbds::container\\_base< Key, Mapped, Tag, Policy\\_Tl, Allocator >](#)  
*An abstract basic associative container.*
- [class \\_\\_gnu\\_pbds::gp\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Comb\\_Probe\\_Fn, Probe\\_Fn, Resize\\_Policy, Store\\_Hash, Allocator >](#)  
*A concrete general-probing hash-based associative container.*
- [class \\_\\_gnu\\_pbds::list\\_update< Key, Mapped, Eq\\_Fn, Update\\_Policy, Allocator >](#)  
*A list-update based associative container.*

- class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, Allocator >`  
*A concrete basic tree-based associative container.*
- class `__gnu_pbds::trie< Key, Mapped, E_Access_Traits, Tag, Node_Update, Allocator >`  
*A concrete basic trie-based associative container.*

## Namespaces

- namespace `__gnu_pbds`

## Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_CLASS_NAME`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS_C_DEC`
- `#define PB_DS_TRIE_NODE_AND_ITS_TRAITS`

### 6.12.1 Detailed Description

Contains associative containers.

Definition in file [assoc\\_container.hpp](#).

## 6.13 atomic File Reference

### Classes

- struct `std::atomic< _Tp >`

*atomic* /// 29.4.3, Generic atomic type, primary class template.

- struct `std::atomic< _Tp * >`  
*Partial specialization for pointer types.*
- struct `std::atomic< bool >`  
*Explicit specialization for bool.*
- struct `std::atomic< char >`  
*Explicit specialization for char.*
- struct `std::atomic< char16_t >`  
*Explicit specialization for char16\_t.*
- struct `std::atomic< char32_t >`  
*Explicit specialization for char32\_t.*
- struct `std::atomic< int >`  
*Explicit specialization for int.*
- struct `std::atomic< long >`  
*Explicit specialization for long.*
- struct `std::atomic< long long >`  
*Explicit specialization for long long.*
- struct `std::atomic< short >`  
*Explicit specialization for short.*
- struct `std::atomic< signed char >`  
*Explicit specialization for signed char.*
- struct `std::atomic< unsigned char >`  
*Explicit specialization for unsigned char.*
- struct `std::atomic< unsigned int >`  
*Explicit specialization for unsigned int.*
- struct `std::atomic< unsigned long >`  
*Explicit specialization for unsigned long.*
- struct `std::atomic< unsigned long long >`

*Explicit specialization for unsigned long long.*

- struct `std::atomic< unsigned short >`  
*Explicit specialization for unsigned short.*
- struct `std::atomic< void * >`  
*Explicit specialization for void\*.*
- struct `std::atomic< wchar_t >`  
*Explicit specialization for wchar\_t.*

## Namespaces

- namespace `std`

## Functions

- memory\_order `std::__calculate_memory_order` (memory\_order \_\_m)
- template<typename \_ITp >  
bool `std::atomic_compare_exchange_strong` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool `std::atomic_compare_exchange_strong` (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- bool `std::atomic_compare_exchange_strong` (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- template<typename \_ITp >  
bool `std::atomic_compare_exchange_strong_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `std::atomic_compare_exchange_strong_explicit` (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- bool `std::atomic_compare_exchange_strong_explicit` (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- template<typename \_ITp >  
bool `std::atomic_compare_exchange_weak` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2)
- bool `std::atomic_compare_exchange_weak` (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2)
- bool `std::atomic_compare_exchange_weak` (atomic\_bool \*\_\_a, bool \*\_\_i1, bool \_\_i2)
- template<typename \_ITp >  
bool `std::atomic_compare_exchange_weak_explicit` (\_\_atomic\_base< \_ITp > \*\_\_a, \_ITp \*\_\_i1, \_ITp \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)

- **bool std::atomic\_compare\_exchange\_weak\_explicit** (atomic\_bool \*\_\_a, bool \_\_i1, bool \_\_i2, memory\_order \_\_m1, memory\_order \_\_m2)
- **bool std::atomic\_compare\_exchange\_weak\_explicit** (atomic\_address \*\_\_a, void \*\*\_\_v1, void \*\_\_v2, memory\_order \_\_m1, memory\_order \_\_m2)
- **void \* std::atomic\_exchange** (atomic\_address \*\_\_a, void \*\_\_v)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_exchange** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i)
- **bool std::atomic\_exchange** (atomic\_bool \*\_\_a, bool \_\_i)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_exchange\_explicit** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- **void \* std::atomic\_exchange\_explicit** (atomic\_address \*\_\_a, void \*\_\_v, memory\_order \_\_m)
- **bool std::atomic\_exchange\_explicit** (atomic\_bool \*\_\_a, bool \_\_i, memory\_order \_\_m)
- **void \* std::atomic\_fetch\_add** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_add** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i)
- **void \* std::atomic\_fetch\_add\_explicit** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d, memory\_order \_\_m)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_add\_explicit** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_and** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_and\_explicit** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_or** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_or\_explicit** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- **void \* std::atomic\_fetch\_sub** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_sub** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i)
- **void \* std::atomic\_fetch\_sub\_explicit** (atomic\_address \*\_\_a, ptrdiff\_t \_\_d, memory\_order \_\_m)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_sub\_explicit** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i, memory\_order \_\_m)
- **template<typename \_ITp >**  
**\_ITp std::atomic\_fetch\_xor** (\_\_atomic\_base<\_ITp> \*\_\_a, \_ITp \_\_i)

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit` (`__atomic_base<_ITp> *__a`, `_ITp __i`,  
`memory_order __m`)
- `void std::atomic_flag_clear_explicit` (`atomic_flag *__a`, `memory_order __m`)
- `bool std::atomic_flag_test_and_set_explicit` (`atomic_flag *__a`, `memory_order __m`)
- `bool std::atomic_is_lock_free` (`const atomic_bool *__a`)
- `bool std::atomic_is_lock_free` (`const atomic_address *__a`)
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free` (`const __atomic_base<_ITp> *__a`)
- `void * std::atomic_load` (`const atomic_address *__a`)
- `template<typename _ITp >`  
`_ITp std::atomic_load` (`const __atomic_base<_ITp> *__a`)
- `bool std::atomic_load` (`const atomic_bool *__a`)
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit` (`const __atomic_base<_ITp> *__a`,  
`memory_order __m`)
- `void * std::atomic_load_explicit` (`const atomic_address *__a`, `memory_order __m`)
- `bool std::atomic_load_explicit` (`const atomic_bool *__a`, `memory_order __m`)
- `void std::atomic_store` (`atomic_bool *__a`, `bool __i`)
- `template<typename _ITp >`  
`void std::atomic_store` (`__atomic_base<_ITp> *__a`, `_ITp __i`)
- `void std::atomic_store` (`atomic_address *__a`, `void *__v`)
- `void std::atomic_store_explicit` (`atomic_bool *__a`, `bool __i`, `memory_order __m`)
- `void std::atomic_store_explicit` (`atomic_address *__a`, `void *__v`, `memory_order __m`)
- `template<typename _ITp >`  
`void std::atomic_store_explicit` (`__atomic_base<_ITp> *__a`, `_ITp __i`,  
`memory_order __m`)
- `template<typename _Tp >`  
`_Tp std::kill_dependency` (`_Tp __y`)

### 6.13.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic](#).

## 6.14 `atomic_0.h` File Reference

### Classes

- struct [\\_\\_atomic0::atomic\\_address](#)  
*29.4.2, address types*
- struct [\\_\\_atomic0::atomic\\_bool](#)  
*atomic\_bool*
- struct [\\_\\_atomic0::atomic\\_flag](#)  
*atomic\_flag*

### Defines

- `#define _ATOMIC_CMPEXCHNG_(__a, __e, __m, __x)`
- `#define _ATOMIC_LOAD_(__a, __x)`
- `#define _ATOMIC_MODIFY_(__a, __o, __m, __x)`
- `#define _ATOMIC_STORE_(__a, __m, __x)`

### 6.14.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomic\\_0.h](#).

## 6.15 `atomic_2.h` File Reference

### Classes

- struct [\\_\\_atomic2::atomic\\_address](#)  
*29.4.2, address types*
- struct [\\_\\_atomic2::atomic\\_bool](#)  
*atomic\_bool*
- struct [\\_\\_atomic2::atomic\\_flag](#)  
*atomic\_flag*



### 6.15.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomic\\_2.h](#).

## 6.16 atomic\_base.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _ATOMIC_CMPEXCHNG(__a, __e, __m, __x)`
- `#define _ATOMIC_LOAD(__a, __x)`
- `#define _ATOMIC_MODIFY(__a, __o, __m, __x)`
- `#define _ATOMIC_STORE(__a, __m, __x)`
- `#define _GLIBCXX_ATOMIC_BASE_H`
- `#define _GLIBCXX_ATOMIC_NAMESPACE`
- `#define _GLIBCXX_ATOMIC_PROPERTY`
- `#define ATOMIC_ADDRESS_LOCK_FREE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_INTEGRAL_LOCK_FREE`

### Typedefs

- `typedef struct std::__atomic_flag_base std::__atomic_flag_base`
- `typedef atomic\_short std::atomic_int_fast16_t`
- `typedef atomic\_int std::atomic_int_fast32_t`
- `typedef atomic\_llong std::atomic_int_fast64_t`
- `typedef atomic\_schar std::atomic_int_fast8_t`
- `typedef atomic\_short std::atomic_int_least16_t`
- `typedef atomic\_int std::atomic_int_least32_t`
- `typedef atomic\_llong std::atomic_int_least64_t`
- `typedef atomic\_schar std::atomic_int_least8_t`
- `typedef atomic\_llong std::atomic_intmax_t`
- `typedef atomic\_long std::atomic_intptr_t`
- `typedef atomic\_long std::atomic_ptrdiff_t`
- `typedef atomic\_ulong std::atomic_size_t`

- typedef [atomic\\_long](#) `std::atomic_ssize_t`
- typedef [atomic\\_ushort](#) `std::atomic_uint_fast16_t`
- typedef [atomic\\_uint](#) `std::atomic_uint_fast32_t`
- typedef [atomic\\_ullong](#) `std::atomic_uint_fast64_t`
- typedef [atomic\\_uchar](#) `std::atomic_uint_fast8_t`
- typedef [atomic\\_ushort](#) `std::atomic_uint_least16_t`
- typedef [atomic\\_uint](#) `std::atomic_uint_least32_t`
- typedef [atomic\\_ullong](#) `std::atomic_uint_least64_t`
- typedef [atomic\\_uchar](#) `std::atomic_uint_least8_t`
- typedef [atomic\\_ullong](#) `std::atomic_uintmax_t`
- typedef [atomic\\_ulong](#) `std::atomic_uintptr_t`
- typedef enum [std::memory\\_order](#) `std::memory_order`

## Enumerations

- enum [std::memory\\_order](#) {  
     [memory\\_order\\_relaxed](#),   [memory\\_order\\_consume](#),   [memory\\_order\\_-  
     acquire](#), [memory\\_order\\_release](#),  
     [memory\\_order\\_acq\\_rel](#), [memory\\_order\\_seq\\_cst](#) }

## Functions

- void [std::\\_\\_atomic\\_flag\\_wait\\_explicit](#) ([\\_\\_atomic\\_flag\\_base](#) \*, [memory\\_order](#))  
     [\\_GLIBCXX\\_NOTHROW](#)
- [std::\\_\\_attribute\\_\\_](#) (([\\_\\_const\\_\\_](#))) [\\_\\_atomic\\_flag\\_base](#) \*[\\_\\_atomic\\_flag\\_for\\_-  
     address](#)([const void](#) \*[\\_\\_z](#)) [\\_GLIBCXX\\_NOTHROW](#)
- void [std::atomic\\_flag\\_clear](#) ([\\_\\_atomic\\_flag\\_base](#) \*[\\_\\_a](#))
- void [std::atomic\\_flag\\_clear\\_explicit](#) ([\\_\\_atomic\\_flag\\_base](#) \*, [memory\\_order](#))  
     [\\_GLIBCXX\\_NOTHROW](#)
- bool [std::atomic\\_flag\\_test\\_and\\_set](#) ([\\_\\_atomic\\_flag\\_base](#) \*[\\_\\_a](#))
- bool    [std::atomic\\_flag\\_test\\_and\\_set\\_explicit](#)    ([\\_\\_atomic\\_flag\\_base](#)    \*,  
     [memory\\_order](#)) [\\_GLIBCXX\\_NOTHROW](#)

### 6.16.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [atomic\\_base.h](#).

## 6.17 atomic\_word.h File Reference

### Typedefs

- typedef int **\_Atomic\_word**

### 6.17.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [atomic\\_word.h](#).

## 6.18 atomicfwd\_c.h File Reference

### Defines

- #define **\_ATOMIC\_MEMBER\_**
- #define **atomic\_compare\_exchange**(\_\_a, \_\_e, \_\_m)
- #define **atomic\_compare\_exchange\_explicit**(\_\_a, \_\_e, \_\_m, \_\_x, \_\_y)
- #define **atomic\_exchange**(\_\_a, \_\_m)
- #define **atomic\_exchange\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_add**(\_\_a, \_\_m)
- #define **atomic\_fetch\_add\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_and**(\_\_a, \_\_m)
- #define **atomic\_fetch\_and\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_or**(\_\_a, \_\_m)
- #define **atomic\_fetch\_or\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_sub**(\_\_a, \_\_m)
- #define **atomic\_fetch\_sub\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_fetch\_xor**(\_\_a, \_\_m)
- #define **atomic\_fetch\_xor\_explicit**(\_\_a, \_\_m, \_\_x)
- #define **atomic\_is\_lock\_free**(\_\_a)
- #define **atomic\_load**(\_\_a)
- #define **atomic\_load\_explicit**(\_\_a, \_\_x)
- #define **atomic\_store**(\_\_a, \_\_m)
- #define **atomic\_store\_explicit**(\_\_a, \_\_m, \_\_x)

## Typedefs

- typedef struct \_\_atomic\_address\_base **atomic\_address**
- typedef struct \_\_atomic\_bool\_base **atomic\_bool**
- typedef struct \_\_atomic\_char\_base **atomic\_char**
- typedef struct \_\_atomic\_short\_base **atomic\_char16\_t**
- typedef struct \_\_atomic\_int\_base **atomic\_char32\_t**
- typedef struct \_\_atomic\_flag\_base **atomic\_flag**
- typedef struct \_\_atomic\_int\_base **atomic\_int**
- typedef struct \_\_atomic\_llong\_base **atomic\_llong**
- typedef struct \_\_atomic\_long\_base **atomic\_long**
- typedef struct \_\_atomic\_schar\_base **atomic\_schar**
- typedef struct \_\_atomic\_short\_base **atomic\_short**
- typedef struct \_\_atomic\_uchar\_base **atomic\_uchar**
- typedef struct \_\_atomic\_uint\_base **atomic\_uint**
- typedef struct \_\_atomic\_ullong\_base **atomic\_ullong**
- typedef struct \_\_atomic\_ulong\_base **atomic\_ulong**
- typedef struct \_\_atomic\_ushort\_base **atomic\_ushort**
- typedef struct \_\_atomic\_wchar\_t\_base **atomic\_wchar\_t**

### 6.18.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicfwd\\_c.h](#).

## 6.19 atomicfwd\_cxx.h File Reference

### Defines

- #define **\_ATOMIC\_MEMBER\_**

### Typedefs

- typedef \_\_atomic\_base< char > [atomic\\_char](#)
- typedef \_\_atomic\_base< char16\_t > [atomic\\_char16\\_t](#)
- typedef \_\_atomic\_base< char32\_t > [atomic\\_char32\\_t](#)
- typedef \_\_atomic\_base< int > [atomic\\_int](#)
- typedef \_\_atomic\_base< long long > [atomic\\_llong](#)
- typedef \_\_atomic\_base< long > [atomic\\_long](#)

- typedef \_\_atomic\_base< signed char > [atomic\\_schar](#)
- typedef \_\_atomic\_base< short > [atomic\\_short](#)
- typedef \_\_atomic\_base< unsigned char > [atomic\\_uchar](#)
- typedef \_\_atomic\_base< unsigned int > [atomic\\_uint](#)
- typedef \_\_atomic\_base< unsigned long long > [atomic\\_ullong](#)
- typedef \_\_atomic\_base< unsigned long > [atomic\\_ulong](#)
- typedef \_\_atomic\_base< unsigned short > [atomic\\_ushort](#)
- typedef \_\_atomic\_base< wchar\_t > [atomic\\_wchar\\_t](#)

### 6.19.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicfwd\\_cxx.h](#).

## 6.20 atomicity.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- #define [\\_GLIBCXX\\_READ\\_MEM\\_BARRIER](#)
- #define [\\_GLIBCXX\\_WRITE\\_MEM\\_BARRIER](#)

### Functions

- static void [\\_\\_gnu\\_cxx::\\_\\_atomic\\_add](#) (volatile \_Atomic\_word \*\_\_mem, int \_\_val)
- static void [\\_\\_gnu\\_cxx::\\_\\_atomic\\_add\\_single](#) (\_Atomic\_word \*\_\_mem, int \_\_val)
- static \_Atomic\_word [\\_\\_gnu\\_cxx::\\_\\_attribute\\_\\_\(\(\\_\\_unused\\_\\_\)\) \\_\\_exchange\\_and\\_add\\_dispatch](#) (\_Atomic\_word \*\_\_mem)
- static \_Atomic\_word [\\_\\_gnu\\_cxx::\\_\\_exchange\\_and\\_add](#) (volatile \_Atomic\_word \*\_\_mem, int \_\_val)
- else return [\\_\\_gnu\\_cxx::\\_\\_exchange\\_and\\_add\\_single](#) (\_\_mem, \_\_val)
- static \_Atomic\_word [\\_\\_gnu\\_cxx::\\_\\_exchange\\_and\\_add\\_single](#) (\_Atomic\_word \*\_\_mem, int \_\_val)
- static \_Atomic\_word int \_\_val [\\_\\_gnu\\_cxx::if](#) (\_\_gthread\_active\_p()) return \_\_exchange\_and\_add(\_\_mem

## Variables

- static `_Atomic_word` `int __val` `__gnu_cxx::__val`

### 6.20.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [atomicity.h](#).

## 6.21 `auto_ptr.h` File Reference

### Classes

- class [std::auto\\_ptr<\\_Tp>](#)  
*A simple smart pointer providing strict ownership semantics.*
- struct [std::auto\\_ptr\\_ref<\\_Tp1>](#)

### Namespaces

- namespace [std](#)

### 6.21.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [auto\\_ptr.h](#).

## 6.22 `balanced_quicksort.h` File Reference

Implementation of a dynamically load-balanced parallel quicksort.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter>](#)  
*Information local to one thread in the parallel quicksort run.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort\_qsb (_RAIter __begin, _RAIter __end, _-`  
`Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_qsb\_conquer (_QSBThreadLocal< _RAIter > **__tls,`  
`_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _-`  
`ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type \_\_gnu\_parallel::\_\_qsb\_divide`  
`(_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_`  
`threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_qsb\_local\_sort\_with\_helping (_QSBThreadLocal< _-`  
`RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)`

### 6.22.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort. It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [balanced\\_quicksort.h](#).

## 6.23 base.h File Reference

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

## Classes

- class [\\_\\_gnu\\_parallel::\\_\\_binder1st](#)< \_Operation, \_FirstArgumentType, \_-  
[SecondArgumentType, \\_ResultType >](#)

Similar to [`std::binder1st`](#), but giving the argument types explicitly.

- class [`\_\_gnu\_parallel::\_\_binder2nd`](#)< [`\_Operation`](#), [`\_FirstArgumentType`](#), [`\_SecondArgumentType`](#), [`\_ResultType`](#) >

Similar to [`std::binder2nd`](#), but giving the argument types explicitly.

- class [`\_\_gnu\_parallel::\_\_unary\_negate`](#)< [`\_Predicate`](#), [`argument\_type`](#) >

Similar to [`std::unary\_negate`](#), but giving the argument types explicitly.

- class [`\_\_gnu\_parallel::\_EqualFromLess`](#)< [`\_T1`](#), [`\_T2`](#), [`\_Compare`](#) >

Constructs predicate for equality from strict weak ordering predicate.

- struct [`\_\_gnu\_parallel::\_EqualTo`](#)< [`\_T1`](#), [`\_T2`](#) >

Similar to [`std::equal\_to`](#), but allows two different types.

- struct [`\_\_gnu\_parallel::\_Less`](#)< [`\_T1`](#), [`\_T2`](#) >

Similar to [`std::less`](#), but allows two different types.

- struct [`\_\_gnu\_parallel::\_Multiplies`](#)< [`\_Tp1`](#), [`\_Tp2`](#), [`\_Result`](#) >

Similar to [`std::multiplies`](#), but allows two different types.

- struct [`\_\_gnu\_parallel::\_Plus`](#)< [`\_Tp1`](#), [`\_Tp2`](#), [`\_Result`](#) >

Similar to [`std::plus`](#), but allows two different types.

- class [`\_\_gnu\_parallel::\_PseudoSequence`](#)< [`\_Tp`](#), [`\_DifferenceTp`](#) >

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

- class [`\_\_gnu\_parallel::\_PseudoSequenceIterator`](#)< [`\_Tp`](#), [`\_DifferenceTp`](#) >

[`\_Iterator`](#) associated with [`\_\_gnu\_parallel::\_PseudoSequence`](#). If features the usual random-access iterator functionality.

## Namespaces

- namespace [`\_\_gnu\_parallel`](#)
- namespace [`\_\_gnu\_sequential`](#)
- namespace [`std`](#)
- namespace [`std::\_\_parallel`](#)

## Defines

- [`#define \_GLIBCXX\_PARALLEL\_ASSERT\(\_Condition\)`](#)



## Functions

- void [\\_\\_gnu\\_parallel::\\_\\_decode2](#) (\_CASable \_\_x, int &\_\_a, int &\_\_b)
- \_CASable [\\_\\_gnu\\_parallel::\\_\\_encode2](#) (int \_\_a, int \_\_b)
- \_ThreadIndex [\\_\\_gnu\\_parallel::\\_\\_get\\_max\\_threads](#) ()
- bool [\\_\\_gnu\\_parallel::\\_\\_is\\_parallel](#) (const \_Parallelism \_\_p)
- template<typename \_RAIter, typename \_Compare >  
\_RAIter [\\_\\_gnu\\_parallel::\\_\\_median\\_of\\_three\\_iterators](#) (\_RAIter \_\_a, \_RAIter \_\_b, \_RAIter \_\_c, \_Compare \_\_comp)
- template<typename \_Size >  
\_Size [\\_\\_gnu\\_parallel::\\_\\_rd\\_log2](#) (\_Size \_\_n)
- template<typename \_Tp >  
const \_Tp & [\\_\\_gnu\\_parallel::max](#) (const \_Tp &\_\_a, const \_Tp &\_\_b)
- template<typename \_Tp >  
const \_Tp & [\\_\\_gnu\\_parallel::min](#) (const \_Tp &\_\_a, const \_Tp &\_\_b)

### 6.23.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/base.h](#).

## 6.24 base.h File Reference

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)
- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### 6.24.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/base.h](#).

## 6.25 `basic_file.h` File Reference

### Namespaces

- namespace [std](#)

### 6.25.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_file.h](#).

## 6.26 `basic_ios.h` File Reference

### Classes

- class [std::basic\\_ios<\\_CharT, \\_Traits>](#)

*Virtual base class for all stream classes.*

*Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Facet>`  
`const _Facet & std::__check_facet (const _Facet *__f)`

### 6.26.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_ios.h](#).

## 6.27 `basic_ios.tcc` File Reference

### Namespaces

- namespace `std`

### 6.27.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `basic_ios.tcc`.

## 6.28 `basic_iterator.h` File Reference

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

### 6.28.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file `basic_iterator.h`.

## 6.29 `basic_string.h` File Reference

### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`  
*Managing sequences of characters and character-like objects.*
- struct `std::hash<string>`  
*`std::hash` specialization for `string`.*
- struct `std::hash<u16string>`  
*`std::hash` specialization for `u16string`.*
- struct `std::hash<u32string>`  
*`std::hash` specialization for `u32string`.*

- struct [std::hash<wstring>](#)  
[std::hash](#) specialization for *wstring*.

## Namespaces

- namespace [std](#)

## Functions

- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_istream< \\_CharT, \\_Traits > & std::getline](#) ([basic\\_istream< \\_CharT, \\_Traits > &\\_\\_is](#), [basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_str](#), [\\_CharT \\_\\_delim](#))
- template<>  
[basic\\_istream< wchar\\_t > & std::getline](#) ([basic\\_istream< wchar\\_t > &\\_\\_in](#), [basic\\_string< wchar\\_t > &\\_\\_str](#), [wchar\\_t \\_\\_delim](#))
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_istream< \\_CharT, \\_Traits > & std::getline](#) ([basic\\_istream< \\_CharT, \\_Traits > &\\_\\_is](#), [basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_str](#))
- template<>  
[basic\\_istream< char > & std::getline](#) ([basic\\_istream< char > &\\_\\_in](#), [basic\\_string< char > &\\_\\_str](#), [char \\_\\_delim](#))
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[bool std::operator!=](#) ([const \\_CharT \\*\\_\\_lhs](#), [const basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_rhs](#))
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[bool std::operator!=](#) ([const basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_lhs](#), [const basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_rhs](#))
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[bool std::operator!=](#) ([const basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_lhs](#), [const \\_CharT \\*\\_\\_rhs](#))
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string< \\_CharT, \\_Traits, \\_Alloc > std::operator+](#) ([const basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_lhs](#), [const \\_CharT \\*\\_\\_rhs](#))
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string< \\_CharT, \\_Traits, \\_Alloc > std::operator+](#) ([const basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_lhs](#), [const basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_rhs](#))
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[basic\\_string< \\_CharT, \\_Traits, \\_Alloc > std::operator+](#) ([const basic\\_string< \\_CharT, \\_Traits, \\_Alloc > &\\_\\_lhs](#), [\\_CharT \\_\\_rhs](#))

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const`  
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _`  
`CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const _CharT *__lhs, const basic_string< _CharT, _`  
`Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type`  
`std::operator== (const basic_string< _CharT > &__lhs, const basic_string<`  
`_CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const _CharT *__lhs, const basic_string< _CharT, _-`  
`_Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`  
`_Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > &__is,`  
`basic_string< char > &__str)`
- `double std::stod (const string &__str, size_t *__idx=0)`
- `float std::stof (const string &__str, size_t *__idx=0)`
- `int std::stoi (const string &__str, size_t *__idx=0, int __base=10)`
- `long std::stol (const string &__str, size_t *__idx=0, int __base=10)`
- `long double std::stold (const string &__str, size_t *__idx=0)`
- `long long std::stoll (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long std::stoul (const string &__str, size_t *__idx=0, int __base=10)`
- `unsigned long long std::stoull (const string &__str, size_t *__idx=0, int __-`  
`base=10)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string<`  
`_CharT, _Traits, _Alloc > &__rhs)`
- `string std::to\_string (long double __val)`
- `string std::to\_string (int __val)`
- `string std::to\_string (long __val)`
- `string std::to\_string (unsigned __val)`
- `string std::to\_string (float __val)`
- `string std::to\_string (unsigned long long __val)`
- `string std::to\_string (unsigned long __val)`
- `string std::to\_string (double __val)`
- `string std::to\_string (long long __val)`
- `wstring std::to\_wstring (unsigned long __val)`
- `wstring std::to\_wstring (long __val)`

- `wstring std::to_wstring` (int \_\_val)
- `wstring std::to_wstring` (unsigned \_\_val)
- `wstring std::to_wstring` (unsigned long long \_\_val)
- `wstring std::to_wstring` (long long \_\_val)
- `wstring std::to_wstring` (double \_\_val)
- `wstring std::to_wstring` (long double \_\_val)
- `wstring std::to_wstring` (float \_\_val)

### 6.29.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_string.h](#).

## 6.30 `basic_string.tcc` File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::getline` (`basic_istream< _CharT, _Traits > &__is`, `basic_string< _CharT, _Traits, _Alloc > &__str`, `_CharT __delim`)
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+` (`_CharT __lhs`, `const basic_string< _CharT, _Traits, _Alloc > &__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+` (`const _CharT *__lhs`, `const basic_string< _CharT, _Traits, _Alloc > &__rhs`)
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>>` (`basic_istream< _CharT, _Traits > &__is`, `basic_string< _CharT, _Traits, _Alloc > &__str`)

### 6.30.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [basic\\_string.tcc](#).

## 6.31 `basic_types.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::value\\_type\\_base< Key, Mapped, Allocator, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::value\\_type\\_base< Key, Mapped, Allocator, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::value\\_type\\_base< Key, null\\_mapped\\_type, Allocator, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::value\\_type\\_base< Key, null\\_mapped\\_type, Allocator, true >](#)

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_C_DEC`
- #define `PB_DS_CLASS_T_DEC`
- #define `PB_DS_CLASS_T_DEC`

#### 6.31.1 Detailed Description

Contains basic types used by containers.

Definition in file [basic\\_types.hpp](#).

## 6.32 `binders.h` File Reference

### Classes

- class [std::binder1st< \\_Operation >](#)  
*One of the [binder functors](#).*
- class [std::binder2nd< \\_Operation >](#)  
*One of the [binder functors](#).*



## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Operation , typename _Tp >  
binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &_  
_x)`
- `template<typename _Operation , typename _Tp >  
binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &_  
_x)`

## Variables

- [std::binder1st](#) [std::\\_GLIBCXX\\_DEPRECATED\\_ATTR](#)

### 6.32.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [binders.h](#).

## 6.33 bitmap\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector< \\_Tp >](#)  
*[\\_\\_mini\\_vector<>](#) is a stripped down version of the full-fledged [std::vector<>](#).*
- class [\\_\\_gnu\\_cxx::\\_\\_detail::Bitmap\\_counter< \\_Tp >](#)  
*The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.*
- class [\\_\\_gnu\\_cxx::\\_\\_detail::Ffit\\_finder< \\_Tp >](#)  
*The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.*
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator< \\_Tp >](#)  
*Bitmap Allocator, primary template.*

- class [\\_\\_gnu\\_cxx::free\\_list](#)

*The free list class for managing chunks of memory to be given to and returned by the [bitmap\\_allocator](#).*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::\\_\\_detail](#)

## Defines

- #define [\\_BALLOC\\_ALIGN\\_BYTES](#)

## Enumerations

- enum { **bits\_per\_byte**, **bits\_per\_block** }

## Functions

- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_allocate](#) (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- void [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_bit\\_free](#) (size\_t \*\_\_pbmap, size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
\_ForwardIterator [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_lower\\_bound](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair >  
size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_bitmaps](#) (\_AddrPair \_\_ap)
- template<typename \_AddrPair >  
size\_t [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_num\\_blocks](#) (\_AddrPair \_\_ap)
- size\_t [\\_\\_gnu\\_cxx::\\_\\_Bit\\_scan\\_forward](#) (size\_t \_\_num)
- template<typename \_Tp1, typename \_Tp2 >  
bool [\\_\\_gnu\\_cxx::operator!=](#) (const bitmap\_allocator< \_Tp1 > &, const bitmap\_allocator< \_Tp2 > &) throw ()
- template<typename \_Tp1, typename \_Tp2 >  
bool [\\_\\_gnu\\_cxx::operator==](#) (const bitmap\_allocator< \_Tp1 > &, const bitmap\_allocator< \_Tp2 > &) throw ()

### 6.33.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [bitmap\\_allocator.h](#).

### 6.33.2 Define Documentation

#### 6.33.2.1 #define \_BALLOC\_ALIGN\_BYTES

The constant in the expression below is the alignment required in bytes.

Definition at line 44 of file [bitmap\\_allocator.h](#).

## 6.34 bitset File Reference

### Classes

- struct [std::\\_Base\\_bitset< \\_Nw >](#)
- struct [std::\\_Base\\_bitset< 0 >](#)
- struct [std::\\_Base\\_bitset< 1 >](#)
- class [std::bitset< \\_Nb >](#)

*The bitset class represents a fixed-size sequence of bits.*

- class [std::bitset< \\_Nb >::reference](#)
- struct [std::hash<::bitset< \\_Nb > >](#)

*[std::hash](#) specialization for bitset.*

### Namespaces

- namespace [std](#)

### Defines

- #define [\\_GLIBCXX\\_BITSET\\_BITS\\_PER\\_WORD](#)
- #define [\\_GLIBCXX\\_BITSET\\_WORDS\(\\_\\_n\)](#)

## Functions

- `template<size_t _Nb>`  
`bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_ostream< _CharT, _Traits > & std::operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`

### 6.34.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [bitset](#).

## 6.35 bitset File Reference

### Classes

- class [std::\\_\\_debug::bitset< \\_Nb >](#)  
*Class [std::bitset](#) with additional safety/checking/debug instrumentation.*
- struct [std::hash< \\_\\_debug::bitset< \\_Nb > >](#)  
*[std::hash](#) specialization for [bitset](#).*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<size_t _Nb>`  
`bitset< _Nb > std::__debug::operator& (const bitset< _Nb > &__x, const`  
`bitset< _Nb > &__y)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_ostream< _CharT, _Traits > & std::__debug::operator<<`  
`(std::basic\_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic\_istream< _CharT, _Traits > & std::__debug::operator>>`  
`(std::basic\_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::__debug::operator^ (const bitset< _Nb > &__x, const`  
`bitset< _Nb > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::__debug::operator| (const bitset< _Nb > &__x, const`  
`bitset< _Nb > &__y)`

### 6.35.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/bitset](#).

## 6.36 bitset File Reference

### Classes

- class [std::\\_\\_profile::bitset< \\_Nb >](#)  
*Class [std::bitset](#) wrapper with performance instrumentation.*
- struct [std::hash< \\_\\_profile::bitset< \\_Nb > >](#)  
*[std::hash](#) specialization for [bitset](#).*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator& (const bitset< _Nb > &__x, const`  
`bitset< _Nb > &__y)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & std::__profile::operator<<`  
`(std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & std::__profile::operator>>`  
`(std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator^ (const bitset< _Nb > &__x, const`  
`bitset< _Nb > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > std::__profile::operator| (const bitset< _Nb > &__x, const`  
`bitset< _Nb > &__y)`

### 6.36.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/bitset](#).

## 6.37 boost\_concept\_check.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- `#define _GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _-`  
`concept)`
- `#define _GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_-`  
`var3, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_-`  
`var3, _type_var4, _ns, _concept)`
- `#define _GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_`  
`OP, _NAME)`
- `#define _GLIBCXX_DEFINE_BINARY_PREDICATE_OP_-`  
`CONSTRAINT(_OP, _NAME)`

- `#define _IsUnused`

## Functions

- `template<class _Tp >`  
`void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept >`  
`void __gnu_cxx::__function_requires ()`

### 6.37.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [boost\\_concept\\_check.h](#).

## 6.38 boost\_sp\_counted\_base.h File Reference

### Classes

- class [std::tr1::bad\\_weak\\_ptr](#)  
*Exception possibly thrown by [shared\\_ptr](#).*

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- `void std::tr1::__throw_bad_weak_ptr ()`

### 6.38.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [boost\\_sp\\_counted\\_base.h](#).

## 6.39 `c++0x_warning.h` File Reference

### 6.39.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [c++0x\\_warning.h](#).

## 6.40 `c++allocator.h` File Reference

### Defines

- `#define __glibcxx_base_allocator`

### 6.40.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++allocator.h](#).

## 6.41 `c++config.h` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define __GLIBCXX__`
- `#define __glibcxx_assert(_Condition)`
- `#define __N(msgid)`
- `#define _GLIBCXX_ATOMIC_BUILTINS_1`
- `#define _GLIBCXX_ATOMIC_BUILTINS_2`
- `#define _GLIBCXX_ATOMIC_BUILTINS_4`
- `#define _GLIBCXX_ATOMIC_BUILTINS_8`
- `#define _GLIBCXX_BEGIN_EXTERN_C`
- `#define _GLIBCXX_BEGIN_LDBL_NAMESPACE`
- `#define _GLIBCXX_BEGIN_NAMESPACE(X)`
- `#define _GLIBCXX_BEGIN_NESTED_NAMESPACE(X, Y)`



- `#define _GLIBCXX_CONST`
- `#define _GLIBCXX_DEPRECATED_ATTR`
- `#define _GLIBCXX_END_EXTERN_C`
- `#define _GLIBCXX_END_LDBL_NAMESPACE`
- `#define _GLIBCXX_END_NAMESPACE`
- `#define _GLIBCXX_END_NESTED_NAMESPACE`
- `#define _GLIBCXX_EXTERN_TEMPLATE`
- `#define _GLIBCXX_FAST_MATH`
- `#define _GLIBCXX_HAS_GTHREADS`
- `#define _GLIBCXX_HAVE_ACOSF`
- `#define _GLIBCXX_HAVE_ACOSL`
- `#define _GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE`
- `#define _GLIBCXX_HAVE_ASINF`
- `#define _GLIBCXX_HAVE_ASINL`
- `#define _GLIBCXX_HAVE_ATAN2F`
- `#define _GLIBCXX_HAVE_ATAN2L`
- `#define _GLIBCXX_HAVE_ATANF`
- `#define _GLIBCXX_HAVE_ATANL`
- `#define _GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY`
- `#define _GLIBCXX_HAVE_CEILF`
- `#define _GLIBCXX_HAVE_CEILL`
- `#define _GLIBCXX_HAVE_COMPLEX_H`
- `#define _GLIBCXX_HAVE_COSF`
- `#define _GLIBCXX_HAVE_COSHF`
- `#define _GLIBCXX_HAVE_COSHL`
- `#define _GLIBCXX_HAVE_COSL`
- `#define _GLIBCXX_HAVE_DLFCN_H`
- `#define _GLIBCXX_HAVE_EBADMSG`
- `#define _GLIBCXX_HAVE_ECANCELED`
- `#define _GLIBCXX_HAVE_EIDRM`
- `#define _GLIBCXX_HAVE_ENDIAN_H`
- `#define _GLIBCXX_HAVE_ENODATA`
- `#define _GLIBCXX_HAVE_ENOLINK`
- `#define _GLIBCXX_HAVE_ENOSR`
- `#define _GLIBCXX_HAVE_ENOSTR`
- `#define _GLIBCXX_HAVE_ENOTRECOVERABLE`
- `#define _GLIBCXX_HAVE_ENOTSUP`
- `#define _GLIBCXX_HAVE_EOVERFLOW`
- `#define _GLIBCXX_HAVE_EOWNERDEAD`
- `#define _GLIBCXX_HAVE_EPROTO`
- `#define _GLIBCXX_HAVE_ETIME`
- `#define _GLIBCXX_HAVE_ETXTBSY`
- `#define _GLIBCXX_HAVE_EXECINFO_H`

- **#define \_GLIBCXX\_HAVE\_EXPF**
- **#define \_GLIBCXX\_HAVE\_EXPL**
- **#define \_GLIBCXX\_HAVE\_FABSF**
- **#define \_GLIBCXX\_HAVE\_FABSL**
- **#define \_GLIBCXX\_HAVE\_FENV\_H**
- **#define \_GLIBCXX\_HAVE\_FINITE**
- **#define \_GLIBCXX\_HAVE\_FINITEF**
- **#define \_GLIBCXX\_HAVE\_FINITEL**
- **#define \_GLIBCXX\_HAVE\_FLOAT\_H**
- **#define \_GLIBCXX\_HAVE\_FLOORF**
- **#define \_GLIBCXX\_HAVE\_FLOORL**
- **#define \_GLIBCXX\_HAVE\_FMODF**
- **#define \_GLIBCXX\_HAVE\_FMODL**
- **#define \_GLIBCXX\_HAVE\_FREXPF**
- **#define \_GLIBCXX\_HAVE\_FREXPL**
- **#define \_GLIBCXX\_HAVE\_GETIPINFO**
- **#define \_GLIBCXX\_HAVE\_GTHR\_DEFAULT**
- **#define \_GLIBCXX\_HAVE\_HYPOT**
- **#define \_GLIBCXX\_HAVE\_HYPOTF**
- **#define \_GLIBCXX\_HAVE\_HYPOTL**
- **#define \_GLIBCXX\_HAVE\_ICONV**
- **#define \_GLIBCXX\_HAVE\_INT64\_T**
- **#define \_GLIBCXX\_HAVE\_INT64\_T\_LONG**
- **#define \_GLIBCXX\_HAVE\_INTPTR\_T**
- **#define \_GLIBCXX\_HAVE\_ISINF**
- **#define \_GLIBCXX\_HAVE\_ISINFF**
- **#define \_GLIBCXX\_HAVE\_ISINFL**
- **#define \_GLIBCXX\_HAVE\_ISNAN**
- **#define \_GLIBCXX\_HAVE\_ISNANF**
- **#define \_GLIBCXX\_HAVE\_ISNANL**
- **#define \_GLIBCXX\_HAVE\_ISWBLANK**
- **#define \_GLIBCXX\_HAVE\_LC\_MESSAGES**
- **#define \_GLIBCXX\_HAVE\_LDEXPF**
- **#define \_GLIBCXX\_HAVE\_LDEXPL**
- **#define \_GLIBCXX\_HAVE\_LIBINTL\_H**
- **#define \_GLIBCXX\_HAVE\_LIMIT\_AS**
- **#define \_GLIBCXX\_HAVE\_LIMIT\_DATA**
- **#define \_GLIBCXX\_HAVE\_LIMIT\_FSIZE**
- **#define \_GLIBCXX\_HAVE\_LIMIT\_RSS**
- **#define \_GLIBCXX\_HAVE\_LIMIT\_VMEM**
- **#define \_GLIBCXX\_HAVE\_LINUX\_FUTEX**
- **#define \_GLIBCXX\_HAVE\_LOCALE\_H**
- **#define \_GLIBCXX\_HAVE\_LOG10F**

- `#define _GLIBCXX_HAVE_LOG10L`
- `#define _GLIBCXX_HAVE_LOGF`
- `#define _GLIBCXX_HAVE_LOGL`
- `#define _GLIBCXX_HAVE_MBSTATE_T`
- `#define _GLIBCXX_HAVE_MEMORY_H`
- `#define _GLIBCXX_HAVE_MODF`
- `#define _GLIBCXX_HAVE_MODFF`
- `#define _GLIBCXX_HAVE_MODFL`
- `#define _GLIBCXX_HAVE_POLL`
- `#define _GLIBCXX_HAVE_POWF`
- `#define _GLIBCXX_HAVE_POWL`
- `#define _GLIBCXX_HAVE_S_ISREG`
- `#define _GLIBCXX_HAVE_SETENV`
- `#define _GLIBCXX_HAVE_SINCOS`
- `#define _GLIBCXX_HAVE_SINCOSF`
- `#define _GLIBCXX_HAVE_SINCOSL`
- `#define _GLIBCXX_HAVE_SINF`
- `#define _GLIBCXX_HAVE_SINHF`
- `#define _GLIBCXX_HAVE_SINHL`
- `#define _GLIBCXX_HAVE_SINL`
- `#define _GLIBCXX_HAVE_SQRTF`
- `#define _GLIBCXX_HAVE_SQRTL`
- `#define _GLIBCXX_HAVE_STDBOOL_H`
- `#define _GLIBCXX_HAVE_STDINT_H`
- `#define _GLIBCXX_HAVE_STDLIB_H`
- `#define _GLIBCXX_HAVE_STRERROR_L`
- `#define _GLIBCXX_HAVE_STRERROR_R`
- `#define _GLIBCXX_HAVE_STRING_H`
- `#define _GLIBCXX_HAVE_STRINGS_H`
- `#define _GLIBCXX_HAVE_STRTOF`
- `#define _GLIBCXX_HAVE_STRTOLD`
- `#define _GLIBCXX_HAVE_STRXFRM_L`
- `#define _GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT`
- `#define _GLIBCXX_HAVE_SYS_IOCTL_H`
- `#define _GLIBCXX_HAVE_SYS_IPC_H`
- `#define _GLIBCXX_HAVE_SYS_PARAM_H`
- `#define _GLIBCXX_HAVE_SYS_RESOURCE_H`
- `#define _GLIBCXX_HAVE_SYS_SEM_H`
- `#define _GLIBCXX_HAVE_SYS_STAT_H`
- `#define _GLIBCXX_HAVE_SYS_TIME_H`
- `#define _GLIBCXX_HAVE_SYS_TYPES_H`
- `#define _GLIBCXX_HAVE_SYS_UIO_H`

- #define **\_GLIBCXX\_HAVE\_TANF**
- #define **\_GLIBCXX\_HAVE\_TANHF**
- #define **\_GLIBCXX\_HAVE\_TANHL**
- #define **\_GLIBCXX\_HAVE\_TANL**
- #define **\_GLIBCXX\_HAVE\_TGMATH\_H**
- #define **\_GLIBCXX\_HAVE\_TLS**
- #define **\_GLIBCXX\_HAVE\_UNISTD\_H**
- #define **\_GLIBCXX\_HAVE\_VFWSCANF**
- #define **\_GLIBCXX\_HAVE\_VSWSCANF**
- #define **\_GLIBCXX\_HAVE\_VWSCANF**
- #define **\_GLIBCXX\_HAVE\_WCHAR\_H**
- #define **\_GLIBCXX\_HAVE\_WCSTOF**
- #define **\_GLIBCXX\_HAVE\_WCTYPE\_H**
- #define **\_GLIBCXX\_HAVE\_WRITEV**
- #define **\_GLIBCXX\_HOSTED**
- #define **\_GLIBCXX\_ICONV\_CONST**
- #define **\_GLIBCXX\_LDBL\_NAMESPACE**
- #define **\_GLIBCXX\_NAMESPACE\_ASSOCIATION\_VERSION**
- #define **\_GLIBCXX\_NORETURN**
- #define **\_GLIBCXX\_NOTHROW**
- #define **\_GLIBCXX\_PACKAGE\_\_GLIBCXX\_VERSION**
- #define **\_GLIBCXX\_PACKAGE\_BUGREPORT**
- #define **\_GLIBCXX\_PACKAGE\_NAME**
- #define **\_GLIBCXX\_PACKAGE\_STRING**
- #define **\_GLIBCXX\_PACKAGE\_TARNAME**
- #define **\_GLIBCXX\_PACKAGE\_URL**
- #define **\_GLIBCXX\_PSEUDO\_VISIBILITY(V)**
- #define **\_GLIBCXX\_PURE**
- #define **\_GLIBCXX\_RES\_LIMITS**
- #define **\_GLIBCXX\_STD**
- #define **\_GLIBCXX\_STD\_D**
- #define **\_GLIBCXX\_STD\_P**
- #define **\_GLIBCXX\_STD\_PR**
- #define **\_GLIBCXX\_STDIO\_EOF**
- #define **\_GLIBCXX\_STDIO\_SEEK\_CUR**
- #define **\_GLIBCXX\_STDIO\_SEEK\_END**
- #define **\_GLIBCXX\_SYMVER**
- #define **\_GLIBCXX\_SYMVER\_GNU**
- #define **\_GLIBCXX\_SYNCHRONIZATION\_HAPPENS\_AFTER(A)**
- #define **\_GLIBCXX\_SYNCHRONIZATION\_HAPPENS\_BEFORE(A)**
- #define **\_GLIBCXX\_USE\_C99**
- #define **\_GLIBCXX\_USE\_C99\_COMPLEX**
- #define **\_GLIBCXX\_USE\_C99\_COMPLEX\_TR1**

- `#define _GLIBCXX_USE_C99_CTYPE_TR1`
- `#define _GLIBCXX_USE_C99_FENV_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1`
- `#define _GLIBCXX_USE_C99_MATH`
- `#define _GLIBCXX_USE_C99_MATH_TR1`
- `#define _GLIBCXX_USE_C99_STDINT_TR1`
- `#define _GLIBCXX_USE_DECIMAL_FLOAT`
- `#define _GLIBCXX_USE_GETTIMEOFDAY`
- `#define _GLIBCXX_USE_LFS`
- `#define _GLIBCXX_USE_LONG_LONG`
- `#define _GLIBCXX_USE_NLS`
- `#define _GLIBCXX_USE_RANDOM_TR1`
- `#define _GLIBCXX_USE_WCHAR_T`
- `#define _GLIBCXX_VISIBILITY_ATTR(V)`
- `#define _GLIBCXX_WEAK_DEFINITION`
- `#define LT_OBJDIR`
- `#define STDC_HEADERS`

## Typedefs

- `typedef __PTRDIFF_TYPE__ std::ptrdiff_t`
- `typedef __SIZE_TYPE__ std::size_t`

## Functions

- `typedef std::decltype (nullptr) nullptr_t`

### 6.41.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++config.h](#).

## 6.42 c++io.h File Reference

### Namespaces

- namespace [std](#)

## Typedefs

- typedef FILE **std::\_\_c\_file**
- typedef \_\_pthread\_mutex\_t **std::\_\_c\_lock**

### 6.42.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++io.h](#).

## 6.43 c++locale.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define **\_GLIBCXX\_C\_LOCALE\_GNU**
- #define **\_GLIBCXX\_NUM\_CATEGORIES**

### Typedefs

- typedef \_\_locale\_t **std::\_\_c\_locale**

### Functions

- int **std::\_\_convert\_from\_v** (const \_\_c\_locale &\_\_cloc \_\_attribute\_\_((\_\_unused\_\_)), char \*\_\_out, const int \_\_size \_\_attribute\_\_((\_\_unused\_\_)), const char \*\_\_fmt,...)

### 6.43.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++locale.h](#).

## 6.44 `c++locale_internal.h` File Reference

### 6.44.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [c++locale\\_internal.h](#).

## 6.45 `cassert` File Reference

### 6.45.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cassert](#).

## 6.46 `ccomplex` File Reference

### 6.46.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ccomplex](#).

## 6.47 `ccomplex` File Reference

### 6.47.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ccomplex](#).

## 6.48 cctype File Reference

### Namespaces

- namespace [std](#)

#### 6.48.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `ctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cctype](#).

## 6.49 cctype File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

#### 6.49.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cctype](#).

## 6.50 cctype File Reference

#### 6.50.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cctype](#).



## 6.51 cerrno File Reference

### Defines

- `#define errno`

### 6.51.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cerrno](#).

## 6.52 cfenv File Reference

### 6.52.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cfenv](#).

## 6.53 cfenv File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.53.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfenv](#).

## 6.54 `cfenv` File Reference

### 6.54.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cfenv](#).

## 6.55 `cfloat` File Reference

### Defines

- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

### 6.55.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cfloat](#).

## 6.56 `cfloat` File Reference

### Defines

- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

### 6.56.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cfloat](#).

## 6.57 char\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_Char\\_types<\\_CharT>](#)  
*Mapping from character type to associated types.*
- struct [\\_\\_gnu\\_cxx::char\\_traits<\\_CharT>](#)  
*Base class used to implement [std::char\\_traits](#).*
- struct [std::char\\_traits<\\_CharT>](#)  
*Basis for explicit traits specializations.*
- struct [std::char\\_traits<char>](#)  
*21.1.3.1 [char\\_traits](#) specializations*
- struct [std::char\\_traits<wchar\\_t>](#)  
*21.1.3.2 [char\\_traits](#) specializations*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

#### 6.57.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [char\\_traits.h](#).

## 6.58 checkers.h File Reference

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter, typename _Compare >`  
`bool \_\_gnu\_parallel::\_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __-`  
`comp)`

### 6.58.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [checkers.h](#).

## 6.59 chrono File Reference

### Classes

- struct [std::chrono::duration< \\_Rep, \\_Period >](#)  
*duration*
- struct [std::chrono::duration\\_values< \\_Rep >](#)  
*duration\_values*
- struct [std::chrono::system\\_clock](#)  
*system\_clock*
- struct [std::chrono::time\\_point< \\_Clock, \\_Duration >](#)  
*time\_point*
- struct [std::chrono::treat\\_as\\_floating\\_point< \\_Rep >](#)  
*treat\_as\_floating\_point*

### Namespaces

- namespace [std](#)
- namespace [std::chrono](#)

## Typedefs

- typedef system\_clock **std::chrono::high\_resolution\_clock**
- typedef duration< int, ratio< 3600 > > **std::chrono::hours**
- typedef duration< int64\_t, micro > **std::chrono::microseconds**
- typedef duration< int64\_t, milli > **std::chrono::milliseconds**
- typedef duration< int, ratio< 60 > > **std::chrono::minutes**
- typedef system\_clock **std::chrono::monotonic\_clock**
- typedef duration< int64\_t, nano > **std::chrono::nanoseconds**
- typedef duration< int64\_t > **std::chrono::seconds**

## Functions

- template<typename \_ToDuration, typename \_Rep, typename \_Period >  
enable\_if< \_\_is\_duration< \_ToDuration >::value, \_ToDuration >::type  
**std::chrono::duration\_cast** (const duration< \_Rep, \_Period > &\_\_d)
- template<typename \_Clock, typename \_Duration1, typename \_Duration2 >  
bool **std::chrono::operator!=** (const time\_point< \_Clock, \_Duration1 > &\_\_lhs,  
const time\_point< \_Clock, \_Duration2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
bool **std::chrono::operator!=** (const duration< \_Rep1, \_Period1 > &\_\_lhs,  
const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
common\_type< duration< \_Rep1, \_Period1 >, duration< \_Rep2, \_Period2 >  
>::type **std::chrono::operator%** (const duration< \_Rep1, \_Period1 > &\_\_lhs,  
const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
duration< typename \_\_common\_rep\_type< \_Rep1, typename enable\_  
if<!\_\_is\_duration< \_Rep2 >::value, \_Rep2 >::type >::type, \_Period >  
**std::chrono::operator%** (const duration< \_Rep1, \_Period > &\_\_d, const \_  
Rep2 &\_\_s)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
duration< typename \_\_common\_rep\_type< \_Rep1, \_Rep2 >::type, \_Period >  
**std::chrono::operator\*** (const duration< \_Rep1, \_Period > &\_\_d, const \_Rep2  
&\_\_s)
- template<typename \_Rep1, typename \_Period, typename \_Rep2 >  
duration< typename \_\_common\_rep\_type< \_Rep2, \_Rep1 >::type, \_Period >  
**std::chrono::operator\*** (const \_Rep1 &\_\_s, const duration< \_Rep2, \_Period >  
&\_\_d)
- template<typename \_Rep1, typename \_Period1, typename \_Rep2, typename \_Period2 >  
common\_type< duration< \_Rep1, \_Period1 >, duration< \_Rep2, \_Period2 >  
>::type **std::chrono::operator+** (const duration< \_Rep1, \_Period1 > &\_\_lhs,  
const duration< \_Rep2, \_Period2 > &\_\_rhs)

- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >  
time_point< _Clock, typename common_type< _Duration1, duration< _Rep2, _Period2 > >::type > std::chrono::operator+ (const time_point< _Clock, _Duration1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Duration2 >  
time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Duration2 >::type > std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Rep2, typename _Period2 >  
time_point< _Clock, typename common_type< _Duration1, duration< _Rep2, _Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _Duration1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
common_type< _Duration1, _Duration2 >::type std::chrono::operator- (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >  
duration< typename __common_rep_type< _Rep1, typename enable_if<!__is_duration< _Rep2 >::value, _Rep2 >::type >::type, _Period > std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator< (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >  
bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator<= (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >  
bool std::chrono::operator== (const time_point< _Clock, _Duration1 > &__lhs, const time_point< _Clock, _Duration2 > &__rhs)`

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`bool std::chrono::operator== (const duration< _Rep1, _Period1 > &__lhs,`  
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
`bool std::chrono::operator> (const time_point< _Clock, _Duration1 > &__-`  
`lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs,`  
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Duration1, typename _Duration2 >`  
`bool std::chrono::operator>= (const time_point< _Clock, _Duration1 > &__-`  
`_lhs, const time_point< _Clock, _Duration2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
`bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs,`  
`const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _ToDuration, typename _Clock, typename _Duration >`  
`enable_if< __is_duration< _ToDuration >::value, time_point< _Clock, _-`  
`ToDuration > >::type std::chrono::time_point_cast (const time_point< _Clock,`  
`_Duration > &__t)`

### 6.59.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [chrono](#).

## 6.60 cinttypes File Reference

### 6.60.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cinttypes](#).

## 6.61 cinttypes File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.61.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cinttypes](#).

## 6.62 cinttypes File Reference

### 6.62.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cinttypes](#).

## 6.63 ciso646 File Reference

### 6.63.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ciso646](#).

## 6.64 climits File Reference

### Defines

- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

### 6.64.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.



This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [climits](#).

## 6.65 `climits` File Reference

### Defines

- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

### 6.65.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/climits](#).

## 6.66 `locale` File Reference

### Namespaces

- namespace [std](#)

### 6.66.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [clocale](#).

## 6.67 cmath File Reference

### Namespaces

- namespace [std](#)

### Functions

- double **std::abs** (double \_\_x)
- float **std::abs** (float \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::abs** (\_Tp \_\_x)
- long double **std::abs** (long double \_\_x)
- float **std::acos** (float \_\_x)
- long double **std::acos** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::acos** (\_Tp \_\_x)
- float **std::asin** (float \_\_x)
- long double **std::asin** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::asin** (\_Tp \_\_x)
- float **std::atan** (float \_\_x)
- long double **std::atan** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::atan** (\_Tp \_\_x)
- template<typename \_Tp, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< typename \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_arithmetic< \_Tp >::\_\_value && \_\_is\_arithmetic< \_Up >::\_\_value, \_Tp >::\_\_type, \_Up >::\_\_type **std::atan2** (\_Tp \_\_y, \_Up \_\_x)
- float **std::atan2** (float \_\_y, float \_\_x)
- long double **std::atan2** (long double \_\_y, long double \_\_x)
- float **std::ceil** (float \_\_x)
- long double **std::ceil** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type  
**std::ceil** (\_Tp \_\_x)
- float **std::cos** (float \_\_x)
- long double **std::cos** (long double \_\_x)

- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::cos (_Tp __x)`
- `float std::cosh (float __x)`
- `long double std::cosh (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::cosh (_Tp __x)`
- `float std::exp (float __x)`
- `long double std::exp (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::exp (_Tp __x)`
- `float std::fabs (float __x)`
- `long double std::fabs (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::fabs (_Tp __x)`
- `float std::floor (float __x)`
- `long double std::floor (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::floor (_Tp __x)`
- `float std::fmod (float __x, float __y)`
- `long double std::fmod (long double __x, long double __y)`
- `float std::frexp (float __x, int *__exp)`
- `long double std::frexp (long double __x, int *__exp)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::frexp (_Tp __x, int *__exp)`
- `float std::ldexp (float __x, int __exp)`
- `long double std::ldexp (long double __x, int __exp)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::ldexp (_Tp __x, int __exp)`
- `long double std::log (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::log (_Tp __x)`
- `float std::log (float __x)`
- `long double std::log10 (long double __x)`
- `float std::log10 (float __x)`

- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::log10 (_Tp __x)`
- `long double std::modf (long double __x, long double *__iptr)`
- `float std::modf (float __x, float *__iptr)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< typename __gnu_cxx::__enable_if< __is_`  
`arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value, _Tp`  
`>::__type, _Up >::__type std::pow (_Tp __x, _Up __y)`
- `float std::pow (float __x, float __y)`
- `long double std::pow (long double __x, long double __y)`
- `float std::sin (float __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::sin (_Tp __x)`
- `long double std::sin (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::sinh (_Tp __x)`
- `long double std::sinh (long double __x)`
- `float std::sinh (float __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::sqrt (_Tp __x)`
- `long double std::sqrt (long double __x)`
- `float std::sqrt (float __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::tan (_Tp __x)`
- `long double std::tan (long double __x)`
- `float std::tan (float __x)`
- `long double std::tanh (long double __x)`
- `float std::tanh (float __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type`  
`std::tanh (_Tp __x)`

### 6.67.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cmath](#).

## 6.68 cmath File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_laguerre (unsigned int`  
`__n, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc\_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::tr1::assoc\_laguerrel (unsigned int __n, unsigned int __m, long`  
`double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc\_legendre (unsigned int`  
`__l, unsigned int __m, _Tp __x)`
- `float std::tr1::assoc\_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::tr1::assoc\_legendrel (unsigned int __l, unsigned int __m, long`  
`double __x)`
- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (_Tpx __x, _-`  
`Tpy __y)`
- `float std::tr1::betaf (float __x, float __y)`
- `long double std::tr1::betal (long double __x, long double __y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_1 (_Tp __k)`
- `float std::tr1::comp\_ellint\_1f (float __k)`
- `long double std::tr1::comp\_ellint\_1l (long double __k)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp\_ellint\_2 (_Tp __k)`
- `float std::tr1::comp\_ellint\_2f (float __k)`

- long double **std::tr1::comp\_ellint\_2l** (long double \_\_k)
- template<typename \_Tp, typename \_Tpn >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpn >::\_\_type **std::tr1::comp\_ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu)
- float **std::tr1::comp\_ellint\_3f** (float \_\_k, float \_\_nu)
- long double **std::tr1::comp\_ellint\_3l** (long double \_\_k, long double \_\_nu)
- template<typename \_Tpa, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tpa, \_Tpc, \_Tp >::\_\_type **std::tr1::conf\_hyperg** (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::conf\_hypergf** (float \_\_a, float \_\_c, float \_\_x)
- long double **std::tr1::conf\_hypergl** (long double \_\_a, long double \_\_c, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_i** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_if** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_il** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_j** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_jf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_jl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_bessel\_k** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_bessel\_kf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_bessel\_kl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tpnu, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tpnu, \_Tp >::\_\_type **std::tr1::cyl\_neumann** (\_Tpnu \_\_nu, \_Tp \_\_x)
- float **std::tr1::cyl\_neumannf** (float \_\_nu, float \_\_x)
- long double **std::tr1::cyl\_neumannl** (long double \_\_nu, long double \_\_x)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_1** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_1f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **std::tr1::ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)

- `template<typename _Tp, typename _Tpn, typename _Tpp >`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint\_3 (_Tp`  
`__k, _Tpn __nu, _Tpp __phi)`
- `float std::tr1::ellint\_3f (float __k, float __nu, float __phi)`
- `long double std::tr1::ellint\_3l (long double __k, long double __nu, long double`  
`__phi)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)`
- `float std::tr1::expintf (float __x)`
- `long double std::tr1::expintl (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp`  
`__x)`
- `float std::tr1::hermitef (unsigned int __n, float __x)`
- `long double std::tr1::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg`  
`(_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float std::tr1::hypergf (float __a, float __b, float __c, float __x)`
- `long double std::tr1::hypergl (long double __a, long double __b, long double`  
`__c, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _-`  
`Tp __x)`
- `float std::tr1::laguerref (unsigned int __n, float __x)`
- `long double std::tr1::laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _-`  
`Tp __x)`
- `float std::tr1::legendref (unsigned int __n, float __x)`
- `long double std::tr1::legendrel (unsigned int __n, long double __x)`
- `float std::tr1::pow (float __x, float __y)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type std::tr1::pow (_Tp __x, _Up`  
`__y)`
- `double std::tr1::pow (double __x, double __y)`
- `long double std::tr1::pow (long double __x, long double __y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::riemann\_zeta (_Tp __x)`
- `float std::tr1::riemann\_zetaf (float __x)`
- `long double std::tr1::riemann\_zetal (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph\_bessel (unsigned int __n,`  
`_Tp __x)`

- float **std::tr1::sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
   \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **std::tr1::sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **std::tr1::sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
   \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_neumannl** (unsigned int \_\_n, long double \_\_x)

### 6.68.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cmath](#).

## 6.69 cmath File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.69.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cmath](#).

## 6.70 codecvt.h File Reference

### Classes

- class [std::\\_\\_codecvt\\_abstract\\_base](#)< \_InternT, \_ExternT, \_StateT >  
*Common base for codecvt functions.*



- class [std::codecvt< \\_InternT, \\_ExternT, \\_StateT >](#)  
*Primary class template codecvt.*  
*NB: Generic, mostly useless implementation.*
- class [std::codecvt< char, char, mbstate\\_t >](#)  
*class [codecvt<char, char, mbstate\\_t>](#) specialization.*
- class [std::codecvt< wchar\\_t, char, mbstate\\_t >](#)  
*class [codecvt<wchar\\_t, char, mbstate\\_t>](#) specialization.*
- class [std::codecvt\\_base](#)  
*Empty base class for codecvt facet [22.2.1.5].*
- class [std::codecvt\\_byname< \\_InternT, \\_ExternT, \\_StateT >](#)  
*class [codecvt\\_byname](#) [22.2.1.6].*

## Namespaces

- namespace [std](#)

### 6.70.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [codecvt.h](#).

## 6.71 codecvt\_specializations.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::encoding\\_char\\_traits< \\_CharT >](#)  
*[encoding\\_char\\_traits](#)*
- class [\\_\\_gnu\\_cxx::encoding\\_state](#)  
*Extension to use iconv for dealing with character encodings.*
- class [std::codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#)  
*[codecvt<InternT, \\_ExternT, encoding\\_state>](#) specialization.*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- `template<typename _Tp >  
size_t std::__iconv_adaptor (size_t(*__func)(iconv_t, _Tp, size_t *, char **,  
size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf,  
size_t *__outbytes)`

### 6.71.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [codecvt\\_specializations.h](#).

## 6.72 compatibility.h File Reference

### 6.72.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

Definition in file [x86\\_64-unknown-linux-gnu/bits/compatibility.h](#).

## 6.73 compatibility.h File Reference

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Tp >  
bool \_\_gnu\_parallel::\_\_compare\_and\_swap (volatile _Tp *__ptr, _Tp __-  
comparand, _Tp __replacement)`

- `bool __gnu_parallel::__compare_and_swap_32` (volatile `int32_t *``__ptr`, `int32_t` `__comparand`, `int32_t` `__replacement`)
- `bool __gnu_parallel::__compare_and_swap_64` (volatile `int64_t *``__ptr`, `int64_t` `__comparand`, `int64_t` `__replacement`)
- `template<typename _Tp >`  
`_Tp __gnu_parallel::__fetch_and_add` (volatile `_Tp *``__ptr`, `_Tp` `__addend`)
- `int32_t __gnu_parallel::__fetch_and_add_32` (volatile `int32_t *``__ptr`, `int32_t` `__addend`)
- `int64_t __gnu_parallel::__fetch_and_add_64` (volatile `int64_t *``__ptr`, `int64_t` `__addend`)
- `void __gnu_parallel::__yield` ()

### 6.73.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/compatibility.h](#).

## 6.74 `compiletime_settings.h` File Reference

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

### Defines

- `#define _GLIBCXX_ASSERTIONS`
- `#define _GLIBCXX_CALL(__n)`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
- `#define _GLIBCXX_SCALE_DOWN_FPU`
- `#define _GLIBCXX_VERBOSE_LEVEL`

### 6.74.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [compiletime\\_settings.h](#).

## 6.74.2 Define Documentation

### 6.74.2.1 `#define _GLIBCXX_ASSERTIONS`

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Definition at line 61 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

### 6.74.2.2 `#define _GLIBCXX_CALL( __n )`

Macro to produce log message when entering a function.

#### Parameters

`__n` Input size.

#### See also

[\\_GLIBCXX\\_VERBOSE\\_LEVEL](#)

Definition at line 44 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__multiway_merge()`, `__gnu_parallel::__multiway_merge_3_variant()`, `__gnu_parallel::__multiway_merge_4_variant()`, `__gnu_parallel::__multiway_merge_loser_tree()`, `__gnu_parallel::__multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::__multiway_merge_loser_tree_unguarded()`, `__gnu_parallel::__multiway_merge_sentinels()`, `__gnu_parallel::__parallel_multiway_merge()`, and `__gnu_parallel::__parallel_sort_mwms()`.

### 6.74.2.3 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Consider the size of the L1 cache for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 68 of file `compiletime_settings.h`.

**6.74.2.4 #define \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_TLB**

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the TLB for `gnu_parallel::__parallel_random_shuffle()`.

Definition at line 74 of file `completetime_settings.h`.

**6.74.2.5 #define \_GLIBCXX\_SCALE\_DOWN\_FPU**

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `completetime_settings.h`.

**6.74.2.6 #define \_GLIBCXX\_VERBOSE\_LEVEL**

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `completetime_settings.h`.

**6.75 complex File Reference****Classes**

- struct [std::complex<\\_Tp>](#)

**Namespaces**

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

**Functions**

- `template<typename _Tp>`  
`_Tp std::__complex_abs (const complex<_Tp> &__z)`
- `template<typename _Tp>`  
`_Tp std::__complex_arg (const complex<_Tp> &__z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_cos (const complex<_Tp> &__z)`
- `template<typename _Tp>`  
`complex<_Tp> std::__complex_cosh (const complex<_Tp> &__z)`

- `template<typename _Tp >`  
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_proj (const std::complex< _Tp > &__-`  
`z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`\_\_gnu\_cxx::\_\_promote< _Tp >::__type std::conj (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x)`

- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::proj (_Tp __x)`
- `template<typename _Tp >`  
`_Tp std::real (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp >`  
`&__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp >`  
`&__y)`
  
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const`  
`complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp`  
`&__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp >`  
`&__y)`
  
- `template<typename _Tp >`  
`bool std::operator== (const complex< _Tp > &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp >`  
`&__y)`
- `template<typename _Tp >`  
`bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`



### 6.75.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex](#).

## 6.76 complex File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### Functions

- `template<typename _Tp >  
std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
std::complex< typename __gnu_cxx::__promote< _Tp >::__type >  
std::tr1::conj (_Tp __x)`
- `template<typename _Tp, typename _Up >  
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >  
std::tr1::polar (const _Tp &__rho, const _Up &__theta)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const  
_Tp &__y)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp  
> &__y)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const  
std::complex< _Tp > &__y)`

### 6.76.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/complex](#).

## 6.77 complex File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- `template<typename _Tp >  
std::complex< _Tp > std::tr1::\_\_complex\_acos (const std::complex< _Tp >  
&__z)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::\_\_complex\_acosh (const std::complex< _Tp >  
&__z)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::\_\_complex\_asin (const std::complex< _Tp >  
&__z)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::\_\_complex\_asinh (const std::complex< _Tp >  
&__z)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::\_\_complex\_atan (const std::complex< _Tp >  
&__z)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::\_\_complex\_atanh (const std::complex< _Tp >  
&__z)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
__gnu_cxx::__promote< _Tp >::__type std::tr1::arg (_Tp __x)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >  
std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)`

- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::imag (_Tp)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::norm (_Tp __x)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type >`  
`std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Up >`  
`&__y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::real (_Tp __x)`

### 6.77.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/complex](#).

## 6.78 complex.h File Reference

### 6.78.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [complex.h](#).

## 6.79 `concept_check.h` File Reference

### Defines

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

### 6.79.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [concept\\_check.h](#).

## 6.80 `concurrency.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_scoped\\_lock](#)  
*Scoped lock idiom.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

### Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

## Variables

- static const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

### 6.80.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [concurrency.h](#).

## 6.81 cond\_dealtor.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_COND_DEALTOR_CLASS_C_DEC`
- `#define PB_DS_COND_DEALTOR_CLASS_T_DEC`

### 6.81.1 Detailed Description

Contains a conditional deallocator.

Definition in file [cond\\_dealtor.hpp](#).

## 6.82 condition\_variable File Reference

### Classes

- class [std::condition\\_variable](#)  
*condition\_variable*
- class [std::condition\\_variable\\_any](#)  
*condition\_variable\_any*

## Namespaces

- namespace [std](#)

## Enumerations

- enum [std::cv\\_status](#) { **no\_timeout**, **timeout** }

### 6.82.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [condition\\_variable](#).

## 6.83 constructors\_destructor\_fn\_imps.hpp File Reference

### Functions

- **PB\_DS\_CLASS\_NAME** ()
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 >  
**PB\_DS\_CLASS\_NAME** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >  
**PB\_DS\_CLASS\_NAME** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >  
**PB\_DS\_CLASS\_NAME** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >  
**PB\_DS\_CLASS\_NAME** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >  
**PB\_DS\_CLASS\_NAME** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0 , typename T1 , typename T2 , typename T3 >  
**PB\_DS\_CLASS\_NAME** (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0 , typename T1 , typename T2 >  
**PB\_DS\_CLASS\_NAME** (T0 t0, T1 t1, T2 t2)
- template<typename T0 , typename T1 >  
**PB\_DS\_CLASS\_NAME** (T0 t0, T1 t1)

- `template<typename T0 >`  
`PB_DS_CLASS_NAME` (T0 t0)
- `PB_DS_CLASS_NAME` (const PB\_DS\_CLASS\_NAME &other)

### 6.83.1 Detailed Description

Contains `constructors_destructor_fn_imps` applicable to different containers.

Definition in file [constructors\\_destructor\\_fn\\_imps.hpp](#).

## 6.84 container\_base\_dispatch.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.84.1 Detailed Description

Contains an associative container dispatching base.

Definition in file [container\\_base\\_dispatch.hpp](#).

## 6.85 cpp\_type\_traits.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### 6.85.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cpp\\_type\\_traits.h](#).

## 6.86 `cpu_defines.h` File Reference

### 6.86.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [cpu\\_defines.h](#).

## 6.87 `csetjmp` File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `setjmp(env)`

### 6.87.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csetjmp](#).

## 6.88 `csignal` File Reference

### Namespaces

- namespace [std](#)

### 6.88.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.



This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [csignal](#).

## 6.89 cstdarg File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define `va_end(ap)`

#### 6.89.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdarg](#).

## 6.90 cstdarg File Reference

#### 6.90.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdarg](#).

## 6.91 cstdbool File Reference

#### 6.91.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdbool](#).

## 6.92 cstdint File Reference

### 6.92.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdint](#).

## 6.93 cstdint File Reference

### 6.93.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `stdint.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdint](#).

## 6.94 cstdint File Reference

### 6.94.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [cstdint](#).

## 6.95 cstdint File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.95.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdint](#).

## 6.96 cstdint File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

#### 6.96.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdint](#).

## 6.97 cstdio File Reference

### Namespaces

- namespace [std](#)

#### 6.97.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdio](#).

## 6.98 cstdio File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.98.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdio](#).

## 6.99 cstdio File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.99.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdio](#).

## 6.100 cstdlib File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define EXIT_FAILURE`
- `#define EXIT_SUCCESS`

### Functions

- void **std::abort** (void) \_GLIBCXX\_NORETURN throw ()
- int **std::atexit** (void(\*)()) throw ()
- void **std::exit** (int) \_GLIBCXX\_NORETURN throw ()

### 6.100.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstdlib](#).

## 6.101 cstdlib File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.101.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cstdlib](#).

## 6.102 cstdlib File Reference

### 6.102.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cstdlib](#).

## 6.103 cstring File Reference

### Namespaces

- namespace [std](#)

## Functions

- void \* **std::memchr** (void \*\_\_s, int \_\_c, size\_t \_\_n)
- char \* **std::strchr** (char \*\_\_s, int \_\_n)
- char \* **std::strpbrk** (char \*\_\_s1, const char \*\_\_s2)
- char \* **std::strrchr** (char \*\_\_s, int \_\_n)
- char \* **std::strstr** (char \*\_\_s1, const char \*\_\_s2)

### 6.103.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cstring](#).

## 6.104 ctgmath File Reference

### 6.104.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ctgmath](#).

## 6.105 ctgmath File Reference

### 6.105.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctgmath](#).

## 6.106 ctime File Reference

## Namespaces

- namespace [std](#)

### 6.106.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [ctime](#).

## 6.107 ctime File Reference

### 6.107.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/ctime](#).

## 6.108 ctype\_base.h File Reference

### Classes

- struct [std::ctype\\_base](#)

*Base class for ctype.*

### Namespaces

- namespace [std](#)

### 6.108.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_base.h](#).

## 6.109 ctype\_inline.h File Reference

### Namespaces

- namespace [std](#)

#### 6.109.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_inline.h](#).

## 6.110 ctype\_noninline.h File Reference

#### 6.110.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ctype\\_noninline.h](#).

## 6.111 cwchar File Reference

### Namespaces

- namespace [std](#)

### Functions

- `wchar_t * std::wcschr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcpbrk (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wcsrchr (wchar_t *__p, wchar_t __c)`
- `wchar_t * std::wcsstr (wchar_t *__s1, const wchar_t *__s2)`
- `wchar_t * std::wmemchr (wchar_t *__p, wchar_t __c, size_t __n)`

#### 6.111.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the \*.h implementation files.



This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cwchar](#).

## 6.112 `wchar` File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.112.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwchar](#).

## 6.113 `wchar` File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.113.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cwchar](#).

## 6.114 `cwctype` File Reference

### Namespaces

- namespace [std](#)

## Defines

- `#define _GLIBCXX_CWCTYPE`

### 6.114.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file [cwctype](#).

## 6.115 cwctype File Reference

### Defines

- `#define _GLIBCXX_BEGIN_NAMESPACE_TR1`
- `#define _GLIBCXX_END_NAMESPACE_TR1`
- `#define _GLIBCXX_TR1`

### 6.115.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file [tr1/cwctype](#).

## 6.116 cwctype File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### 6.116.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/cwctype](#).

## 6.117 cxxabi-forced.h File Reference

### Classes

- class [\\_\\_cxxabiv1::\\_\\_forced\\_unwind](#)

*Thrown as part of forced unwinding.*

*A magic placeholder class that can be caught by reference to recognize forced unwinding.*

### 6.117.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi-forced.h](#).

## 6.118 cxxabi.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)

*Exception thrown by `__cxa_guard_acquire`.*

*6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [abi](#)

### Defines

- `#define _GLIBCXX_NOTHROW`

### Typedefs

- typedef `__cxa_cdtor_return_type(* __cxxabiv1::__cxa_cdtor_type)(void *)`

## Functions

- `int __cxxabiv1::__cxa_atexit (void(*)(void *), void *, void *) throw ()`
- `void __cxxabiv1::__cxa_bad_cast ()`
- `void __cxxabiv1::__cxa_bad_typeid ()`
- `std::type_info * __cxxabiv1::__cxa_current_exception_type () __attribute__((__pure__)) throw ()`
- `char * __cxxabiv1::__cxa_demangle (const char * __mangled_name, char * __output_buffer, size_t * __length, int * __status)`
- `int __cxxabiv1::__cxa_finalize (void *)`
- `void __cxxabiv1::__cxa_guard_abort (__guard *) throw ()`
- `int __cxxabiv1::__cxa_guard_acquire (__guard *)`
- `void __cxxabiv1::__cxa_guard_release (__guard *) throw ()`
- `void __cxxabiv1::__cxa_pure_virtual (void) __attribute__((__noreturn__))`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void * __dest_array, void * __src_array, size_t __element_count, size_t __element_size, __cxa_ctor_return_type(* __constructor)(void *, void *), __cxa_ctor_type __destructor)`
- `void __cxxabiv1::__cxa_vec_cleanup (void * __array_address, size_t __element_count, size_t __s, __cxa_ctor_type __destructor) throw ()`
- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor (void * __array_address, size_t __element_count, size_t __element_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor)`
- `void __cxxabiv1::__cxa_vec_delete (void * __array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type __destructor)`
- `void __cxxabiv1::__cxa_vec_delete2 (void * __array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type __destructor, void(* __dealloc)(void *))`
- `void __cxxabiv1::__cxa_vec_delete3 (void * __array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type __destructor, void(* __dealloc)(void *, size_t))`
- `void __cxxabiv1::__cxa_vec_dtor (void * __array_address, size_t __element_count, size_t __element_size, __cxa_ctor_type __destructor)`
- `void * __cxxabiv1::__cxa_vec_new (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor)`
- `void * __cxxabiv1::__cxa_vec_new2 (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor, void(* __alloc)(size_t), void(* __dealloc)(void *))`
- `void * __cxxabiv1::__cxa_vec_new3 (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor, void(* __alloc)(size_t), void(* __dealloc)(void *, size_t))`
- `void * __cxxabiv1::__dynamic_cast (const void * __src_ptr, const __class_type_info * __src_type, const __class_type_info * __dst_type, ptrdiff_t __src2dst)`

### 6.118.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file [cxxabi.h](#).

## 6.119 cxxabi\_tweaks.h File Reference

### Defines

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

### Typedefs

- `typedef void __cxxabiv1::__cxa_cdtor_return_type`
- `typedef void __cxxabiv1::__cxa_vec_ctor_return_type`

### Functions

- `__extension__ typedef int __guard __cxxabiv1::__attribute__ ((mode(__DI_ - __)))`

### 6.119.1 Detailed Description

The header provides an CPU-variable interface to the C++ ABI.

Definition in file [cxxabi\\_tweaks.h](#).

## 6.120 debug.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)
- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Defines

- #define `__glibcxx_requires_cond`(\_Cond, \_Msg)
- #define `__glibcxx_requires_heap`(\_First, \_Last)
- #define `__glibcxx_requires_heap_pred`(\_First, \_Last, \_Pred)
- #define `__glibcxx_requires_nonempty`()
- #define `__glibcxx_requires_partitioned_lower`(\_First, \_Last, \_Value)
- #define `__glibcxx_requires_partitioned_lower_pred`(\_First, \_Last, \_Value, \_Pred)
- #define `__glibcxx_requires_partitioned_upper`(\_First, \_Last, \_Value)
- #define `__glibcxx_requires_partitioned_upper_pred`(\_First, \_Last, \_Value, \_Pred)
- #define `__glibcxx_requires_sorted`(\_First, \_Last)
- #define `__glibcxx_requires_sorted_pred`(\_First, \_Last, \_Pred)
- #define `__glibcxx_requires_sorted_set`(\_First1, \_Last1, \_First2)
- #define `__glibcxx_requires_sorted_set_pred`(\_First1, \_Last1, \_First2, \_Pred)
- #define `__glibcxx_requires_string`(\_String)
- #define `__glibcxx_requires_string_len`(\_String, \_Len)
- #define `__glibcxx_requires_subscript`(\_N)
- #define `__glibcxx_requires_valid_range`(\_First, \_Last)
- #define `_GLIBCXX_DEBUG_ASSERT`(\_Condition)
- #define `_GLIBCXX_DEBUG_ONLY`(\_Statement)
- #define `_GLIBCXX_DEBUG_PEDASSERT`(\_Condition)

### 6.120.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug.h](#).

## 6.121 `debug_allocator.h` File Reference

### Classes

- class `__gnu_cxx::debug_allocator< _Alloc >`  
*A meta-allocator with debugging bits, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls operator new
  - all deallocation calls operator delete.

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.121.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. You should only include this header if you are using GCC 3 or later.

Definition in file [debug\\_allocator.h](#).

## 6.122 debug\_map\_base.hpp File Reference

### 6.122.1 Detailed Description

Contains a debug-mode base for all maps.

Definition in file [debug\\_map\\_base.hpp](#).

## 6.123 decimal File Reference

### Classes

- class [std::decimal::decimal128](#)  
*3.2.4 Class [decimal128](#).*
- class [std::decimal::decimal32](#)  
*3.2.2 Class [decimal32](#).*
- class [std::decimal::decimal64](#)  
*3.2.3 Class [decimal64](#).*

## Namespaces

- namespace [std](#)
- namespace [std::decimal](#)

## Defines

- #define **\_DECLARE\_DECIMAL128\_COMPOUND\_ASSIGNMENT**(\_Op)
- #define **\_DECLARE\_DECIMAL32\_COMPOUND\_ASSIGNMENT**(\_Op)
- #define **\_DECLARE\_DECIMAL64\_COMPOUND\_ASSIGNMENT**(\_Op)
- #define **\_DECLARE\_DECIMAL\_BINARY\_OP\_WITH\_DEC**(\_Op, \_T1, \_T2, \_T3)
- #define **\_DECLARE\_DECIMAL\_BINARY\_OP\_WITH\_INT**(\_Op, \_Tp)
- #define **\_DECLARE\_DECIMAL\_COMPARISON**(\_Op, \_Tp)
- #define **\_GLIBCXX\_USE\_DECIMAL\_**

## Functions

- double **std::decimal::decimal128\_to\_double** (decimal128 \_\_d)
- float **std::decimal::decimal128\_to\_float** (decimal128 \_\_d)
- long double **std::decimal::decimal128\_to\_long\_double** (decimal128 \_\_d)
- long long **std::decimal::decimal128\_to\_long\_long** (decimal128 \_\_d)
- double **std::decimal::decimal32\_to\_double** (decimal32 \_\_d)
- float **std::decimal::decimal32\_to\_float** (decimal32 \_\_d)
- long double **std::decimal::decimal32\_to\_long\_double** (decimal32 \_\_d)
- long long **std::decimal::decimal32\_to\_long\_long** (decimal32 \_\_d)
- double **std::decimal::decimal64\_to\_double** (decimal64 \_\_d)
- float **std::decimal::decimal64\_to\_float** (decimal64 \_\_d)
- long double **std::decimal::decimal64\_to\_long\_double** (decimal64 \_\_d)
- long long **std::decimal::decimal64\_to\_long\_long** (decimal64 \_\_d)
- double **std::decimal::decimal\_to\_double** (decimal32 \_\_d)
- double **std::decimal::decimal\_to\_double** (decimal64 \_\_d)
- double **std::decimal::decimal\_to\_double** (decimal128 \_\_d)
- float **std::decimal::decimal\_to\_float** (decimal32 \_\_d)
- float **std::decimal::decimal\_to\_float** (decimal64 \_\_d)
- float **std::decimal::decimal\_to\_float** (decimal128 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal32 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal64 \_\_d)
- long double **std::decimal::decimal\_to\_long\_double** (decimal128 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal32 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal64 \_\_d)
- long long **std::decimal::decimal\_to\_long\_long** (decimal128 \_\_d)
- static decimal128 **std::decimal::make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static decimal128 **std::decimal::make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal32 **std::decimal::make\_decimal32** (long long \_\_coeff, int \_\_exp)



- static decimal32 **std::decimal::make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal64 **std::decimal::make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- static decimal64 **std::decimal::make\_decimal64** (long long \_\_coeff, int \_\_exp)
  
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator!=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator!=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator!=** (decimal128 \_\_lhs, unsigned long \_\_rhs)

- `bool std::decimal::operator!= (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator!= (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator!= (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (unsigned long long __lhs, decimal128 __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned int __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, int __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, long long __rhs)`
- `decimal32 std::decimal::operator* (decimal32 __lhs, unsigned long long __-  
rhs)`
- `decimal32 std::decimal::operator* (int __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned int __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (long long __lhs, decimal32 __rhs)`
- `decimal32 std::decimal::operator* (unsigned long long __lhs, decimal32 __-  
rhs)`
- `decimal64 std::decimal::operator* (decimal32 __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, decimal32 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, int __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned int __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, long long __rhs)`
- `decimal64 std::decimal::operator* (decimal64 __lhs, unsigned long long __-  
rhs)`
- `decimal64 std::decimal::operator* (int __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned int __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (long long __lhs, decimal64 __rhs)`
- `decimal64 std::decimal::operator* (unsigned long long __lhs, decimal64 __-  
rhs)`
- `decimal128 std::decimal::operator* (decimal32 __lhs, decimal128 __rhs)`
- `decimal128 std::decimal::operator* (decimal64 __lhs, decimal128 __rhs)`
- `decimal128 std::decimal::operator* (decimal128 __lhs, decimal32 __rhs)`
- `decimal128 std::decimal::operator* (decimal128 __lhs, decimal64 __rhs)`

- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator+** (int \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal128 \_\_rhs)

- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator+** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator+** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator-** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long long \_\_lhs, decimal32 \_\_rhs)

- decimal32 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator-** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator-** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)

- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator/** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator/** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator/** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (long \_\_lhs, decimal128 \_\_rhs)

- decimal128 **std::decimal::operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal64 \_\_rhs)

- `bool std::decimal::operator< (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator< (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator< (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator== (long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator== (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator== (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator== (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator== (decimal64 __lhs, unsigned long __rhs)`



- `bool std::decimal::operator==(decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator==(int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator==(decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator==(unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>(long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>(unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>(long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>(decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>(decimal128 __lhs, long long __rhs)`

- `bool std::decimal::operator> (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator> (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator> (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator> (decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator> (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal128 __rhs)`

- bool **std::decimal::operator>=** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator>=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator>=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator>=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator>=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator>=** (decimal128 \_\_lhs, decimal64 \_\_rhs)

### 6.123.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [decimal](#).

## 6.124 deque File Reference

### 6.124.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [deque](#).

## 6.125 deque File Reference

### Classes

- class [std::\\_\\_debug::deque< \\_Tp, \\_Allocator >](#)  
*Class [std::deque](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

### 6.125.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/deque](#).

## 6.126 deque File Reference

### Classes

- class [std::\\_\\_profile::deque< \\_Tp, \\_Allocator >](#)  
*Class [std::deque](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)`

### 6.126.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/deque](#).

## 6.127 deque.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`  
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _`  
`Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _Tp &__value)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp,`  
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`  
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`  
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`  
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`  
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`

### 6.127.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [deque.tcc](#).

## 6.128 `enc_filebuf.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::enc\\_filebuf<\\_CharT>](#)  
*class [enc\\_filebuf](#).*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### 6.128.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [enc\\_filebuf.h](#).

## 6.129 `equally_split.h` File Reference

### Namespaces

- namespace `__gnu_parallel`

### Functions

- `template<typename _DifferenceType, typename _OutputIterator >  
_OutputIterator \_\_gnu\_parallel::equally\_split (_DifferenceType __n, _-  
ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType >  
_DifferenceType \_\_gnu\_parallel::equally\_split\_point (_DifferenceType __n, _-  
ThreadIndex __num_threads, _ThreadIndex __thread_no)`

### 6.129.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [equally\\_split.h](#).

## 6.130 `error_constants.h` File Reference

### Namespaces

- namespace `std`

### Enumerations

- `enum errc {  
    address_family_not_supported,   address_in_use,   address_not_available,  
    already_connected,  
    argument_list_too_long,   argument_out_of_domain,   bad_address,   bad_-  
    file_descriptor,  
    bad_message,   broken_pipe,   connection_aborted,   connection_already_in_-  
    progress,  
    connection_refused,   connection_reset,   cross_device_link,   destination_-  
    address_required,  
    device_or_resource_busy,   directory_not_empty,   executable_format_error,  
    file_exists,`

```

file_too_large, filename_too_long, function_not_supported, host_ -
unreachable,

identifier_removed, illegal_byte_sequence, inappropriate_io_control_ -
operation, interrupted,

invalid_argument, invalid_seek, io_error, is_a_directory,

message_size, network_down, network_reset, network_unreachable,

no_buffer_space, no_child_process, no_link, no_lock_available,

no_message_available, no_message, no_protocol_option, no_space_on_ -
device,

no_stream_resources, no_such_device_or_address, no_such_device, no_ -
such_file_or_directory,

no_such_process, not_a_directory, not_a_socket, not_a_stream,

not_connected, not_enough_memory, not_supported, operation_canceled,

operation_in_progress, operation_not_permitted, operation_not_ -
supported, operation_would_block,

owner_dead, permission_denied, protocol_error, protocol_not_supported,

read_only_file_system, resource_deadlock_would_occur, resource_ -
unavailable_try_again, result_out_of_range,

state_not_recoverable, stream_timeout, text_file_busy, timed_out,

too_many_files_open_in_system, too_many_files_open, too_many_links,
too_many_symbolic_link_levels,

value_too_large, wrong_protocol_type }

```

### 6.130.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [error\\_constants.h](#).

## 6.131 exception File Reference

### Classes

- class [std::bad\\_exception](#)
- class [std::exception](#)

*Base class for all library exceptions.*



## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Typedefs

- typedef void(\* [std::terminate\\_handler](#) )()
- typedef void(\* [std::unexpected\\_handler](#) )()

## Functions

- void [\\_\\_gnu\\_cxx::\\_\\_verbose\\_terminate\\_handler](#) ()
- terminate\_handler [std::set\\_terminate](#) (terminate\_handler) throw ()
- unexpected\_handler [std::set\\_unexpected](#) (unexpected\_handler) throw ()
- void [std::terminate](#) () \_\_attribute\_\_((\_\_noreturn\_\_)) throw ()
- bool [std::uncaught\\_exception](#) () \_\_attribute\_\_((\_\_pure\_\_)) throw ()
- void [std::unexpected](#) () \_\_attribute\_\_((\_\_noreturn\_\_))

### 6.131.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [exception](#).

## 6.132 exception.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Functions

- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_container\\_error](#) (void)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_insert\\_error](#) (void)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_join\\_error](#) (void)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_resize\\_error](#) (void)

### 6.132.1 Detailed Description

Contains exception classes.

Definition in file [exception.hpp](#).

## 6.133 exception\_ptr.h File Reference

### Classes

- class [std::\\_\\_exception\\_ptr::exception\\_ptr](#)  
*An opaque pointer to an arbitrary exception.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Ex >  
exception_ptr std::copy\_exception (_Ex __ex) throw ()`
- `exception_ptr std::current\_exception () throw ()`
- `template<typename _Ex >  
exception_ptr std::make\_exception\_ptr (_Ex __ex) throw ()`
- `bool std::\_\_exception\_ptr::operator!= (const exception_ptr &, const  
exception_ptr &) __attribute__((__pure__)) throw ()`
- `bool std::\_\_exception\_ptr::operator== (const exception_ptr &, const  
exception_ptr &) __attribute__((__pure__)) throw ()`
- `void std::rethrow\_exception (exception_ptr) __attribute__((__noreturn__))`
- `void std::\_\_exception\_ptr::swap (exception_ptr &__lhs, exception_ptr &__-  
rhs)`

### 6.133.1 Detailed Description

This is an internal header file, included by other headers and the implementation. You should not attempt to use it directly.

Definition in file [exception\\_ptr.h](#).

## 6.134 extptr\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_ExtPtr\\_allocator< \\_Tp >](#)  
*An example allocator which uses a non-standard pointer type.  
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See [ext/pointer.h](#)) Memory allocation in this example is still performed using [std::allocator](#).*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp >  
void \_\_gnu\_cxx::swap (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)`

#### 6.134.1 Detailed Description

##### Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

Definition in file [extptr\\_allocator.h](#).

## 6.135 features.h File Reference

Defines on whether to include algorithm variants.

### Defines

- `#define \_GLIBCXX\_BAL\_QUICKSORT`
- `#define \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS`
- `#define \_GLIBCXX\_FIND\_EQUAL\_SPLIT`
- `#define \_GLIBCXX\_FIND\_GROWING\_BLOCKS`

- `#define _GLIBCXX_MERGESORT`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

### 6.135.1 Detailed Description

Defines on whether to include algorithm variants. Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [features.h](#).

### 6.135.2 Define Documentation

#### 6.135.2.1 `#define _GLIBCXX_BAL_QUICKSORT`

Include parallel dynamically load-balanced quicksort.

**See also**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 55 of file [features.h](#).

#### 6.135.2.2 `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

Include the equal-sized blocks variant for `std::find`.

**See also**

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 67 of file [features.h](#).

#### 6.135.2.3 `#define _GLIBCXX_FIND_EQUAL_SPLIT`

Include the equal splitting variant for `std::find`.

**See also**

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file [features.h](#).

**6.135.2.4 #define \_GLIBCXX\_FIND\_GROWING\_BLOCKS**

Include the growing blocks variant for `std::find`.

**See also**

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file `features.h`.

**6.135.2.5 #define \_GLIBCXX\_MERGESORT**

Include parallel multi-way mergesort.

**See also**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 41 of file `features.h`.

**6.135.2.6 #define \_GLIBCXX\_QUICKSORT**

Include parallel unbalanced quicksort.

**See also**

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 48 of file `features.h`.

**6.135.2.7 #define \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING**

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

**See also**

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file `features.h`.

#### 6.135.2.8 `#define _GLIBCXX_TREE_FULL_COPY`

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

##### See also

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file `features.h`.

#### 6.135.2.9 `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

##### See also

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file `features.h`.

## 6.136 `fnv.h` File Reference

### 6.136.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fnv.h](#).

## 6.137 `find.h` File Reference

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1`  
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __-`  
`selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1`  
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __-`  
`selector, constant_size_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1`  
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __-`  
`selector, growing_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1`  
`__begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __-`  
`selector, equal_split_tag)`

### 6.137.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find.h](#).

## 6.138 find\_selectors.h File Reference

`_Function` objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

## Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#)  
*Test predicate on two adjacent elements.*
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector< \\_FIterator >](#)  
*Test predicate on several elements.*
- struct [\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#)  
*Test predicate on a single element, used for `std::find()` and `std::find_if()`.*

- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#)  
*Base class of all [\\_\\_gnu\\_parallel::\\_\\_find\\_template](#) selectors.*
- struct [\\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#)  
*Test inverted predicate on a single element.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.138.1 Detailed Description

[\\_Function](#) objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [find\\_selectors.h](#).

## 6.139 [for\\_each.h](#) File Reference

Main interface for embarrassingly parallel functions.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- template<typename [\\_Iter](#) , typename [\\_UserOp](#) , typename [\\_Functionality](#) , typename [\\_Red](#) , typename [\\_Result](#) >  
[\\_UserOp](#) [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access](#) ([\\_Iter](#) \_\_begin, [\\_Iter](#) \_\_end, [\\_UserOp](#) \_\_user\_op, [\\_Functionality](#) &\_\_functionality, [\\_Red](#) \_\_reduction, [\\_Result](#) \_\_reduction\_start, [\\_Result](#) &\_\_output, typename [std::iterator\\_traits< \\_Iter >::difference\\_type](#) \_\_bound, [\\_Parallelism](#) \_\_parallelism\_tag)

### 6.139.1 Detailed Description

Main interface for embarrassingly parallel functions. The explicit implementation are in other header files, like [workstealing.h](#), [par\\_loop.h](#), [omp\\_loop.h](#), and [omp\\_loop\\_static.h](#). This file is a GNU parallel extension to the Standard C++ Library.



Definition in file [for\\_each.h](#).

## 6.140 for\_each\_selectors.h File Reference

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_binop\\_reduct< \\_BinOp >](#)  
*General reduction, using a binary operator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_accumulate\\_selector< \\_It >](#)  
*std::accumulate() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_adjacent\\_difference\\_selector< \\_It >](#)  
*Selector that returns the difference between two adjacent \_\_elements.*
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_if\\_selector< \\_It, \\_Diff >](#)  
*std::count\_if() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_count\\_selector< \\_It, \\_Diff >](#)  
*std::count() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_fill\\_selector< \\_It >](#)  
*std::fill() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_selector< \\_It >](#)  
*std::for\_each() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_generate\\_selector< \\_It >](#)  
*std::generate() selector.*
- struct [\\_\\_gnu\\_parallel::\\_\\_generic\\_for\\_each\\_selector< \\_It >](#)  
*Generic \_\_selector for embarrassingly parallel functions.*
- struct [\\_\\_gnu\\_parallel::\\_\\_identity\\_selector< \\_It >](#)  
*Selector that just returns the passed iterator.*
- struct [\\_\\_gnu\\_parallel::\\_\\_inner\\_product\\_selector< \\_It, \\_It2, \\_Tp >](#)

[`std::inner\_product\(\)`](#) selector.

- struct [`\_\_gnu\_parallel::\_\_max\_element\_reduct<\_Compare, \_It>`](#)

*Reduction for finding the maximum element, using a comparator.*

- struct [`\_\_gnu\_parallel::\_\_min\_element\_reduct<\_Compare, \_It>`](#)

*Reduction for finding the maximum element, using a comparator.*

- struct [`\_\_gnu\_parallel::\_\_replace\_if\_selector<\_It, \_Op, \_Tp>`](#)

*`std::replace()` selector.*

- struct [`\_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>`](#)

*`std::replace()` selector.*

- struct [`\_\_gnu\_parallel::\_\_transform1\_selector<\_It>`](#)

*`std::transform()` \_\_selector, one input sequence variant.*

- struct [`\_\_gnu\_parallel::\_\_transform2\_selector<\_It>`](#)

*`std::transform()` \_\_selector, two input sequences variant.*

- struct [`\_\_gnu\_parallel::\_\_DummyReduct`](#)

*Reduction function doing nothing.*

- struct [`\_\_gnu\_parallel::\_\_Nothing`](#)

*Functor doing nothing.*

## Namespaces

- namespace [`\_\_gnu\_parallel`](#)

### 6.140.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [for\\_each\\_selectors.h](#).

## 6.141 formatter.h File Reference

### Classes

- struct `__gnu_debug::__is_same<_Type1, _Type2>`

### Namespaces

- namespace `__gnu_debug`

### Enumerations

- enum `_Debug_msg_id` {  
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`,  
`__msg_bad_bitset_flip`,  
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,  
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,  
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,  
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,  
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`,  
`__msg_distance_bad`,  
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,  
`__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`,  
`__msg_erase_after_bad`,  
`__msg_valid_range2` }

### Functions

- template<typename `_Iterator`>  
bool `__gnu_debug::__check_singular` (`_Iterator` &)

### 6.141.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [formatter.h](#).

## 6.142 `forward_list.h` File Reference

### Classes

- struct [std::\\_Fwd\\_list\\_base< \\_Tp, \\_Alloc >](#)  
*Base class for forward\_list.*
- struct [std::\\_Fwd\\_list\\_const\\_iterator< \\_Tp >](#)  
*A forward\_list::const\_iterator.*
- struct [std::\\_Fwd\\_list\\_iterator< \\_Tp >](#)  
*A forward\_list::iterator.*
- struct [std::\\_Fwd\\_list\\_node< \\_Tp >](#)  
*A helper node class for forward\_list. This is just a linked list with a data value in each node. There is a sorting utility method.*
- struct [std::\\_Fwd\\_list\\_node\\_base](#)  
*A helper basic node class for forward\_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.*
- class [std::forward\\_list< \\_Tp, \\_Alloc >](#)  
*A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.*

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Tp >  
bool [std::operator!=](#) (const \_Fwd\_list\_iterator< \_Tp > &\_\_x, const \_Fwd\_list\_iterator< \_Tp > &\_\_y)

- template<typename \_Tp, typename \_Alloc >  
bool [std::operator!=](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::operator<](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::operator<=](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp >  
bool [std::operator==](#) (const Fwd\_list\_iterator< \_Tp > &\_\_x, const Fwd\_list\_iterator< \_Tp > &\_\_y)
- template<typename \_Tp, typename \_Alloc >  
bool [std::operator==](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::operator>](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
bool [std::operator>=](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)
- template<typename \_Tp, typename \_Alloc >  
void [std::swap](#) (forward\_list< \_Tp, \_Alloc > &\_\_lx, forward\_list< \_Tp, \_Alloc > &\_\_ly)

### 6.142.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward\\_list.h](#).

## 6.143 forward\_list.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- template<typename \_Tp, typename \_Alloc >  
bool [std::operator==](#) (const forward\_list< \_Tp, \_Alloc > &\_\_lx, const forward\_list< \_Tp, \_Alloc > &\_\_ly)

### 6.143.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [forward\\_list.tcc](#).

## 6.144 fstream File Reference

### Classes

- class [std::basic\\_filebuf<\\_CharT, \\_Traits>](#)  
*The actual work of input and output (for files).  
This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's FILE streams.*
- class [std::basic\\_fstream<\\_CharT, \\_Traits>](#)  
*Controlling input and output for files.  
This class supports reading from and writing to named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_ifstream<\\_CharT, \\_Traits>](#)  
*Controlling input for files.  
This class supports reading from named files, using the inherited functions from [std::basic\\_istream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*
- class [std::basic\\_ofstream<\\_CharT, \\_Traits>](#)  
*Controlling output for files.  
This class supports reading from named files, using the inherited functions from [std::basic\\_ostream](#). To control the associated sequence, an instance of [std::basic\\_filebuf](#) is used, which this page refers to as *sb*.*

### Namespaces

- namespace [std](#)

### 6.144.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [fstream](#).

## 6.145 fstream.tcc File Reference

### Namespaces

- namespace [std](#)

### 6.145.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [fstream.tcc](#).

## 6.146 functexcept.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- void **std::\_\_throw\_bad\_alloc** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_cast** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_exception** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_function\_call** () \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_bad\_typeid** (void) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_domain\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_future\_error** (int) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_invalid\_argument** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_ios\_failure** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_length\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_logic\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_out\_of\_range** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_overflow\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_-  
))
- void **std::\_\_throw\_range\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_runtime\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_system\_error** (int) \_\_attribute\_\_((\_\_noreturn\_\_))
- void **std::\_\_throw\_underflow\_error** (const char \*) \_\_attribute\_\_((\_\_noreturn\_\_-  
\_))

### 6.146.1 Detailed Description

This header provides support for `-fno-exceptions`.

Definition in file [functexcept.h](#).

## 6.147 functional File Reference

### Classes

- struct [std::\\_\\_is\\_location\\_invariant< \\_Tp >](#)
- struct [std::\\_\\_Derives\\_from\\_binary\\_function< \\_Tp >](#)  
*Determines if the type `_Tp` derives from `binary_function`.*
- struct [std::\\_\\_Derives\\_from\\_unary\\_function< \\_Tp >](#)  
*Determines if the type `_Tp` derives from `unary_function`.*
- class [std::\\_\\_Function\\_base](#)  
*Base class of all polymorphic function object wrappers.*
- struct [std::\\_\\_Function\\_to\\_function\\_pointer< \\_Tp, \\_IsFunctionType >](#)  
*Turns a function type into a function pointer type.*
- struct [std::\\_\\_Maybe\\_get\\_result\\_type< \\_Has\\_result\\_type, \\_Functor >](#)  
*If we have found a `result_type`, extract it.*
- struct [std::\\_\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_ArgTypes >](#)
- struct [std::\\_\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1 >](#)  
*Derives from `unary_function`, as appropriate.*
- struct [std::\\_\\_Maybe\\_unary\\_or\\_binary\\_function< \\_Res, \\_T1, \\_T2 >](#)  
*Derives from `binary_function`, as appropriate.*
- struct [std::\\_\\_Maybe\\_wrap\\_member\\_pointer< \\_Tp >](#)
- struct [std::\\_\\_Maybe\\_wrap\\_member\\_pointer< \\_Tp \\_Class::\\* >](#)
- class [std::\\_\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const >](#)  
*Implementation of `mem_fn` for const member function pointers.*
- class [std::\\_\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) const volatile >](#)  
*Implementation of `mem_fn` for const volatile member function pointers.*
- class [std::\\_\\_Mem\\_fn< \\_Res\(\\_Class::\\*\)\(\\_ArgTypes...\) volatile >](#)



*Implementation of `mem_fn` for volatile member function pointers.*

- class `std::_Mem_fn<_Res(_Class::*)(_ArgTypes...)>`

*Implementation of `mem_fn` for member function pointers.*

- class `std::_Mu<_Arg, false, false >`
- class `std::_Mu<_Arg, false, true >`
- class `std::_Mu<_Arg, true, false >`
- class `std::_Mu<reference_wrapper<_Tp>, false, false >`
- struct `std::_Placeholder<_Num >`

*The type of placeholder objects defined by `libstdc++`.*

- struct `std::_Reference_wrapper_base<_Tp >`
- struct `std::_Safe_tuple_element<__i, _Tuple >`
- struct `std::_Safe_tuple_element_impl<__i, _Tuple, _IsSafe >`
- struct `std::_Safe_tuple_element_impl<__i, _Tuple, false >`
- struct `std::_Weak_result_type<_Functor >`
- struct `std::_Weak_result_type_impl<_Functor >`
- struct `std::_Weak_result_type_impl<_Res(&)(_ArgTypes...)>`

*Retrieve the result type for a function reference.*

- struct `std::_Weak_result_type_impl<_Res(*)(_ArgTypes...)>`

*Retrieve the result type for a function pointer.*

- struct `std::_Weak_result_type_impl<_Res(_ArgTypes...)>`

*Retrieve the result type for a function type.*

- struct `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const >`

*Retrieve result type for a const member function pointer.*

- struct `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) const volatile >`

*Retrieve result type for a const volatile member function pointer.*

- struct `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...) volatile >`

*Retrieve result type for a volatile member function pointer.*

- struct `std::_Weak_result_type_impl<_Res(_Class::*)(_ArgTypes...)>`

*Retrieve result type for a member function pointer.*

- class `std::bad_function_call`

Exception class thrown when class template function's operator() is called with an empty target.

- class [std::function< \\_Res\(\\_ArgTypes...\)>](#)  
Primary class template for `std::function`.  
Polymorphic function wrapper.
- struct [std::is\\_bind\\_expression< \\_Tp >](#)  
Determines if the given type `_Tp` is a function object should be treated as a subexpression when evaluating calls to function objects returned by [bind\(\)](#). [TR1 3.6.1].
- struct [std::is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)  
Class template `_Bind` is always a bind expression.
- struct [std::is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)  
Class template `_Bind` is always a bind expression.
- struct [std::is\\_placeholder< \\_Tp >](#)  
Determines if the given type `_Tp` is a placeholder in a [bind\(\)](#) expression and, if so, which placeholder it is. [TR1 3.6.2].
- struct [std::is\\_placeholder< \\_Placeholder< \\_Num > >](#)
- class [std::reference\\_wrapper< \\_Tp >](#)  
Primary class template for `reference_wrapper`.

## Namespaces

- namespace [std](#)
- namespace [std::placeholders](#)

## Enumerations

- enum `_Manager_operation` { `__get_type_info`, `__get_functor_ptr`, `__clone_functor`, `__destroy_functor` }

## Functions

- template<typename `_Functor` >  
`_Functor & std::__callable_functor (_Functor &__f)`
- template<typename `_Member` , typename `_Class` >  
`_Mem_fn< _Member _Class::* > std::__callable_functor (_Member _Class::*const &__p)`

- `template<typename _Member, typename _Class>  
_Mem_fn< _Member _Class::*> std::__callable_function (_Member _-  
Class::*&__p)`
  - `template<size_t _Ind, typename... _Tp>  
auto std::__volget (const volatile tuple< _Tp...> &__tuple)-> typename  
tuple_element< _Ind`
  - `template<size_t _Ind, typename... _Tp>  
auto std::__volget (volatile tuple< _Tp...> &__tuple)-> typename tuple_  
element< _Ind`
  - `template<typename _Functor, typename... _ArgTypes>  
_Bind_helper< _Functor, _ArgTypes...>::type std::bind (_Functor &&__f, _-  
ArgTypes &&...__args)`
  - `template<typename _Result, typename _Functor, typename... _ArgTypes>  
_Bindres_helper< _Result, _Functor, _ArgTypes...>::type std::bind (_Functor  
&&__f, _ArgTypes &&...__args)`
  - `template<typename _Tp, typename _Class>  
_Mem_fn< _Tp _Class::*> std::mem_fn (_Tp _Class::*__pm)`
  - `template<typename _Res, typename... _Args>  
bool std::operator!= (const function< _Res(_Args...)> &__f, nullptr_t)`
  - `template<typename _Res, typename... _Args>  
bool std::operator!= (nullptr_t, const function< _Res(_Args...)> &__f)`
  - `template<typename _Res, typename... _Args>  
bool std::operator== (nullptr_t, const function< _Res(_Args...)> &__f)`
  - `template<typename _Res, typename... _Args>  
bool std::operator== (const function< _Res(_Args...)> &__f, nullptr_t)`
  - `template<typename _Res, typename... _Args>  
void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)>  
&__y)`
- 
- `template<typename _Tp>  
reference_wrapper< _Tp> std::ref (_Tp &__t)`
  - `template<typename _Tp>  
reference_wrapper< const _Tp> std::cref (const _Tp &__t)`
  - `template<typename _Tp>  
reference_wrapper< _Tp> std::ref (reference_wrapper< _Tp> __t)`
  - `template<typename _Tp>  
reference_wrapper< const _Tp> std::cref (reference_wrapper< _Tp> __t)`

## Variables

- `enable_if< (!is_member_pointer< _Functor>::value &&!is_function< _-  
Functor>::value &&!is_function< typename remove_pointer< _Functor  
>::type>::value), typename result_of< _Functor(_Args...)>::type>::type  
std::__invoke (_Functor &__f, _Args &&...__args)`

### 6.147.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [functional](#).

## 6.148 functional File Reference

### Classes

- class [\\_\\_gnu\\_cxx::binary\\_compose< \\_Operation1, \\_Operation2, \\_Operation3 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_binary\\_fun< \\_Result, \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_unary\\_fun< \\_Result, \\_Argument >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::constant\\_void\\_fun< \\_Result >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::project1st< \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::project2nd< \\_Arg1, \\_Arg2 >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::select1st< \\_Pair >](#)  
*An SGI extension .*
- struct [\\_\\_gnu\\_cxx::select2nd< \\_Pair >](#)  
*An SGI extension .*
- class [\\_\\_gnu\\_cxx::subtractive\\_rng](#)
- class [\\_\\_gnu\\_cxx::unary\\_compose< \\_Operation1, \\_Operation2 >](#)  
*An SGI extension .*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Functions

- `template<class _Operation1 , class _Operation2 >`  
`unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (const _-`  
`Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1 , class _Operation2 , class _Operation3 >`  
`binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_-`  
`cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const`  
`_Operation3 &__fn3)`
- `template<class _Result >`  
`constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result`  
`&__val)`
- `template<class _Result >`  
`constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const`  
`_Result &__val)`
- `template<class _Tp >`  
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
- `template<class _Tp >`  
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
- `template<class _Ret , class _Tp , class _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > __gnu_cxx::mem_fun1 (_Ret(_Tp::*__f)(_-`  
`Arg))`
- `template<class _Ret , class _Tp , class _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > __gnu_cxx::mem_fun1_ref (_Ret(_-`  
`Tp::*__f)(_Arg))`

### 6.148.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/functional](#).

## 6.149 functional\_hash.h File Reference

### Classes

- `struct std::hash< _Tp >`  
*Primary class template hash.*
- `struct std::hash< _Tp * >`

*Partial specializations for pointer types.*

## Namespaces

- namespace [std](#)

## Defines

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

## Functions

- `size_t std::Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t std::Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`

### 6.149.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [functional\\_hash.h](#).

## 6.150 functions.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Functions

- `template<typename _Iterator >  
bool __gnu_debug::__check_dereferenceable (_Iterator &)`
- `template<typename _Tp >  
bool __gnu_debug::__check_dereferenceable (const _Tp *__ptr)`
- `template<typename _Iterator, typename _Sequence >  
bool __gnu_debug::__check_dereferenceable (const _Safe_iterator< _Iterator,  
_Sequence > &__x)`
- `template<typename _ForwardIterator, typename _Tp >  
bool __gnu_debug::__check_partitioned_lower (_ForwardIterator __first, _-  
ForwardIterator __last, const _Tp &__value)`

- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool __gnu_debug::__check_partitioned_lower (_ForwardIterator __first, _-`  
`ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`bool __gnu_debug::__check_partitioned_upper (_ForwardIterator __first, _-`  
`ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`bool __gnu_debug::__check_partitioned_upper (_ForwardIterator __first, _-`  
`ForwardIterator __last, const _Tp &__value)`
- `template<typename _Iterator >`  
`bool __gnu_debug::__check_singular (_Iterator &)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::__check_singular (const _Safe_iterator< _Iterator, _-`  
`Sequence > &__x)`
- `template<typename _Tp >`  
`bool __gnu_debug::__check_singular (const _Tp *__ptr)`
- `bool __gnu_debug::__check_singular_aux (const void *)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _-`  
`InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _-`  
`InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _-`  
`InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _-`  
`ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _-`  
`InputIterator &, _Predicate, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _-`  
`ForwardIterator __last, _Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const`  
`_InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const`  
`_InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first,`  
`const _InputIterator &__last, _Predicate __pred, std::\_\_true\_type)`

- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first,`  
`const _InputIterator &__last, std::__true_type)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _-`  
`InputIterator &, std::__false_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _-`  
`InputIterator &, _Predicate, std::__false_type)`
- `template<typename _CharT >`  
`const _CharT * __gnu_debug::__check_string (const _CharT *__s)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * __gnu_debug::__check_string (const _CharT *__s, const _-`  
`Integer &__n __attribute__((__unused__)))`
- `template<typename _InputIterator >`  
`_InputIterator __gnu_debug::__check_valid_range (const _InputIterator &__-`  
`first, const _InputIterator &__last __attribute__((__unused__)))`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence`  
`> &__first, const _Safe_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__valid_range (const _InputIterator &__first, const _-`  
`InputIterator &__last)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__valid_range_aux (const _InputIterator &__first, const _-`  
`InputIterator &__last, std::__false_type)`
- `template<typename _Integral >`  
`bool __gnu_debug::__valid_range_aux (const _Integral &, const _Integral &,`  
`std::__true_type)`
- `template<typename _InputIterator >`  
`bool __gnu_debug::__valid_range_aux2 (const _InputIterator &, const _-`  
`InputIterator &, std::input_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`bool __gnu_debug::__valid_range_aux2 (const _RandomAccessIterator &__-`  
`first, const _RandomAccessIterator &__last, std::random_access_iterator_tag)`

### 6.150.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [functions.h](#).



## 6.151 future File Reference

### Classes

- class `std::__basic_future<_Res>`  
*Common implementation for future and `shared_future`.*
- struct `std::__future_base`  
*Base class and enclosing scope.*
- struct `std::__future_base::Ptr<_Res>`  
*A `unique_ptr` based on the instantiating type.*
- struct `std::__future_base::Result<_Res>`  
*Result.*
- struct `std::__future_base::Result<_Res &>`  
*Partial specialization for reference types.*
- struct `std::__future_base::Result<void>`  
*Explicit specialization for void.*
- struct `std::__future_base::Result_alloc<_Res, _Alloc>`  
*Result\_alloc.*
- struct `std::__future_base::Result_base`  
*Base class for results.*
- class `std::__future_base::State`  
*Shared state between a promise and one or more associated futures.*
- class `std::future<_Res>`  
*Primary template for future.*
- class `std::future<_Res &>`  
*Partial specialization for `future<R&>`*
- class `std::future<void>`  
*Explicit specialization for `future<void>`*
- class `std::future_error`  
*Exception type thrown by futures.*

- class `std::packaged_task< _Res(_ArgTypes...)>`  
*packaged\_task*
- class `std::promise< _Res >`  
*Primary template for promise.*
- class `std::promise< _Res & >`  
*Partial specialization for promise<R&>*
- class `std::promise< void >`  
*Explicit specialization for promise<void>*
- class `std::shared_future< _Res >`  
*Primary template for shared\_future.*
- class `std::shared_future< _Res & >`  
*Partial specialization for shared\_future<R&>*
- class `std::shared_future< void >`  
*Explicit specialization for shared\_future<void>*

## Namespaces

- namespace `std`

## Enumerations

- enum `std::future_errc` { `broken_promise`, `future_already_retrieved`, `promise_already_satisfied`, `no_state` }
- enum `launch` { `any`, `async`, `sync` }

## Functions

- template<typename \_Fn, typename... \_Args>  
future< typename result\_of< \_Fn(\_Args...)>::type > **std::async** (launch \_\_-  
policy, \_Fn &&\_\_fn, \_Args &&...\_\_args)
- template<typename \_Fn, typename... \_Args>  
enable\_if<!is\_same< typename decay< \_Fn >::type, launch >::value, future<  
decltype(std::declval< \_Fn >)(std::declval< \_Args >)...)> >::type **std::async**  
(\_Fn &&\_\_fn, \_Args &&...\_\_args)

- error\_code **std::make\_error\_code** (future\_errc \_\_errc)
- error\_condition **std::make\_error\_condition** (future\_errc \_\_errc)
- template<typename \_Res >  
void **std::swap** (promise< \_Res > &\_\_x, promise< \_Res > &\_\_y)
- template<typename \_Res, typename... \_ArgTypes>  
void **std::swap** (packaged\_task< \_Res(\_ArgTypes...)> &\_\_x, packaged\_task< \_Res(\_ArgTypes...)> &\_\_y)

## Variables

- const error\_category \*const [std::future\\_category](#)

### 6.151.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [future](#).

## 6.152 `gslice.h` File Reference

### Classes

- class [std::gslice](#)  
*Class defining multi-dimensional subset of an array.*

### Namespaces

- namespace [std](#)

### 6.152.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [gslice.h](#).

## 6.153 `gslice_array.h` File Reference

### Classes

- class [std::gslice\\_array< \\_Tp >](#)

*Reference to multi-dimensional subset of an array.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

#### 6.153.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [gslice\\_array.h](#).

## 6.154 `hash_fun.h` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `size_t __gnu_cxx::__stl_hash_string (const char *__s)`

#### 6.154.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_fun.h](#).

## 6.155 hash\_map File Reference

### Classes

- class [\\_\\_gnu\\_cxx::hash\\_map<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::hash\\_multimap<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const hash\_map<\_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc> &\_\_hm1, const hash\_map<\_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc> &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HF, class \_EqKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator!=** (const hash\_multimap<\_Key, \_Tp, \_HF, \_EqKey, \_Alloc> &\_\_hm1, const hash\_multimap<\_Key, \_Tp, \_HF, \_EqKey, \_Alloc> &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator==** (const hash\_map<\_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc> &\_\_hm1, const hash\_map<\_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc> &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HF, class \_EqKey, class \_Alloc>  
bool **\_\_gnu\_cxx::operator==** (const hash\_multimap<\_Key, \_Tp, \_HF, \_EqKey, \_Alloc> &\_\_hm1, const hash\_multimap<\_Key, \_Tp, \_HF, \_EqKey, \_Alloc> &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc>  
void **\_\_gnu\_cxx::swap** (hash\_map<\_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc> &\_\_hm1, hash\_map<\_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc> &\_\_hm2)
- template<class \_Key, class \_Tp, class \_HashFn, class \_EqKey, class \_Alloc>  
void **\_\_gnu\_cxx::swap** (hash\_multimap<\_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc> &\_\_hm1, hash\_multimap<\_Key, \_Tp, \_HashFn, \_EqKey, \_Alloc> &\_\_hm2)

#### 6.155.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_map](#).

## 6.156 hash\_policy.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_C\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_CLASS\_T\_DEC**
- #define **PB\_DS\_SIZE\_BASE\_C\_DEC**

### 6.156.1 Detailed Description

Contains hash-related policies.

Definition in file [hash\\_policy.hpp](#).

## 6.157 hash\_set File Reference

### Classes

- class [\\_\\_gnu\\_cxx::hash\\_multiset<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc >](#)
- class [\\_\\_gnu\\_cxx::hash\\_set<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc >](#)

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool \_\_gnu\_cxx::operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool \_\_gnu\_cxx::operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool \_\_gnu\_cxx::operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`bool \_\_gnu\_cxx::operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void \_\_gnu\_cxx::swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void \_\_gnu\_cxx::swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`

### 6.157.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [hash\\_set](#).

## 6.158 hashtable.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

## Enumerations

- enum { **\_S\_num\_primes** }

## Functions

- unsigned long **\_\_gnu\_cxx::\_\_stl\_next\_prime** (unsigned long \_\_n)
- template<class \_Val, class \_Key, class \_HF, class \_Ex, class \_Eq, class \_All >  
bool **\_\_gnu\_cxx::operator!=** (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- template<class \_Val, class \_Key, class \_HF, class \_Ex, class \_Eq, class \_All >  
bool **\_\_gnu\_cxx::operator==** (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- template<class \_Val, class \_Key, class \_HF, class \_Extract, class \_EqKey, class \_All >  
void **\_\_gnu\_cxx::swap** (hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht1, hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht2)

## Variables

- static const unsigned long **\_\_gnu\_cxx::\_\_stl\_prime\_list** [\_S\_num\_primes]

### 6.158.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [backward/hashtable.h](#).

## 6.159 hashtable.h File Reference

### Namespaces

- namespace [std](#)

### 6.159.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [bits/hashtable.h](#).



## 6.160 hashtable\_policy.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<class _Iterator >  
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw  
( _Iterator __first, _Iterator __last, std::input\_iterator\_tag)`
- `template<class _Iterator >  
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw  
( _Iterator __first, _Iterator __last, std::forward\_iterator\_tag)`
- `template<class _Iterator >  
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw  
( _Iterator __first, _Iterator __last)`
- `template<typename _Value , bool __cache>  
bool std::__detail::operator!= (const _Node_iterator_base< _Value, __cache  
> &__x, const _Node_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>  
bool std::__detail::operator!= (const _Hashtable_iterator_base< _Value, __-  
cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>  
bool std::__detail::operator== (const _Hashtable_iterator_base< _Value, __-  
cache > &__x, const _Hashtable_iterator_base< _Value, __cache > &__y)`
- `template<typename _Value , bool __cache>  
bool std::__detail::operator== (const _Node_iterator_base< _Value, __cache  
> &__x, const _Node_iterator_base< _Value, __cache > &__y)`

### Variables

- `const unsigned long std::__detail::__prime_list []`

#### 6.160.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [hashtable\\_policy.h](#).

## 6.161 `indirect_array.h` File Reference

### Classes

- class `std::indirect_array<_Tp>`  
*Reference to arbitrary subset of an array.*

### Namespaces

- namespace `std`

### Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

#### 6.161.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `indirect_array.h`.

## 6.162 `initializer_list` File Reference

### Classes

- class `std::initializer_list<_E>`  
*`initializer_list`*

### Namespaces

- namespace `std`

### Functions

- `template<class _Tp>`  
`const _Tp * std::begin (initializer_list<_Tp> __ils)`
- `template<class _Tp>`  
`const _Tp * std::end (initializer_list<_Tp> __ils)`

### 6.162.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [initializer\\_list](#).

## 6.163 iomanip File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _MoneyT >  
_Get_money< _MoneyT > std::get\_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision __f)`
- `template<typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)`
- `template<typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)`

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Get_money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__is, _Setprecision __f)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > std::put_money (const _MoneyT &__mon, bool __-`  
`intl=false)`
- `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
- `_Setbase std::setbase (int __base)`
- `template<typename _CharT >`  
`_Setfill< _CharT > std::setfill (_CharT __c)`
- `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision std::setprecision (int __n)`
- `_Setw std::setw (int __n)`

### 6.163.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iomanip](#).

## 6.164 ios File Reference

### 6.164.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ios](#).

## 6.165 ios\_base.h File Reference

### Classes

- class [std::ios\\_base](#)  
*The base of the I/O class hierarchy.  
This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see [ios\\_base](#) when they need to specify the full name of the various I/O flags (e.g., the openmodes).*
- class [std::ios\\_base::failure](#)  
*These are thrown to indicate problems with io.  
27.4.2.1.1 Class [ios\\_base::failure](#).*

### Namespaces

- namespace [std](#)

### Enumerations

- enum [\\_Ios\\_Fmtflags](#) {  
    [\\_S\\_boolalpha](#), [\\_S\\_dec](#), [\\_S\\_fixed](#), [\\_S\\_hex](#),  
    [\\_S\\_internal](#), [\\_S\\_left](#), [\\_S\\_oct](#), [\\_S\\_right](#),  
    [\\_S\\_scientific](#), [\\_S\\_showbase](#), [\\_S\\_showpoint](#), [\\_S\\_showpos](#),  
    [\\_S\\_skipws](#), [\\_S\\_unitbuf](#), [\\_S\\_uppercase](#), [\\_S\\_adjustfield](#),  
    [\\_S\\_basefield](#), [\\_S\\_floatfield](#), [\\_S\\_ios\\_fmtflags\\_end](#) }
- enum [\\_Ios\\_Iostate](#) {  
    [\\_S\\_goodbit](#), [\\_S\\_badbit](#), [\\_S\\_eofbit](#), [\\_S\\_failbit](#),  
    [\\_S\\_ios\\_iostate\\_end](#) }
- enum [\\_Ios\\_Openmode](#) {  
    [\\_S\\_app](#), [\\_S\\_ate](#), [\\_S\\_bin](#), [\\_S\\_in](#),  
    [\\_S\\_out](#), [\\_S\\_trunc](#), [\\_S\\_ios\\_openmode\\_end](#) }
- enum [\\_Ios\\_Seekdir](#) { [\\_S\\_beg](#), [\\_S\\_cur](#), [\\_S\\_end](#), [\\_S\\_ios\\_seekdir\\_end](#) }

### Functions

- [ios\\_base & std::boolalpha](#) ([ios\\_base & \\_\\_base](#))
- [ios\\_base & std::dec](#) ([ios\\_base & \\_\\_base](#))
- [ios\\_base & std::fixed](#) ([ios\\_base & \\_\\_base](#))

- `ios_base & std::hex (ios_base &__base)`
- `ios_base & std::internal (ios_base &__base)`
- `ios_base & std::left (ios_base &__base)`
- `ios_base & std::noboolalpha (ios_base &__base)`
- `ios_base & std::noshowbase (ios_base &__base)`
- `ios_base & std::noshowpoint (ios_base &__base)`
- `ios_base & std::noshowpos (ios_base &__base)`
- `ios_base & std::noskipws (ios_base &__base)`
- `ios_base & std::nounitbuf (ios_base &__base)`
- `ios_base & std::nouppercase (ios_base &__base)`
- `ios_base & std::oct (ios_base &__base)`
- `_Ios_Fmtflags std::operator& (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `_Ios_Openmode std::operator& (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Iostate std::operator& (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Fmtflags & std::operator&= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `_Ios_Iostate & std::operator&= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Openmode & std::operator&= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate std::operator^ (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Openmode std::operator^ (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Fmtflags std::operator^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `_Ios_Iostate & std::operator^= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Fmtflags & std::operator^= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `_Ios_Openmode & std::operator^= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate std::operator| (_Ios_Iostate __a, _Ios_Iostate __b)`
- `_Ios_Fmtflags std::operator| (_Ios_Fmtflags __a, _Ios_Fmtflags __b)`
- `_Ios_Openmode std::operator| (_Ios_Openmode __a, _Ios_Openmode __b)`
- `_Ios_Fmtflags & std::operator|= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)`
- `_Ios_Openmode & std::operator|= (_Ios_Openmode &__a, _Ios_Openmode __b)`
- `_Ios_Iostate & std::operator|= (_Ios_Iostate &__a, _Ios_Iostate __b)`
- `_Ios_Iostate std::operator~ (_Ios_Iostate __a)`
- `_Ios_Fmtflags std::operator~ (_Ios_Fmtflags __a)`
- `_Ios_Openmode std::operator~ (_Ios_Openmode __a)`
- `ios_base & std::right (ios_base &__base)`
- `ios_base & std::scientific (ios_base &__base)`
- `ios_base & std::showbase (ios_base &__base)`
- `ios_base & std::showpoint (ios_base &__base)`
- `ios_base & std::showpos (ios_base &__base)`
- `ios_base & std::skipws (ios_base &__base)`
- `ios_base & std::unitbuf (ios_base &__base)`
- `ios_base & std::uppercase (ios_base &__base)`

### 6.165.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ios\\_base.h](#).

## 6.166 iosfwd File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- typedef basic\_filebuf< char > [std::filebuf](#)
- typedef basic\_fstream< char > [std::fstream](#)
- typedef basic\_ifstream< char > [std::ifstream](#)
- typedef basic\_ios< char > [std::ios](#)
- typedef basic\_iostream< char > [std::iostream](#)
- typedef basic\_istream< char > [std::istream](#)
- typedef basic\_istreamstream< char > [std::istreamstream](#)
- typedef basic\_ofstream< char > [std::ofstream](#)
- typedef basic\_ostream< char > [std::ostream](#)
- typedef basic\_ostringstream< char > [std::ostringstream](#)
- typedef basic\_streambuf< char > [std::streambuf](#)
- typedef basic\_stringbuf< char > [std::stringbuf](#)
- typedef basic\_stringstream< char > [std::stringstream](#)
- typedef basic\_filebuf< wchar\_t > [std::wfilebuf](#)
- typedef basic\_fstream< wchar\_t > [std::wfstream](#)
- typedef basic\_ifstream< wchar\_t > [std::wifstream](#)
- typedef basic\_ios< wchar\_t > [std::wios](#)
- typedef basic\_iostream< wchar\_t > [std::wiostream](#)
- typedef basic\_istream< wchar\_t > [std::wistream](#)
- typedef basic\_istreamstream< wchar\_t > [std::wistreamstream](#)
- typedef basic\_ofstream< wchar\_t > [std::wofstream](#)
- typedef basic\_ostream< wchar\_t > [std::wostream](#)
- typedef basic\_ostringstream< wchar\_t > [std::wostringstream](#)
- typedef basic\_streambuf< wchar\_t > [std::wstreambuf](#)
- typedef basic\_stringbuf< wchar\_t > [std::wstringbuf](#)
- typedef basic\_stringstream< wchar\_t > [std::wstringstream](#)

### 6.166.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iosfwd](#).

## 6.167 iostream File Reference

### Namespaces

- namespace [std](#)

### Variables

- static `ios_base::Init` **std::\_\_ioinit**

### Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html> and the *I/O forward declarations*

*They are required by default to cooperate with the global C library's FILE streams, and to be available during program startup and termination. For more information, see the HOWTO linked to above.*

- `istream` [std::cin](#)
- `ostream` [std::cout](#)
- `ostream` [std::cerr](#)
- `ostream` [std::clog](#)
- `wistream` [std::wcin](#)
- `wostream` [std::wcout](#)
- `wostream` [std::wcerr](#)
- `wostream` [std::wclog](#)

### 6.167.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iostream](#).



## 6.168 istream File Reference

### Classes

- class `std::basic_iostream< _CharT, _Traits >`  
*Merging istream and ostream capabilities.*  
*This class multiply inherits from the input and output stream classes simply to provide a single interface.*
- class `std::basic_istream< _CharT, _Traits >`  
*Controlling input.*  
*This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.*
- class `std::basic_istream< _CharT, _Traits >::sentry`  
*Performs setup work for input streams.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _CharT, typename _Traits, typename _Tp >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &&__is, _Tp &__x)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__in, _CharT *__s)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > &__in,`  
`char *__s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _-`  
`Traits > &__in, unsigned char *__s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _-`  
`Traits > &__in, signed char *__s)`

### 6.168.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [istream](#).

## 6.169 istream.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits`  
`> &__is)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__in, _CharT &__c)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _-`  
`CharT, _Traits > &__in, _CharT *__s)`

### 6.169.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [istream.tcc](#).

## 6.170 iterator File Reference

### 6.170.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [iterator](#).

## 6.171 iterator File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _InputIterator, typename _Distance >  
void \_\_gnu\_cxx::\_\_distance (_InputIterator __first, _InputIterator __last, _-  
Distance &__n, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >  
void \_\_gnu\_cxx::\_\_distance (_RandomAccessIterator __first, _-  
RandomAccessIterator __last, _Distance &__n, std::random\_access\_iterator\_-  
tag)`
- `template<typename _InputIterator, typename _Distance >  
void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _-  
Distance &__n)`

### 6.171.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/iterator](#).

## 6.172 iterator.h File Reference

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

## Classes

- class [\\_\\_gnu\\_parallel::\\_IteratorPair< \\_Iterator1, \\_Iterator2, \\_IteratorCategory >](#)  
*A pair of iterators. The usual iterator operations are applied to both child iterators.*
- class [\\_\\_gnu\\_parallel::\\_IteratorTriple< \\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >](#)  
*A triple of iterators. The usual iterator operations are applied to all three child iterators.*

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### 6.172.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [iterator.h](#).

## 6.173 iterator\_tracker.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >  
bool std::__profile::operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >  
bool std::__profile::operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence > std::__profile::operator+ (type-`  
`name __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const`  
`__iterator_tracker< _Iterator, _Sequence > &__i)`
- `template<typename _Iterator, typename _Sequence >`  
`__iterator_tracker< _Iterator, _Sequence >::difference_type std::__-`  
`profile::operator- (const __iterator_tracker< _Iterator, _Sequence > &__lhs,`  
`const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`__iterator_tracker< _IteratorL, _Sequence >::difference_type std::__-`  
`profile::operator- (const __iterator_tracker< _IteratorL, _Sequence > &__lhs,`  
`const __iterator_tracker< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator< (const __iterator_tracker< _Iterator, _Sequence`  
`> &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _Iterator, _-`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator<= (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _Iterator, _-`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator== (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator> (const __iterator_tracker< _Iterator, _Sequence`  
`> &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _Iterator, _-`  
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`  
`bool std::__profile::operator>= (const __iterator_tracker< _IteratorL, _-`  
`Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__-`  
`rhs)`

### 6.173.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [iterator\\_tracker.h](#).

## 6.174 limits File Reference

### Classes

- struct [std::\\_\\_numeric\\_limits\\_base](#)  
*Part of [std::numeric\\_limits](#).*
- struct [std::numeric\\_limits< \\_Tp >](#)  
*Properties of fundamental types.*
- struct [std::numeric\\_limits< bool >](#)  
*[numeric\\_limits<bool>](#) specialization.*
- struct [std::numeric\\_limits< char >](#)  
*[numeric\\_limits<char>](#) specialization.*
- struct [std::numeric\\_limits< char16\\_t >](#)  
*[numeric\\_limits<char16\\_t>](#) specialization.*
- struct [std::numeric\\_limits< char32\\_t >](#)  
*[numeric\\_limits<char32\\_t>](#) specialization.*
- struct [std::numeric\\_limits< double >](#)  
*[numeric\\_limits<double>](#) specialization.*
- struct [std::numeric\\_limits< float >](#)  
*[numeric\\_limits<float>](#) specialization.*
- struct [std::numeric\\_limits< int >](#)  
*[numeric\\_limits<int>](#) specialization.*

- struct `std::numeric_limits< long >`  
*numeric\_limits<long> specialization.*
- struct `std::numeric_limits< long double >`  
*numeric\_limits<long double> specialization.*
- struct `std::numeric_limits< long long >`  
*numeric\_limits<long long> specialization.*
- struct `std::numeric_limits< short >`  
*numeric\_limits<short> specialization.*
- struct `std::numeric_limits< signed char >`  
*numeric\_limits<signed char> specialization.*
- struct `std::numeric_limits< unsigned char >`  
*numeric\_limits<unsigned char> specialization.*
- struct `std::numeric_limits< unsigned int >`  
*numeric\_limits<unsigned int> specialization.*
- struct `std::numeric_limits< unsigned long >`  
*numeric\_limits<unsigned long> specialization.*
- struct `std::numeric_limits< unsigned long long >`  
*numeric\_limits<unsigned long long> specialization.*
- struct `std::numeric_limits< unsigned short >`  
*numeric\_limits<unsigned short> specialization.*
- struct `std::numeric_limits< wchar_t >`  
*numeric\_limits<wchar\_t> specialization.*

## Namespaces

- namespace `std`

## Defines

- `#define __glibcxx_digits(T)`
- `#define __glibcxx_digits10(T)`
- `#define __glibcxx_double_has_denorm_loss`
- `#define __glibcxx_double_tinyness_before`
- `#define __glibcxx_double_traps`
- `#define __glibcxx_float_has_denorm_loss`
- `#define __glibcxx_float_tinyness_before`
- `#define __glibcxx_float_traps`
- `#define __glibcxx_integral_traps`
- `#define __glibcxx_long_double_has_denorm_loss`
- `#define __glibcxx_long_double_tinyness_before`
- `#define __glibcxx_long_double_traps`
- `#define __glibcxx_max(T)`
- `#define __glibcxx_max_digits10(T)`
- `#define __glibcxx_min(T)`
- `#define __glibcxx_signed(T)`

## Enumerations

- `enum std::float_denorm_style { std::denorm_indeterminate, std::denorm_absent, std::denorm_present }`
- `enum std::float_round_style {`  
`std::round_indeterminate, std::round_toward_zero, std::round_to_nearest,`  
`std::round_toward_infinity,`  
`std::round_toward_neg_infinity }`

### 6.174.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [limits](#).

## 6.175 list File Reference

### 6.175.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [list](#).



## 6.176 list File Reference

### Classes

- class [std::\\_\\_debug::list< \\_Tp, \\_Allocator >](#)  
*Class [std::list](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
void std::__debug::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

#### 6.176.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/list](#).

## 6.177 list File Reference

### Classes

- class [std::\\_\\_profile::list< \\_Tp, \\_Allocator >](#)  
*List wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__profile::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
void std::__profile::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)`

#### 6.177.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/list](#).

## 6.178 list.tcc File Reference

### Namespaces

- namespace [std](#)

#### 6.178.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [list.tcc](#).

## 6.179 list\_partition.h File Reference

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter >  
void \_\_gnu\_parallel::\_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >  
void \_\_gnu\_parallel::\_\_shrink\_and\_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FunctorType >  
size_t \_\_gnu\_parallel::list\_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)`

#### 6.179.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [list\\_partition.h](#).

## 6.180 `list_update_policy.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 6.180.1 Detailed Description

Contains policies for list update containers.

Definition in file [list\\_update\\_policy.hpp](#).

## 6.181 `locale` File Reference

#### 6.181.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [locale](#).

## 6.182 `locale_classes.h` File Reference

### Classes

- class [std::collate<\\_CharT>](#)  
*Facet for localized string comparison.*
- class [std::collate\\_byname<\\_CharT>](#)  
*class [collate\\_byname](#) [22.2.4.2].*
- class [std::locale](#)

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

- class [std::locale::facet](#)

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

- class [std::locale::id](#)

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member [locale::id](#), or be derived from a facet that provides this member; otherwise the facet cannot be used in a locale. The [locale::id](#) ensures that each class type gets a unique identifier.

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Facet >  
bool std::has\_facet (const locale &__loc) throw ()`
- `template<typename _Facet >  
const _Facet & std::use\_facet (const locale &__loc)`

### 6.182.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_classes.h](#).

## 6.183 locale\_classes.tcc File Reference

### Namespaces

- namespace [std](#)

## Functions

- `template<typename _Facet >`  
`bool std::has\_facet (const locale &__loc) throw ()`
- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`

### 6.183.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_classes.tcc](#).

## 6.184 [locale\\_facets.h](#) File Reference

### Classes

- class [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#)  
*Common base for ctype facet.*
- class [std::ctype< \\_CharT >](#)  
*Primary class template ctype facet.*  
*This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.*
- class [std::ctype< char >](#)  
*The [ctype<char>](#) specialization.*  
*This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.*
- class [std::ctype< wchar\\_t >](#)  
*The [ctype<wchar\\_t>](#) specialization.*  
*This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.*
- class [std::ctype\\_byname< \\_CharT >](#)  
*class [ctype\\_byname](#) [22.2.1.2].*
- class [std::ctype\\_byname< char >](#)  
*22.2.1.4 Class [ctype\\_byname](#) specializations.*

- class `std::num_get<_CharT, _InIter >`  
*Primary class template `num_get`.  
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.*
- class `std::num_put<_CharT, _OutIter >`  
*Primary class template `num_put`.  
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.*
- class `std::numpunct<_CharT >`  
*Primary class template `numpunct`.  
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.*
- class `std::numpunct_byname<_CharT >`  
*class `numpunct_byname` [22.2.3.2].*

## Namespaces

- namespace `std`

## Defines

- `#define _GLIBCXX_NUM_FACETS`

## Functions

- `template<typename _CharT >`  
`_CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__-gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _Tp >`  
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __-c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __-c_locale &) throw ()`

- `template<>`  
`void std::__convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT >`  
`ostreambuf_iterator< _CharT > std::__write (ostreambuf_iterator< _CharT > __s, const _CharT * __ws, int __len)`
- `template<typename _CharT, typename _OutIter >`  
`_OutIter std::__write (_OutIter __s, const _CharT * __ws, int __len)`
- `template<typename _CharT >`  
`bool std::isalnum (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isalpha (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isctrl (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isgraph (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::islower (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isprint (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale & __loc)`

### 6.184.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets.h](#).



## 6.185 locale\_facets.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT >  
_CharT * std::__add_grouping (_CharT *__s, _CharT __sep, const char *__-  
gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _CharT, typename _ValueT >  
_GLIBCXX_END_LDBL_NAMESPACE int std::__int_to_char (_CharT *__-  
bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __-  
dec)`
- `_GLIBCXX_PURE bool std::__verify_grouping (const char *__grouping,  
size_t __grouping_size, const string &__grouping_tmp) throw ()`

### 6.185.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets.tcc](#).

## 6.186 locale\_facets\_nonio.h File Reference

### Classes

- class [std::messages<\\_CharT>](#)  
*Primary class template messages.  
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.*
- struct [std::messages\\_base](#)  
*Messages facet base class providing catalog typedef.*
- class [std::messages\\_byname<\\_CharT>](#)  
*class messages\_byname [22.2.7.2].*
- class [std::money\\_base](#)

*Money format ordering data.*

*This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.*

- class `std::money_get< _CharT, _InIter >`

*Primary class template `money_get`.*

*This facet encapsulates the code to parse and return a monetary amount from a string.*

- class `std::money_put< _CharT, _OutIter >`

*Primary class template `money_put`.*

*This facet encapsulates the code to format and output a monetary amount.*

- class `std::moneypunct< _CharT, _Intl >`

*Primary class template `moneypunct`.*

*This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.*

- class `std::moneypunct_byname< _CharT, _Intl >`

*class `moneypunct_byname` [22.2.6.4].*

- class `std::time_base`

*Time format ordering data.*

*This class provides an enum representing different orderings of time: day, month, and year.*

- class `std::time_get< _CharT, _InIter >`

*Primary class template `time_get`.*

*This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.*

- class `std::time_get_byname< _CharT, _InIter >`

*class `time_get_byname` [22.2.5.2].*

- class `std::time_put< _CharT, _OutIter >`

*Primary class template `time_put`.*

*This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.*

- class `std::time_put_byname< _CharT, _OutIter >`

*class `time_put_byname` [22.2.5.4].*

## Namespaces

- namespace [std](#)

### 6.186.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets\\_nonio.h](#).

## 6.187 locale\_facets\_nonio.tcc File Reference

## Namespaces

- namespace [std](#)

### 6.187.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [locale\\_facets\\_nonio.tcc](#).

## 6.188 localefwd.h File Reference

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Facet >  
bool std::has\_facet (const locale &__loc) throw ()`
- `template<typename _CharT >  
bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >  
bool std::isctrl (_CharT __c, const locale &__loc)`

- `template<typename _CharT >`  
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`

### 6.188.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [localefwd.h](#).

## 6.189 losertree.h File Reference

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`  
*Stable [\\_LoserTree](#) variant.*
- class `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`  
*Unstable [\\_LoserTree](#) variant.*

- class `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`  
*Guarded loser/tournament tree.*
- struct `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser`  
*Internal representation of a `_LoserTree` element.*
- class `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`  
*Stable `_LoserTree` implementation.*
- class `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`  
*Unstable `_LoserTree` implementation.*
- class `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`  
*Base class of `_LoserTree` implementation using pointers.*
- struct `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser`  
*Internal representation of `_LoserTree` \_\_elements.*
- class `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`  
*Stable unguarded `_LoserTree` variant storing pointers.*
- class `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`  
*Unstable unguarded `_LoserTree` variant storing pointers.*
- class `__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >`  
*Unguarded loser tree, keeping only pointers to the elements in the tree structure.*
- class `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`  
*Stable implementation of unguarded `_LoserTree`.*
- class `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`  
*Non-Stable implementation of unguarded `_LoserTree`.*
- class `__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >`  
*Base class for unguarded `_LoserTree` implementation.*

## Namespaces

- namespace `__gnu_parallel`

### 6.189.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [losertree.h](#).

## 6.190 macros.h File Reference

### Defines

- `#define __glibcxx_check_erase(_Position)`
- `#define __glibcxx_check_erase_after(_Position)`
- `#define __glibcxx_check_erase_range(_First, _Last)`
- `#define __glibcxx_check_erase_range_after(_First, _Last)`
- `#define __glibcxx_check_heap(_First, _Last)`
- `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_insert(_Position)`
- `#define __glibcxx_check_insert_after(_Position)`
- `#define __glibcxx_check_insert_range(_Position, _First, _Last)`
- `#define __glibcxx_check_insert_range_after(_Position, _First, _Last)`
- `#define __glibcxx_check_nonempty()`
- `#define __glibcxx_check_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_sorted(_First, _Last)`
- `#define __glibcxx_check_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_check_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_check_string(_String)`
- `#define __glibcxx_check_string_len(_String, _Len)`
- `#define __glibcxx_check_subscript(_N)`
- `#define __glibcxx_check_valid_range(_First, _Last)`
- `#define __GLIBCXX_DEBUG_VERIFY(_Condition, _ErrorMessage)`

### 6.190.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [macros.h](#).

## 6.190.2 Define Documentation

### 6.190.2.1 `#define __glibcxx_check_erase( _Position )`

Verify that we can erase the element referenced by the iterator *\_Position*. We can erase the element if the *\_Position* iterator is dereferenceable and references this sequence.

Definition at line 126 of file macros.h.

### 6.190.2.2 `#define __glibcxx_check_erase_after( _Position )`

Verify that we can erase the element after the iterator *\_Position*. We can erase the element if the *\_Position* iterator is before a dereferenceable one and references this sequence.

Definition at line 140 of file macros.h.

### 6.190.2.3 `#define __glibcxx_check_erase_range( _First, _Last )`

Verify that we can erase the elements in the iterator range [*\_First*, *\_Last*). We can erase the elements if [*\_First*, *\_Last*) is a valid iterator range within this sequence.

Definition at line 154 of file macros.h.

### 6.190.2.4 `#define __glibcxx_check_erase_range_after( _First, _Last )`

Verify that we can erase the elements in the iterator range (*\_First*, *\_Last*). We can erase the elements if (*\_First*, *\_Last*) is a valid iterator range within this sequence.

Definition at line 166 of file macros.h.

### 6.190.2.5 `#define __glibcxx_check_heap_pred( _First, _Last, _Pred )`

Verify that the iterator range [*\_First*, *\_Last*) is a heap w.r.t. the predicate *\_Pred*.

Definition at line 297 of file macros.h.

### 6.190.2.6 `#define __glibcxx_check_insert( _Position )`

Verify that we can insert into *\*this* with the iterator *\_Position*. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

Definition at line 64 of file macros.h.

#### 6.190.2.7 **#define \_\_glibcxx\_check\_insert\_after( *\_Position* )**

Verify that we can insert into \*this after the iterator *\_Position*. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

Definition at line 81 of file macros.h.

#### 6.190.2.8 **#define \_\_glibcxx\_check\_insert\_range( *\_Position*, *\_First*, *\_Last* )**

Verify that we can insert the values in the iterator range [*\_First*, *\_Last*) into \*this with the iterator *\_Position*. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [*\_First*, *\_Last*) is a valid (possibly empty) range. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

#### **Todo**

We would like to be able to check for noninterference of *\_Position* and the range [*\_First*, *\_Last*), but that can't (in general) be done.

Definition at line 101 of file macros.h.

#### 6.190.2.9 **#define \_\_glibcxx\_check\_insert\_range\_after( *\_Position*, *\_First*, *\_Last* )**

Verify that we can insert the values in the iterator range [*\_First*, *\_Last*) into \*this after the iterator *\_Position*. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [*\_First*, *\_Last*) is a valid (possibly empty) range. Note that this macro is only valid when the container is a *\_Safe\_sequence* and the iterator is a *\_Safe\_iterator*.

#### **Todo**

We would like to be able to check for noninterference of *\_Position* and the range [*\_First*, *\_Last*), but that can't (in general) be done.

Definition at line 118 of file macros.h.



**6.190.2.10** `#define __glibcxx_check_partitioned_lower( _First, _Last, _Value )`

Verify that the iterator range [*\_First*, *\_Last*) is partitioned w.r.t. the value *\_Value*.

Definition at line 245 of file macros.h.

**6.190.2.11** `#define __glibcxx_check_partitioned_lower_pred( _First, _Last, _Value, _Pred )`

Verify that the iterator range [*\_First*, *\_Last*) is partitioned w.r.t. the value *\_Value* and predicate *\_Pred*.

Definition at line 265 of file macros.h.

**6.190.2.12** `#define __glibcxx_check_partitioned_upper_pred( _First, _Last, _Value, _Pred )`

Verify that the iterator range [*\_First*, *\_Last*) is partitioned w.r.t. the value *\_Value* and predicate *\_Pred*.

Definition at line 277 of file macros.h.

**6.190.2.13** `#define __glibcxx_check_sorted_pred( _First, _Last, _Pred )`

Verify that the iterator range [*\_First*, *\_Last*) is sorted by the predicate *\_Pred*.

Definition at line 216 of file macros.h.

**6.190.2.14** `#define _GLIBCXX_DEBUG_VERIFY( _Condition, _ErrorMessage )`

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 42 of file macros.h.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator*()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator--()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=()`.

## 6.191 malloc\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::malloc\\_allocator< \\_Tp >](#)

*An allocator that uses malloc.*

*This is precisely the allocator defined in the C++ Standard.*

- *all allocation calls malloc*
- *all deallocation calls free.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`

#### 6.191.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [malloc\\_allocator.h](#).

## 6.192 map File Reference

### 6.192.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [map](#).

## 6.193 map File Reference

### 6.193.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map](#).

## 6.194 map File Reference

### 6.194.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map](#).

## 6.195 map.h File Reference

### Classes

- class [std::\\_\\_debug::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class [std::map](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator!= (const map< _Key, _Tp, _Compare, _Allocator  
> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator< (const map< _Key, _Tp, _Compare, _Allocator  
> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator<= (const map< _Key, _Tp, _Compare, _-  
Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const map< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const map< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 6.195.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/map.h](#).

## 6.196 map.h File Reference

### Classes

- class [std::\\_\\_profile::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class [std::map](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator!= (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator< (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator<= (const map< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator== (const map< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator> (const map< _Key, _Tp, _Compare, _Allocator`  
`> &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator>= (const map< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__profile::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs,`  
`map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 6.196.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/map.h](#).

## 6.197 mask\_array.h File Reference

### Classes

- class [std::mask\\_array< \\_Tp >](#)  
*Reference to selected subset of an array.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

### 6.197.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [mask\\_array.h](#).

## 6.198 memory File Reference

### 6.198.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [memory](#).

## 6.199 memory File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::temporary\\_buffer<\\_ForwardIterator, \\_Tp>](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_InputIter, typename \_Size, typename \_ForwardIter >  
pair< \_InputIter, \_ForwardIter > [\\_\\_gnu\\_cxx::\\_\\_uninitialized\\_copy\\_n](#) (\_InputIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result, [std::input\\_iterator\\_tag](#))
- template<typename \_RandomAccessIter, typename \_Size, typename \_ForwardIter >  
pair< \_RandomAccessIter, \_ForwardIter > [\\_\\_gnu\\_cxx::\\_\\_uninitialized\\_copy\\_n](#) (\_RandomAccessIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result, [std::random\\_access\\_iterator\\_tag](#))
- template<typename \_InputIter, typename \_Size, typename \_ForwardIter >  
pair< \_InputIter, \_ForwardIter > [\\_\\_gnu\\_cxx::\\_\\_uninitialized\\_copy\\_n](#) (\_InputIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result)
- template<typename \_InputIter, typename \_Size, typename \_ForwardIter, typename \_Tp >  
pair< \_InputIter, \_ForwardIter > [\\_\\_gnu\\_cxx::\\_\\_uninitialized\\_copy\\_n\\_a](#) (\_InputIter \_\_first, \_Size \_\_count, \_ForwardIter \_\_result, [std::allocator<\\_Tp>](#))

- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a (_-`  
`InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter`  
`__first, _Size __count, _ForwardIter __result)`

### 6.199.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/memory](#).

## 6.200 merge.h File Reference

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _-`  
`DifferenceTp, typename _Compare >`  
`_OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _-`  
`RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __-`  
`target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename`  
`_DifferenceTp, typename _Compare >`  
`_OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 &_-`  
`__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _-`  
`OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename`  
`_DifferenceTp, typename _Compare >`  
`_OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 &_-`  
`__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _-`  
`OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _-`  
`RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target,`  
`typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _-`  
`Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _-`  
`RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target,`  
`typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _-`  
`Compare __comp)`

### 6.200.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [merge.h](#).

## 6.201 `messages_members.h` File Reference

### Namespaces

- namespace [std](#)

### 6.201.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [messages\\_members.h](#).

## 6.202 `move.h` File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define \_GLIBCXX\_FORWARD(_Tp, __val)`
- `#define \_GLIBCXX\_MOVE(__val)`



## Functions

- `template<typename _Tp >  
_Tp * std::addressof (_Tp &__r)`
- `template<typename _Tp >  
_Tp * std::addressof (_Tp &__r)`
- `template<typename _Tp >  
enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type std::forward  
(typename std::common_type< _Tp >::type &__t)`
- `template<typename _Tp >  
enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type std::forward  
(typename std::common_type< _Tp >::type __t)`
- `template<typename _Tp >  
enable_if< is_lvalue_reference< _Tp >::value, _Tp >::type std::forward  
(typename std::remove_reference< _Tp >::type &&__t)`
- `template<typename _Tp >  
enable_if<!is_lvalue_reference< _Tp >::value, _Tp && >::type std::forward  
(typename std::common_type< _Tp >::type &&__t)`
- `template<typename _Tp >  
std::remove_reference< _Tp >::type && std::move (_Tp &&__t)`
- `template<typename _Tp, size_t _Nm>  
void std::swap (_Tp(&)[_Nm], _Tp(&)[_Nm])`
- `template<typename _Tp >  
void std::swap (_Tp &__a, _Tp &__b)`

### 6.202.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [move.h](#).

## 6.203 mt\_allocator.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_\\_common\\_pool\\_policy](#)< [\\_PoolTp](#), [\\_Thread](#) >  
*Policy for shared \_\_pool objects.*
- class [\\_\\_gnu\\_cxx::\\_\\_mt\\_alloc](#)< [\\_Tp](#), [\\_Poolp](#) >  
*This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list).  
Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.*

- class `__gnu_cxx::__mt_alloc_base<_Tp>`  
*Base class for \_Tp dependent member functions.*
- struct `__gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread>`  
*Policy for individual \_\_pool objects.*
- class `__gnu_cxx::__pool<false>`  
*Specialization for single thread.*
- class `__gnu_cxx::__pool<true>`  
*Specialization for thread enabled, via gthreads.h.*
- struct `__gnu_cxx::__pool_base`  
*Base class for pool object.*

## Namespaces

- namespace `__gnu_cxx`

## Defines

- `#define __thread_default`

## Typedefs

- typedef void(\* `__gnu_cxx::__destroy_handler`)(void \*)

## Functions

- template<typename \_Tp, typename \_Poolp>  
bool `__gnu_cxx::operator!=`(const \_\_mt\_alloc<\_Tp, \_Poolp> &, const \_\_mt\_alloc<\_Tp, \_Poolp> &)
- template<typename \_Tp, typename \_Poolp>  
bool `__gnu_cxx::operator==`(const \_\_mt\_alloc<\_Tp, \_Poolp> &, const \_\_mt\_alloc<\_Tp, \_Poolp> &)

### 6.203.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [mt\\_allocator.h](#).

## 6.204 multimap.h File Reference

### Classes

- class [std::\\_\\_debug::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class [std::multimap](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _-  
Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_  
rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _-  
Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_  
rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _-  
Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_  
rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _-  
Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_  
rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _-  
Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_  
rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`lhs, multimap< _Key, _Tp, _Compare, _Allocator > &rhs)`

### 6.204.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multimap.h](#).

## 6.205 multimap.h File Reference

### Classes

- class [std::\\_\\_profile::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)  
*Class [std::multimap](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator!= (const multimap< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator< (const multimap< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator<= (const multimap< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator== (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator> (const multimap< _Key, _Tp, _Compare, _-`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__profile::operator>= (const multimap< _Key, _Tp, _Compare, _`  
`Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &_`  
`rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__profile::swap (multimap< _Key, _Tp, _Compare, _Allocator >`  
`&__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

### 6.205.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multimap.h](#).

## 6.206 multiseq\_selection.h File Reference

Functions to find elements of a certain global `__rank` in multiple sorted sequences.  
 Also serves for splitting such sequence sets.

### Classes

- class [\\_\\_gnu\\_parallel::\\_Lexicographic< \\_T1, \\_T2, \\_Compare >](#)  
*Compare \_\_a pair of types lexicographically, ascending.*
- class [\\_\\_gnu\\_parallel::\\_LexicographicReverse< \\_T1, \\_T2, \\_Compare >](#)  
*Compare \_\_a pair of types lexicographically, descending.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Defines

- `#define __S(__i)`
- `#define __S(__i)`

## Functions

- `template<typename _RanSeqs , typename _RankType , typename _RankIterator , typename _Compare >`  
`void \_\_gnu\_parallel::multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp , typename _RanSeqs , typename _RankType , typename _Compare >`  
`_Tp \_\_gnu\_parallel::multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())`

### 6.206.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets. The algorithm description can be found in P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. Journal of Parallel and Distributed Computing, 12(2):171–177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiseq\\_selection.h](#).

## 6.207 multiset.h File Reference

### Classes

- class [std::\\_\\_debug::multiset< \\_Key, \\_Compare, \\_Allocator >](#)  
*Class [std::multiset](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator!= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator< (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator<= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x,`  
`multiset< _Key, _Compare, _Allocator > &__y)`

### 6.207.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/multiset.h](#).

## 6.208 multiset.h File Reference

### Classes

- class [std::\\_\\_profile::multiset< \\_Key, \\_Compare, \\_Allocator >](#)  
*Class [std::multiset](#) wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator!= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator< (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator<= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator== (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator> (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`bool std::__profile::operator>= (const multiset< _Key, _Compare, _Allocator`  
`> &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key , typename _Compare , typename _Allocator >`  
`void std::__profile::swap (multiset< _Key, _Compare, _Allocator > &__x,`  
`multiset< _Key, _Compare, _Allocator > &__y)`

### 6.208.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/multiset.h](#).

## 6.209 multiway\_merge.h File Reference

Implementation of sequential and parallel multiway merge.

### Classes

- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __-`  
`sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`  
*Switch for 3-way merging with \_\_sentinels turned off.*
- `struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _-`  
`RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`



*Switch for 3-way merging with `__sentinels` turned on.*

- struct `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for 4-way merging with `__sentinels` turned off.*

- struct `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for 4-way merging with `__sentinels` turned on.*

- struct `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for k-way merging with `__sentinels` turned on.*

- struct `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`

*Switch for k-way merging with `__sentinels` turned off.*

- class `__gnu_parallel::_GuardedIterator< _RAIter, _Compare >`

*\_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.*

- struct `__gnu_parallel::_LoserTreeTraits< _Tp >`

*Traits for determining whether the loser tree should use pointers or copies.*

- struct `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >`

*Stable sorting functor.*

- struct `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >`

*Non-\_\_stable sorting functor.*

## Namespaces

- namespace `__gnu_parallel`

## Defines

- `#define _GLIBCXX_PARALLEL_DECISION(__a, __b, __c, __d)`

- `#define _GLIBCXX_PARALLEL_LENGTH(__s)`
- `#define _GLIBCXX_PARALLEL_MERGE_3_CASE(__a, __b, __c, __c0, __c1)`
- `#define _GLIBCXX_PARALLEL_MERGE_4_CASE(__a, __b, __c, __d, __c0, __c1, __c2)`

## Functions

- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::sequential\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag\(0\))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 \_\_gnu\_parallel::multiway\_merge\_3\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator ,  
typename _RAIter3 , typename _DifferenceTp , typename _Compare >  
_RAIter3 __gnu_parallel::multiway_merge_4_variant (_RAIterIterator __-  
seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp  
__length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _-  
DifferenceType >  
void __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator __-  
seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _-  
DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _-  
DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp  
, typename _Compare >  
_RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __-  
seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp _-  
__length, _Compare __comp)`
- `template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 ,  
typename _DifferenceTp , typename _Compare >  
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_-  
RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __-  
target, const typename std::iterator_traits< typename std::iterator_traits<  
_RAIterIterator >::value_type::first_type >::value_type &__sentinel, _-  
DifferenceTp __length, _Compare __comp)`
- `template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _-  
DifferenceTp , typename _Compare >  
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_-  
RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __-  
target, const typename std::iterator_traits< typename std::iterator_traits<  
_RAIterIterator >::value_type::first_type >::value_type &__sentinel, _-  
DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator , typename _Compare , typename _-  
DifferenceType >  
void __gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator _-  
seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _-  
DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _-  
DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-  
name _Compare >  
_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator  
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-  
DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , type-  
name _Compare >  
_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator _-`

```

_seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
 _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
 _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, sampling_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
 _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator
__seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
• template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, type-
name _DifferenceTp, typename _Splitter, typename _Compare >
 _RAIter3 __gnu_parallel::parallel_multiway_merge (_RAIterIterator __seqs_-
begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _-
DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
 _RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator _-
seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
 _RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator _-
seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
 _RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator _-
seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, sampling_tag __tag)
• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >
 _RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator _-
seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _-
DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))

• template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, type-
name _Compare >

```

- ```

_RAlterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __-
_seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, __-
DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)

```
- ```

template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp ,
typename _Compare >
_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (__-
RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut
__target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)

```
  - ```

template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp ,
typename _Compare >
_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (__-
RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut
__target, _DifferenceTp __length, _Compare __comp, default_parallel_tag
__tag)

```
 - ```

template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp ,
typename _Compare >
_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (__-
RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut
__target, _DifferenceTp __length, _Compare __comp, parallel_tag __-
tag=parallel_tag(0))

```
  - ```

template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp ,
typename _Compare >
_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (__-
RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut
__target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_-
tag __tag)

```
 - ```

template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp ,
typename _Compare >
_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (__-
RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, __-
RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_-
parallel::sequential_tag)

```

### 6.209.1 Detailed Description

Implementation of sequential and parallel multiway merge. Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway\\_merge.h](#).

## 6.209.2 Define Documentation

### 6.209.2.1 #define \_GLIBCXX\_PARALLEL\_LENGTH( \_\_s )

Length of a sequence described by a pair of iterators.

Definition at line 53 of file multiway\_merge.h.

Referenced by `__gnu_parallel::__sequential_multiway_merge()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `__gnu_parallel::multiway_merge_loser_tree()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

## 6.210 multiway\_mergesort.h File Reference

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- struct [\\_\\_gnu\\_parallel::\\_Piece< \\_DifferenceTp >](#)  
*Subsequence description.*
- struct [\\_\\_gnu\\_parallel::\\_PMWMSSortingData< \\_RAIter >](#)  
*Data accessed by all threads.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< \\_\\_exact, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)  
*Split consistently.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)  
*Split by sampling.*
- struct [\\_\\_gnu\\_parallel::\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)  
*Split by exact splitting.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- template<typename \_RAIter, typename \_DifferenceTp >  
void [\\_\\_gnu\\_parallel::\\_\\_determine\\_samples](#) (\_PMWMSSortingData< \_RAIter > \* \_\_sd, \_DifferenceTp \_\_num\_samples)
- template<bool \_\_stable, bool \_\_exact, typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<bool \_\_stable, bool \_\_exact, typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::parallel\\_sort\\_mwms\\_pu](#) (\_PMWMSSortingData< \_RAIter > \* \_\_sd, \_Compare & \_\_comp)

### 6.210.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [multiway\\_mergesort.h](#).

## 6.211 mutex File Reference

### Classes

- struct [std::adopt\\_lock\\_t](#)  
*Assume the calling thread has already obtained mutex ownership /// and manage it.*
- struct [std::defer\\_lock\\_t](#)  
*Do not acquire ownership of the mutex.*
- class [std::lock\\_guard< \\_Mutex >](#)  
*Scoped lock idiom.*
- class [std::mutex](#)  
*mutex*
- struct [std::once\\_flag](#)  
*once\_flag*
- class [std::recursive\\_mutex](#)  
*recursive\_mutex*
- class [std::recursive\\_timed\\_mutex](#)

*recursive\_timed\_mutex*

- class `std::timed_mutex`  
*timed\_mutex*
- struct `std::try_to_lock_t`  
*Try to acquire ownership of the mutex without blocking.*
- class `std::unique_lock<_Mutex>`  
*unique\_lock*

## Namespaces

- namespace `std`

## Functions

- mutex & `std::__get_once_mutex()`
- void `std::__once_proxy()`
- void `std::__set_once_functor_lock_ptr` (unique\_lock< mutex > \*)
- template<typename \_Callable, typename... \_Args>  
void `std::call_once` (once\_flag &\_\_once, \_Callable &&\_\_f, \_Args &&...\_\_args)
- template<typename \_L1, typename \_L2, typename... \_L3>  
void `std::lock` (\_L1 &, \_L2 &, \_L3 &...)
- template<typename \_Mutex>  
void `std::swap` (unique\_lock< \_Mutex > &\_\_x, unique\_lock< \_Mutex > &\_\_y)
- template<typename \_Lock1, typename \_Lock2, typename... \_Lock3>  
int `std::try_lock` (\_Lock1 &\_\_l1, \_Lock2 &\_\_l2, \_Lock3 &...\_\_l3)

## Variables

- function< void()> `std::__once_functor`
- const adopt\_lock\_t `std::adopt_lock`
- const defer\_lock\_t `std::defer_lock`
- const try\_to\_lock\_t `std::try_to_lock`

### 6.211.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [mutex](#).



## 6.212 nested\_exception.h File Reference

### Classes

- class [std::nested\\_exception](#)  
*Exception class with exception\_ptr data member.*

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Ex >  
const nested_exception * std::\_\_get\_nested\_exception (const _Ex &__ex)`
- `template<typename _Ex >  
void std::\_\_throw\_with\_nested (_Ex &&, const nested_exception *__ex) \_\_attribute\_\_\(\(\_\_noreturn\_\_\)\)`
- `template<typename _Ex >  
void std::\_\_throw\_with\_nested (_Ex &&,...) \_\_attribute\_\_\(\(\_\_noreturn\_\_\)\)`
- `void std::rethrow\_if\_nested (const nested_exception &__ex)`
- `template<typename _Ex >  
void std::rethrow\_if\_nested (const _Ex &__ex)`
- `template<typename _Ex >  
void std::throw\_with\_nested (_Ex __ex)`

#### 6.212.1 Detailed Description

This is an internal header file, included by other headers and the implementation. You should not attempt to use it directly.

Definition in file [nested\\_exception.h](#).

## 6.213 new File Reference

### Classes

- class [std::bad\\_alloc](#)  
*Exception possibly thrown by `new`.  
`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.*

## Namespaces

- namespace [std](#)

## Typedefs

- typedef void(\* [std::new\\_handler](#) )()

## Functions

- new\_handler [std::set\\_new\\_handler](#) (new\_handler) throw ()
- void \* [operator new](#) (std::size\_t) throw (std::bad\_alloc)
- void \* [operator new\[ \]](#) (std::size\_t) throw (std::bad\_alloc)
- void [operator delete](#) (void \*) throw ()
- void [operator delete\[ \]](#) (void \*) throw ()
- void \* [operator new](#) (std::size\_t, const std::nothrow\_t &) throw ()
- void \* [operator new\[ \]](#) (std::size\_t, const std::nothrow\_t &) throw ()
- void [operator delete](#) (void \*, const std::nothrow\_t &) throw ()
- void [operator delete\[ \]](#) (void \*, const std::nothrow\_t &) throw ()
- void \* [operator new](#) (std::size\_t, void \*\_\_p) throw ()
- void \* [operator new\[ \]](#) (std::size\_t, void \*\_\_p) throw ()
- void [operator delete](#) (void \*, void \*) throw ()
- void [operator delete\[ \]](#) (void \*, void \*) throw ()

## Variables

- const nothrow\_t [std::nothrow](#)

### 6.213.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see [http://gcc.gnu.org/onlinedocs/libstdc++/18\\_support/howto.html#4](http://gcc.gnu.org/onlinedocs/libstdc++/18_support/howto.html#4) for more.

Definition in file [new](#).

## 6.213.2 Function Documentation

### 6.213.2.1 `void operator delete ( void * ) throw ()`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

### 6.213.2.2 `void operator delete ( void *, const std::nothrow_t & ) throw ()`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

### 6.213.2.3 `void operator delete ( void *, void * ) throw () [inline]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 107 of file new.

#### 6.213.2.4 void operator delete[] ( void \* ) throw ()

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

#### 6.213.2.5 void operator delete[] ( void \*, const std::nothrow\_t & ) throw ()

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

#### 6.213.2.6 void operator delete[] ( void \*, void \* ) throw () [inline]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)

- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 108 of file new.

#### 6.213.2.7 `void* operator new ( std::size_t, void * __p ) throw () [inline]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 103 of file new.

#### 6.213.2.8 `void* operator new ( std::size_t, const std::nothrow_t & ) throw ()`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

#### 6.213.2.9 `void* operator new ( std::size_t ) throw (std::bad_alloc)`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)

- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

#### 6.213.2.10 `void* operator new[] ( std::size_t, const std::nothrow_t & ) throw ()`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

#### 6.213.2.11 `void* operator new[] ( std::size_t ) throw (std::bad_alloc)`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

### 6.213.2.12 void\* operator new[] ( std::size\_t, void \* \_\_p ) throw () [inline]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 104 of file `new`.

## 6.214 new\_allocator.h File Reference

### Classes

- class `__gnu_cxx::new_allocator<_Tp>`  
*An allocator that uses global new, as per [20.4].  
This is precisely the allocator defined in the C++ Standard.*
  - all allocation calls `operator new`
  - all deallocation calls `operator delete`.

### Namespaces

- namespace `__gnu_cxx`

### Functions

- `template<typename _Tp>`  
`bool __gnu_cxx::operator!= (const new_allocator<_Tp> &, const new_allocator<_Tp> &)`
- `template<typename _Tp>`  
`bool __gnu_cxx::operator== (const new_allocator<_Tp> &, const new_allocator<_Tp> &)`

### 6.214.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [new\\_allocator.h](#).

## 6.215 numeric File Reference

### 6.215.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [numeric](#).

## 6.216 numeric File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp, typename _Integer, typename _MonoidOperation >  
_Tp \_\_gnu\_cxx::\_\_power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >  
_Tp \_\_gnu\_cxx::\_\_power (_Tp __x, _Integer __n)`
- `template<typename _ForwardIter, typename _Tp >  
void \_\_gnu\_cxx::iota (_ForwardIter __first, _ForwardIter __last, _Tp __value)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >  
_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >  
_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`

### 6.216.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [ext/numeric](#).



## 6.217 numeric File Reference

Parallel STL function calls corresponding to [stl\\_numeric.h](#). The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >  
_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >  
_Tp std::__parallel::__accumulate_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename __RAIter, typename _Tp, typename _BinaryOperation >  
_Tp std::__parallel::__accumulate_switch (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >  
_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >  
_OutputIterator std::__parallel::__adjacent_difference_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >  
_Tp std::__parallel::__inner_product_switch (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)`

- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Tp std::__parallel::__inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::__accumulate (_Iter __begin, _Iter __end, _Tp __init)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::__adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::__adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`

- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`

### 6.217.1 Detailed Description

Parallel STL function calls corresponding to [stl\\_numeric.h](#). The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel/numeric](#).

## 6.218 numeric\_traits.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- `#define __glibcxx_digits(_Tp)`
- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max(_Tp)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define __glibcxx_min(_Tp)`
- `#define __glibcxx_signed(_Tp)`

### 6.218.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [numeric\\_traits.h](#).

## 6.219 numericfwd.h File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_parallel](#)

## Functions

- `template<typename _Iter, typename _Tp, typename _Tag >  
_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >  
_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper,  
_Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >  
_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _-  
BinaryOper, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __-  
parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename  
_Tag2 >  
_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _-  
BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >  
_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _-  
BinaryOper, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_-  
parallel::Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1,  
typename BinaryFunction2 >  
_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _-  
RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_-  
tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism=__gnu_-  
parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, type-  
name _BinaryFunction2, typename _Tag1, typename _Tag2 >  
_Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp,  
_BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename  
_Tag2 >  
_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _-  
BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >  
_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _-  
BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >  
_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, \_\_-  
gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >  
_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_-  
parallel::Parallelism)`
- `template<typename _Iter, typename _Tp >  
_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init)`

- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init)`

- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`

### 6.219.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [numericfwd.h](#).

## 6.220 omp\_loop.h File Reference

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop.  
This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_omp\_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits<_RAIter>::difference_type __bound)`

### 6.220.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop.  
This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp\\_loop.h](#).

## 6.221 `omp_loop_static.h` File Reference

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
>  
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static  
(_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _-  
Result __base, _Result &__output, typename std::iterator_traits< _RAIter  
>::difference_type __bound)`

#### 6.221.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [omp\\_loop\\_static.h](#).

## 6.222 `os_defines.h` File Reference

### Defines

- `#define __NO_CTYPE`

#### 6.222.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [os\\_defines.h](#).



## 6.223 ostream File Reference

### Classes

- class `std::basic_ostream< _CharT, _Traits >`  
*Controlling output.*  
*This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.*
- class `std::basic_ostream< _CharT, _Traits >::sentry`  
*Performs setup work for output streams.*

### Namespaces

- namespace `std`

### Functions

- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::endl (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::ends (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::flush (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits, typename _Tp >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const _Tp &__x)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, char __c)`

- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, signed char __c)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, unsigned char __c)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, const char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, const signed char *__s)`
- `template<class _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char,`  
`_Traits > &__out, const unsigned char *__s)`

### 6.223.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ostream](#).

## 6.224 ostream.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__out, const char *__s)`

### 6.224.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ostream.tcc](#).

## 6.225 ostream\_insert.h File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits >  
void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, stream-  
size __n)`
- `template wostream & std::__ostream_insert (wostream &, const wchar_t *,  
streamsize)`
- `template ostream & std::__ostream_insert (ostream &, const char *, stream-  
size)`
- `template<typename _CharT, typename _Traits >  
basic_ostream< _CharT, _Traits > & std::__ostream_insert (basic_ostream<  
_CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >  
void std::__ostream_write (basic_ostream< _CharT, _Traits > &__out, const  
_CharT *__s, streamsize __n)`

### 6.225.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ostream\\_insert.h](#).

## 6.226 par\_loop.h File Reference

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>  
>  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_ed (_RAIter __-  
begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _-  
Result &__output, typename std::iterator_traits<_RAIter>::difference_type __-  
bound)`

### 6.226.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [par\\_loop.h](#).

## 6.227 [parallel.h](#) File Reference

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

### 6.227.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [parallel.h](#).

## 6.228 [partial\\_sum.h](#) File Reference

Parallel implementation of [std::partial\\_sum\(\)](#), i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum (_Iter __begin, _Iter  
__end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_basecase (_Iter __-  
begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, type-  
name std::iterator_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_linear (_Iter __begin,  
_Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename  
std::iterator_traits< _Iter >::difference_type __n)`

### 6.228.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partial\\_sum.h](#).

## 6.229 partition.h File Reference

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

## Defines

- `#define \_GLIBCXX\_VOLATILE`

## Functions

- `template<typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_nth\_element (_RAIter __begin, _RAIter __nth,  
_RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_partial\_sort (_RAIter __begin, _RAIter __-  
middle, _RAIter __end, _Compare __comp)`

- `template<typename _RAIter, typename _Predicate>`  
`std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_`  
`partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex`  
`__num_threads)`

### 6.229.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [partition.h](#).

### 6.229.2 Define Documentation

#### 6.229.2.1 `#define _GLIBCXX_VOLATILE`

Decide whether to declare certain variables volatile.

Definition at line 43 of file [partition.h](#).

Referenced by `__gnu_parallel::__parallel_partition()`.

## 6.230 `pod_char_traits.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::character< V, I, S >](#)  
*A POD class that serves as a character abstraction class.*
- struct [std::char\\_traits< \\_\\_gnu\\_cxx::character< V, I, S > >](#)  
*char\_traits<\_\_gnu\_cxx::character> specialization.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `template<typename V, typename I, typename S>`  
`bool __gnu_cxx::operator< (const character< V, I, S > &lhs, const character<`  
`V, I, S > &rhs)`

- `template<typename V , typename I , typename S >`  
`bool __gnu_cxx::operator== (const character< V, I, S > &lhs, const`  
`character< V, I, S > &rhs)`

### 6.230.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pod\\_char\\_traits.h](#).

## 6.231 pointer.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::Invalid\\_type](#)
- class [\\_\\_gnu\\_cxx::Pointer\\_adapter< \\_Storage\\_policy >](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl< \\_Tp >](#)  
*A storage policy for use with [\\_Pointer\\_adapter<>](#) which stores the pointer's address as an offset value which is relative to its own address.*
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl< const \\_Tp >](#)
- class [\\_\\_gnu\\_cxx::Std\\_pointer\\_impl< \\_Tp >](#)  
*A storage policy for use with [\\_Pointer\\_adapter<>](#) which yields a standard pointer.*
- struct [\\_\\_gnu\\_cxx::Unqualified\\_type< \\_Tp >](#)

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Defines

- `#define _CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define _GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

### Functions

- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator!= (const \_Pointer\_adapter< \_Tp1 > &__lhs, \_Tp2`  
`__rhs)`

- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __-`  
`rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__-`  
`rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const`  
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<<`  
`(std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter<`  
`_StoreT > &__p)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const`  
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const`  
`_Pointer_adapter< _Tp > &__rhs)`



- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __-`  
`rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__-`  
`rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _-`  
`Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2`  
`__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const`  
`_Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 >`  
`&__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const`  
`_Pointer_adapter< _Tp2 > &__rhs)`

### 6.231.1 Detailed Description

#### Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

Definition in file [pointer.h](#).

## 6.232 `pool_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc<\\_Tp>](#)  
*Allocator using a memory pool with a single lock.*
- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc\\_base](#)  
*Base class for [\\_\\_pool\\_alloc](#).*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp>`  
`bool \_\_gnu\_cxx::operator!= (const \_\_pool\_alloc<\_Tp> &, const \_\_pool\_alloc<\_Tp> &)`
- `template<typename _Tp>`  
`bool \_\_gnu\_cxx::operator== (const \_\_pool\_alloc<\_Tp> &, const \_\_pool\_alloc<\_Tp> &)`

#### 6.232.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [pool\\_allocator.h](#).

## 6.233 `postypes.h` File Reference

### Classes

- class `std::fpos<\_StateT>`

*Class representing stream positions.*

## Namespaces

- namespace [std](#)

## Typedefs

- typedef long long [std::streamoff](#)
- typedef fpos< mbstate\_t > [std::streampos](#)
- typedef ptrdiff\_t [std::streamsize](#)
- typedef fpos< mbstate\_t > [std::u16streampos](#)
- typedef fpos< mbstate\_t > [std::u32streampos](#)
- typedef fpos< mbstate\_t > [std::wstreampos](#)

## Functions

- template<typename \_StateT >  
bool **std::operator!=** (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)
- template<typename \_StateT >  
bool **std::operator==** (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)

### 6.233.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [postypes.h](#).

## 6.234 priority\_queue.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.234.1 Detailed Description

Contains `priority_queues`.

Definition in file [priority\\_queue.hpp](#).

## 6.235 `priority_queue_base_dispatch.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.235.1 Detailed Description

Contains an pqiative container dispatching base.

Definition in file [priority\\_queue\\_base\\_dispatch.hpp](#).

## 6.236 `profiler.h` File Reference

Interface of the profiling runtime library.

### Classes

- struct [\\_\\_gnu\\_profile::\\_\\_reentrance\\_guard](#)  
*Reentrance guard.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Defines

- `#define __profcxx_hashtable_construct(__x...)`
- `#define __profcxx_hashtable_construct2(__x...)`
- `#define __profcxx_hashtable_destruct(__x...)`
- `#define __profcxx_hashtable_destruct2(__x...)`

- #define **\_\_profcxx\_hashtable\_resize**(\_\_x...)
- #define **\_\_profcxx\_is\_invalid**()
- #define **\_\_profcxx\_is\_off**()
- #define **\_\_profcxx\_is\_on**()
- #define **\_\_profcxx\_list\_construct**(\_\_x...)
- #define **\_\_profcxx\_list\_construct2**(\_\_x...)
- #define **\_\_profcxx\_list\_destruct**(\_\_x...)
- #define **\_\_profcxx\_list\_destruct2**(\_\_x...)
- #define **\_\_profcxx\_list\_insert**(\_\_x...)
- #define **\_\_profcxx\_list\_invalid\_operator**(\_\_x...)
- #define **\_\_profcxx\_list\_iterate**(\_\_x...)
- #define **\_\_profcxx\_list\_operation**(\_\_x...)
- #define **\_\_profcxx\_list\_rewind**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_construct**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_destruct**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_erase**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_find**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_insert**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_invalidate**(\_\_x...)
- #define **\_\_profcxx\_map\_to\_unordered\_map\_iterate**(\_\_x...)
- #define **\_\_profcxx\_report**()
- #define **\_\_profcxx\_turn\_off**()
- #define **\_\_profcxx\_turn\_on**()
- #define **\_\_profcxx\_vector\_construct**(\_\_x...)
- #define **\_\_profcxx\_vector\_construct2**(\_\_x...)
- #define **\_\_profcxx\_vector\_destruct**(\_\_x...)
- #define **\_\_profcxx\_vector\_destruct2**(\_\_x...)
- #define **\_\_profcxx\_vector\_find**(\_\_x...)
- #define **\_\_profcxx\_vector\_insert**(\_\_x...)
- #define **\_\_profcxx\_vector\_invalid\_operator**(\_\_x...)
- #define **\_\_profcxx\_vector\_iterate**(\_\_x...)
- #define **\_\_profcxx\_vector\_resize**(\_\_x...)
- #define **\_\_profcxx\_vector\_resize2**(\_\_x...)
- #define **\_GLIBCXX\_PROFILE\_DATA**(\_\_name)
- #define **\_GLIBCXX\_PROFILE\_DEFINE\_DATA**(\_\_type, \_\_name, \_\_initial\_value...)
- #define **\_GLIBCXX\_PROFILE\_DEFINE\_UNINIT\_DATA**(\_\_type, \_\_name)
- #define **\_GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH**
- #define **\_GLIBCXX\_PROFILE\_MAX\_STACK\_DEPTH\_ENV\_VAR**
- #define **\_GLIBCXX\_PROFILE\_MAX\_WARN\_COUNT**
- #define **\_GLIBCXX\_PROFILE\_MAX\_WARN\_COUNT\_ENV\_VAR**
- #define **\_GLIBCXX\_PROFILE\_MEM\_PER\_DIAGNOSTIC**
- #define **\_GLIBCXX\_PROFILE\_MEM\_PER\_DIAGNOSTIC\_ENV\_VAR**

- `#define _GLIBCXX_PROFILE_REENTRANCE_GUARD(__x...)`
- `#define _GLIBCXX_PROFILE_TRACE_ENV_VAR`
- `#define _GLIBCXX_PROFILE_TRACE_PATH_ROOT`

## Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__trace_hash_func_construct (const void *)`
- `void __gnu_profile::__trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_construct (const void *, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_construct (const void *)`
- `void __gnu_profile::__trace_list_to_set_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_set_find (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_set_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_slist_construct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_operation (const void *)`
- `void __gnu_profile::__trace_list_to_slist_rewind (const void *)`
- `void __gnu_profile::__trace_list_to_vector_construct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_vector_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_construct (const void *)`
- `void __gnu_profile::__trace_map_to_unordered_map_destruct (const void *)`

- void `__gnu_profile::__trace_map_to_unordered_map_erase` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_find` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_insert` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_map_to_unordered_map_invalidate` (const void \*)
- void `__gnu_profile::__trace_map_to_unordered_map_iterate` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_size_construct` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_size_resize` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_construct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_find` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_insert` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void \*)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void \*, std::size\_t)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void \*, std::size\_t, std::size\_t)
- bool `__gnu_profile::__turn_off` ()
- bool `__gnu_profile::__turn_on` ()

### 6.236.1 Detailed Description

Interface of the profiling runtime library.

Definition in file [profiler.h](#).

## 6.237 profiler\_algos.h File Reference

Algorithms used by the profile extension.

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- `template<typename _InputIterator, typename _Function >`  
`_Function __gnu_profile::__for_each (_InputIterator __first, _InputIterator __-`  
`last, _Function __f)`
- `template<typename _Container >`  
`void __gnu_profile::__insert_top_n (_Container &__output, const typename`  
`_Container::value_type &__value, typename _Container::size_type __n)`
- `template<typename _ForwardIterator, typename _Tp >`  
`_ForwardIterator __gnu_profile::__remove (_ForwardIterator __first, _-`  
`ForwardIterator __last, const _Tp &__value)`
- `template<typename _Container >`  
`void __gnu_profile::__top_n (const _Container &__input, _Container &__-`  
`output, typename _Container::size_type __n)`

### 6.237.1 Detailed Description

Algorithms used by the profile extension. This file is needed to avoid including `<algorithm>` or `<bits/stl_algo.h>`. Including those files would result in recursive includes. These implementations are oversimplified. In general, efficiency may be sacrificed to minimize maintenance overhead.

Definition in file [profiler\\_algos.h](#).

## 6.238 profiler\_hashtable\_size.h File Reference

Collection of hashtable size traces.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_hashtable\\_size](#)  
*Hashtable size instrumentation trace producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- `void __gnu_profile::__trace_hashtable_size_construct (const void *,`  
`std::size_t)`



- void `__gnu_profile::__trace_hashtable_size_destruct` (const void \*, std::size\_t, std::size\_t)
- void `__gnu_profile::__trace_hashtable_size_init` ()
- void `__gnu_profile::__trace_hashtable_size_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__gnu_profile::__trace_hashtable_size_resize` (const void \*, std::size\_t, std::size\_t)

### 6.238.1 Detailed Description

Collection of hashtable size traces.

Definition in file [profiler\\_hashtable\\_size.h](#).

## 6.239 profiler\_list\_to\_slist.h File Reference

Diagnostics for list to slist.

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- void `__gnu_profile::__trace_list_to_slist_construct` (const void \*)
- void `__gnu_profile::__trace_list_to_slist_destruct` (const void \*)
- void `__gnu_profile::__trace_list_to_slist_init` ()
- void `__gnu_profile::__trace_list_to_slist_operation` (const void \*)
- void `__gnu_profile::__trace_list_to_slist_report` (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void `__gnu_profile::__trace_list_to_slist_rewind` (const void \*)

### 6.239.1 Detailed Description

Diagnostics for list to slist.

Definition in file [profiler\\_list\\_to\\_slist.h](#).

## 6.240 profiler\_list\_to\_vector.h File Reference

diagnostics for list to vector.

## Classes

- class [\\_\\_gnu\\_profile::\\_\\_list2vector\\_info](#)

*A list-to-vector instrumentation line in the object table.*

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_insert](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_invalid\\_operator](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_iterate](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_report](#) (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_list\\_to\\_vector\\_resize](#) (const void \*, std::size\_t, std::size\_t)

### 6.240.1 Detailed Description

diagnostics for list to vector.

Definition in file [profiler\\_list\\_to\\_vector.h](#).

## 6.241 profiler\_map\_to\_unordered\_map.h File Reference

Diagnostics for map to unordered\_map.

## Classes

- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_info](#)

*A map-to-unordered\_map instrumentation line in the object table.*

- class [\\_\\_gnu\\_profile::\\_\\_map2umap\\_stack\\_info](#)  
*A map-to-unordered\_map instrumentation line in the stack table.*
- class [\\_\\_gnu\\_profile::\\_\\_trace\\_map2umap](#)  
*Map-to-unordered\_map instrumentation producer.*

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- int [\\_\\_gnu\\_profile::\\_\\_log2](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_erase\\_cost](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_find\\_cost](#) (std::size\_t \_\_size)
- float [\\_\\_gnu\\_profile::\\_\\_map\\_insert\\_cost](#) (std::size\_t \_\_size)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_erase](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_find](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_insert](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_invalidate](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_iterate](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_map\\_to\\_unordered\\_map\\_report](#) (FILE \*\_\_f, \_\_warning\_vector\_t &\_\_warnings)

### 6.241.1 Detailed Description

Diagnostics for map to unordered\_map.

Definition in file [profiler\\_map\\_to\\_unordered\\_map.h](#).

## 6.242 profiler\_node.h File Reference

Data structures to represent a single profiling event.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_object\\_info\\_base](#)  
*Base class for a line in the object table.*
- class [\\_\\_gnu\\_profile::\\_\\_stack\\_hash](#)  
*Hash function for summary trace using call stack as index.*
- class [\\_\\_gnu\\_profile::\\_\\_stack\\_info\\_base](#)< [\\_\\_object\\_info](#) >  
*Base class for a line in the stack table.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Typedefs

- typedef void \* [\\_\\_gnu\\_profile::\\_\\_instruction\\_address\\_t](#)
- typedef const void \* [\\_\\_gnu\\_profile::\\_\\_object\\_t](#)
- typedef std::vector< [\\_\\_instruction\\_address\\_t](#) > [\\_\\_gnu\\_profile::\\_\\_stack\\_npt](#)
- typedef [\\_\\_stack\\_npt](#) \* [\\_\\_gnu\\_profile::\\_\\_stack\\_t](#)

### Functions

- [\\_\\_stack\\_t \\_\\_gnu\\_profile::\\_\\_get\\_stack \(\)](#)
- [std::size\\_t \\_\\_gnu\\_profile::\\_\\_size \(\\_\\_stack\\_t \\_\\_stack\)](#)
- [std::size\\_t \\_\\_gnu\\_profile::\\_\\_stack\\_max\\_depth \(\)](#)
- [void \\_\\_gnu\\_profile::\\_\\_write \(FILE \\*\\_\\_f, \\_\\_stack\\_t \\_\\_stack\)](#)

#### 6.242.1 Detailed Description

Data structures to represent a single profiling event.

Definition in file [profiler\\_node.h](#).

## 6.243 profiler\_state.h File Reference

Global profiler state.

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

### Functions

- bool `__gnu_profile::__is_invalid ()`
- bool `__gnu_profile::__is_off ()`
- bool `__gnu_profile::__is_on ()`
- bool `__gnu_profile::__turn (__state_type __s)`
- bool `__gnu_profile::__turn_off ()`
- bool `__gnu_profile::__turn_on ()`
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA (__state_type, __state, __INVALID)`

#### 6.243.1 Detailed Description

Global profiler state.

Definition in file [profiler\\_state.h](#).

## 6.244 profiler\_trace.h File Reference

Diagnostics for container sizes.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_base< \\_\\_object\\_info, \\_\\_stack\\_info >](#)  
*Base class for all trace producers.*
- struct [\\_\\_gnu\\_profile::\\_\\_warning\\_data](#)  
*Representation of a warning.*

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Defines

- #define [\\_GLIBCXX\\_IMPL\\_UNORDERED\\_MAP](#)

## Typedefs

- typedef std::std::vector< \_\_cost\_factor \* > [\\_\\_gnu\\_profile::\\_\\_cost\\_factor\\_vector](#)
- typedef std::std::unordered\_map< [std::string](#), [std::string](#) > [\\_\\_gnu\\_profile::\\_\\_env\\_t](#)
- typedef std::std::vector< \_\_warning\_data > [\\_\\_gnu\\_profile::\\_\\_warning\\_vector\\_t](#)

## Functions

- std::size\_t [\\_\\_gnu\\_profile::\\_\\_env\\_to\\_size\\_t](#) (const char \*\_\_env\_var, std::size\_t \_\_default\_value)
- \_\_cost\_factor\_vector \* & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_cost\\_factors](#) ()
- \_\_env\_t & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_env](#) ()
- \_\_gnu\_cxx::\_\_mutex & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_global\\_lock](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_list\\_iterate\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_list\\_resize\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_list\\_shift\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_map\\_erase\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_map\\_find\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_map\\_insert\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_map\\_iterate\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_umap\\_erase\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_umap\\_find\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_umap\\_insert\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_umap\\_iterate\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_vector\\_iterate\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_vector\\_resize\\_cost\\_factor](#) ()
- \_\_cost\_factor & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_vector\\_shift\\_cost\\_factor](#) ()
- \_\_trace\_hash\_func \* & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_S\\_hash\\_func](#) ()
- \_\_trace\_hashtable\_size \* & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_S\\_hashtable\\_size](#) ()
- \_\_trace\_list\_to\_slist \* & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_S\\_list\\_to\\_slist](#) ()
- \_\_trace\_list\_to\_vector \* & [\\_\\_gnu\\_profile::\\_\\_get\\_\\_S\\_list\\_to\\_vector](#) ()

- `__trace_map2umap * & __gnu_profile::__get__S_map2umap ()`
- `std::size_t & __gnu_profile::__get__S_max_mem ()`
- `std::size_t & __gnu_profile::__get__S_max_stack_depth ()`
- `std::size_t & __gnu_profile::__get__S_max_warn_count ()`
- `const char * & __gnu_profile::__get__S_trace_file_name ()`
- `__trace_vector_size * & __gnu_profile::__get__S_vector_size ()`
- `__trace_vector_to_list * & __gnu_profile::__get__S_vector_to_list ()`
- `int __gnu_profile::__log_magnitude (float __f)`
- `std::size_t __gnu_profile::__max_mem ()`
- `FILE * __gnu_profile::__open_output_file (const char *__extension)`
- `bool __gnu_profile::__profctx_init ()`
- `void __gnu_profile::__profctx_init_unconditional ()`
- `void __gnu_profile::__read_cost_factors ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__set_cost_factors ()`
- `void __gnu_profile::__set_max_mem ()`
- `void __gnu_profile::__set_max_stack_trace_depth ()`
- `void __gnu_profile::__set_max_warn_count ()`
- `void __gnu_profile::__set_trace_path ()`
- `std::size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__trace_hash_func_init ()`
- `void __gnu_profile::__trace_hash_func_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_hashtable_size_init ()`
- `void __gnu_profile::__trace_hashtable_size_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_list_to_slist_init ()`
- `void __gnu_profile::__trace_list_to_slist_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_list_to_vector_init ()`
- `void __gnu_profile::__trace_list_to_vector_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_map_to_unordered_map_init ()`
- `void __gnu_profile::__trace_map_to_unordered_map_report (FILE *__f, __warning_vector_t & __warnings)`
- `void __gnu_profile::__trace_vector_size_init ()`
- `void __gnu_profile::__trace_vector_size_report (FILE *, __warning_vector_t &)`
- `void __gnu_profile::__trace_vector_to_list_init ()`
- `void __gnu_profile::__trace_vector_to_list_report (FILE *, __warning_vector_t &)`
- `void __gnu_profile::__write_cost_factors ()`

### 6.244.1 Detailed Description

Diagnostics for container sizes. Data structures to represent profiling traces.

Definition in file [profiler\\_trace.h](#).

## 6.245 profiler\_vector\_size.h File Reference

Collection of vector size traces.

### Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size](#)  
*Hashtable size instrumentation trace producer.*

### Namespaces

- namespace [\\_\\_gnu\\_profile](#)

### Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_construct](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_destruct](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_report](#) (FILE \*, \_\_warning\_vector\_t &)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_size\\_resize](#) (const void \*, std::size\_t, std::size\_t)

### 6.245.1 Detailed Description

Collection of vector size traces.

Definition in file [profiler\\_vector\\_size.h](#).

## 6.246 profiler\_vector\_to\_list.h File Reference

diagnostics for vector to list.



## Classes

- class [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list](#)  
*Vector-to-list instrumentation producer.*
- class [\\_\\_gnu\\_profile::\\_\\_vector2list\\_info](#)  
*A vector-to-list instrumentation line in the object table.*
- class [\\_\\_gnu\\_profile::\\_\\_vector2list\\_stack\\_info](#)  
*A vector-to-list instrumentation line in the stack table.*

## Namespaces

- namespace [\\_\\_gnu\\_profile](#)

## Functions

- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_construct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_destruct](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_find](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_init](#) ()
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_insert](#) (const void \*, std::size\_t, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_invalid\\_operator](#) (const void \*)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_iterate](#) (const void \*, std::size\_t)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_report](#) (FILE \*, \_\_warning\_vector\_t &)
- void [\\_\\_gnu\\_profile::\\_\\_trace\\_vector\\_to\\_list\\_resize](#) (const void \*, std::size\_t, std::size\_t)

### 6.246.1 Detailed Description

diagnostics for vector to list.

Definition in file [profiler\\_vector\\_to\\_list.h](#).

## 6.247 queue File Reference

### 6.247.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [queue](#).

## 6.248 queue.h File Reference

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

### Classes

- class [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>](#)

*Double-ended queue of bounded size, allowing lock-free atomic access. [push\\_front\(\)](#) and [pop\\_front\(\)](#) must not be called concurrently to each other, while [pop\\_back\(\)](#) can be called concurrently at all times. [empty\(\)](#), [size\(\)](#), and [top\(\)](#) are intentionally not provided. Calling them would not make sense in a concurrent setting.*

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Defines

- [#define \\_GLIBCXX\\_VOLATILE](#)

#### 6.248.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [queue.h](#).

#### 6.248.2 Define Documentation

##### 6.248.2.1 #define \_GLIBCXX\_VOLATILE

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file [queue.h](#).

## 6.249 quicksort.h File Reference

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort\_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >  
void \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >  
std::iterator_traits< _RAIter >::difference_type \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`

#### 6.249.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [quicksort.h](#).

## 6.250 random File Reference

#### 6.250.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [random](#).

## 6.251 random.h File Reference

### Classes

- class [std::bernoulli\\_distribution](#)  
*A Bernoulli random number distribution.*
- struct [std::bernoulli\\_distribution::param\\_type](#)
- class [std::binomial\\_distribution<\\_IntType>](#)  
*A discrete binomial random number distribution.*
- struct [std::binomial\\_distribution<\\_IntType>::param\\_type](#)
- class [std::cauchy\\_distribution<\\_RealType>](#)  
*A [cauchy\\_distribution](#) random number distribution.*
- struct [std::cauchy\\_distribution<\\_RealType>::param\\_type](#)
- class [std::chi\\_squared\\_distribution<\\_RealType>](#)  
*A [chi\\_squared\\_distribution](#) random number distribution.*
- struct [std::chi\\_squared\\_distribution<\\_RealType>::param\\_type](#)
- class [std::discard\\_block\\_engine<\\_RandomNumberEngine, \\_\\_p, \\_\\_r>](#)
- class [std::discrete\\_distribution<\\_IntType>](#)  
*A [discrete\\_distribution](#) random number distribution.*
- struct [std::discrete\\_distribution<\\_IntType>::param\\_type](#)
- class [std::exponential\\_distribution<\\_RealType>](#)  
*An exponential continuous distribution for random numbers.*
- struct [std::exponential\\_distribution<\\_RealType>::param\\_type](#)
- class [std::extreme\\_value\\_distribution<\\_RealType>](#)  
*A [extreme\\_value\\_distribution](#) random number distribution.*
- struct [std::extreme\\_value\\_distribution<\\_RealType>::param\\_type](#)
- class [std::fisher\\_f\\_distribution<\\_RealType>](#)  
*A [fisher\\_f\\_distribution](#) random number distribution.*
- struct [std::fisher\\_f\\_distribution<\\_RealType>::param\\_type](#)
- class [std::gamma\\_distribution<\\_RealType>](#)  
*A gamma continuous distribution for random numbers.*
- struct [std::gamma\\_distribution<\\_RealType>::param\\_type](#)
- class [std::geometric\\_distribution<\\_IntType>](#)

*A discrete geometric random number distribution.*

- struct `std::geometric_distribution<_IntType>::param_type`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m>`  
*A model of a linear congruential random number generator.*

- class `std::lognormal_distribution<_RealType>`  
*A [lognormal\\_distribution](#) random number distribution.*

- struct `std::lognormal_distribution<_RealType>::param_type`
- class `std::negative_binomial_distribution<_IntType>`  
*A [negative\\_binomial\\_distribution](#) random number distribution.*

- struct `std::negative_binomial_distribution<_IntType>::param_type`
- class `std::normal_distribution<_RealType>`  
*A normal continuous distribution for random numbers.*

- struct `std::normal_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`  
*A [piecewise\\_constant\\_distribution](#) random number distribution.*

- struct `std::piecewise_constant_distribution<_RealType>::param_type`
- class `std::piecewise_linear_distribution<_RealType>`  
*A [piecewise\\_linear\\_distribution](#) random number distribution.*

- struct `std::piecewise_linear_distribution<_RealType>::param_type`
- class `std::poisson_distribution<_IntType>`  
*A discrete Poisson random number distribution.*

- struct `std::poisson_distribution<_IntType>::param_type`
- class `std::random_device`
- class `std::seed_seq`  
*The [seed\\_seq](#) class generates sequences of seeds for random number generators.*

- class `std::shuffle_order_engine<_RandomNumberEngine, __k>`  
*Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_w.*

- class `std::student_t_distribution<_RealType>`  
*A [student\\_t\\_distribution](#) random number distribution.*

- struct [std::student\\_t\\_distribution<\\_RealType>::param\\_type](#)
- class [std::uniform\\_int\\_distribution<\\_IntType>](#)  
*Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.*
- struct [std::uniform\\_int\\_distribution<\\_IntType>::param\\_type](#)
- class [std::uniform\\_real\\_distribution<\\_RealType>](#)  
*Uniform continuous distribution for random numbers.*
- struct [std::uniform\\_real\\_distribution<\\_RealType>::param\\_type](#)
- class [std::weibull\\_distribution<\\_RealType>](#)  
*A [weibull\\_distribution](#) random number distribution.*
- struct [std::weibull\\_distribution<\\_RealType>::param\\_type](#)

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

## Typedefs

- typedef minstd\_rand0 **std::default\_random\_engine**
- typedef shuffle\_order\_engine< minstd\_rand0, 256 > **std::knuth\_b**
- typedef linear\_congruential\_engine< uint\_fast32\_t, 48271UL, 0UL, 2147483647UL > [std::minstd\\_rand](#)
- typedef linear\_congruential\_engine< uint\_fast32\_t, 16807UL, 0UL, 2147483647UL > [std::minstd\\_rand0](#)
- typedef mersenne\_twister\_engine< uint\_fast32\_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [std::mt19937](#)
- typedef mersenne\_twister\_engine< uint\_fast64\_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > [std::mt19937\\_64](#)
- typedef discard\_block\_engine< ranlux24\_base, 223, 23 > **std::ranlux24**
- typedef subtract\_with\_carry\_engine< uint\_fast32\_t, 24, 10, 24 > **std::ranlux24\_base**
- typedef discard\_block\_engine< ranlux48\_base, 389, 11 > **std::ranlux48**
- typedef subtract\_with\_carry\_engine< uint\_fast64\_t, 48, 5, 12 > **std::ranlux48\_base**

## Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >  
_RealType std::generate\_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>  
bool std::operator!= (const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear\_congruential\_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>  
bool std::operator!= (const std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>  
bool std::operator!= (const std::shuffle\_order\_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle\_order\_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _IntType >  
bool std::operator!= (const std::geometric\_distribution< _IntType > &__d1, const std::geometric\_distribution< _IntType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::normal\_distribution< _RealType > &__d1, const std::normal\_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::lognormal\_distribution< _RealType > &__d1, const std::lognormal\_distribution< _RealType > &__d2)`
- `template<typename _IntType >  
bool std::operator!= (const std::negative\_binomial\_distribution< _IntType > &__d1, const std::negative\_binomial\_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool std::operator!= (const std::poisson\_distribution< _IntType > &__d1, const std::poisson\_distribution< _IntType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::gamma\_distribution< _RealType > &__d1, const std::gamma\_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool std::operator!= (const std::exponential\_distribution< _RealType > &__d1, const std::exponential\_distribution< _RealType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>  
bool std::operator!= (const std::discard\_block\_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard\_block\_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _IntType >  
bool std::operator!= (const std::uniform\_int\_distribution< _IntType > &__d1, const std::uniform\_int\_distribution< _IntType > &__d2)`

- `template<typename _RealType >`  
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1,`  
`const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1,`  
`const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1,`  
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1,`  
`const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,`  
`size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _`  
`_UIntType __f>`  
`bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, _`  
`__n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const`  
`std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,`  
`__s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool std::operator!= (const std::independent_bits_engine< _`  
`RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_`  
`bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1,`  
`const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const`  
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1,`  
`const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType >`  
`&__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType >`  
`&__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1,`  
`const std::uniform_real_distribution< _IntType > &__d2)`
- `bool std::operator!= (const std::bernoulli_distribution &__d1, const`  
`std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::binomial_distribution< _IntType > &__d1,`  
`const std::binomial_distribution< _IntType > &__d2)`



- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `bool std::operator== (const std::bernoulli_distribution & __d1, const std::bernoulli_distribution & __d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::exponential_distribution< _RealType > & __d1, const std::exponential_distribution< _RealType > & __d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::uniform_int_distribution< _IntType > & __d1, const std::uniform_int_distribution< _IntType > & __d2)`

- `template<typename _IntType >`  
`bool std::operator== (const std::geometric_distribution< _IntType > &__d1,`  
`const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::uniform_real_distribution< _IntType > &__d1,`  
`const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator== (const std::discrete_distribution< _IntType > &__d1, const`  
`std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::extreme_value_distribution< _RealType > &__d1,`  
`const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::weibull_distribution< _RealType > &__d1,`  
`const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::cauchy_distribution< _RealType > &__d1,`  
`const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::piecewise_linear_distribution< _RealType >`  
`&__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator== (const std::piecewise_constant_distribution< _RealType >`  
`&__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<`  
`_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

- `template<typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<  
_CharT, _Traits > &__is, std::bernoulli_distribution &__x)`

### 6.251.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [random.h](#).

## 6.252 random.tcc File Reference

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_detail](#)

### Functions

- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >  
_OutputIterator std::__detail::__transform (_InputIterator __first, _-  
InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >  
_RealType std::generate\_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT  
, typename _Traits >  
std::basic\_ostream< _CharT, _Traits > & std::operator<< (std::basic\_-  
ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _-  
UIntType, __a, __c, __m > &__lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,  
size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _-  
UIntType __f, typename _CharT, typename _Traits >  
std::basic\_ostream< _CharT, _Traits > & std::operator<< (std::basic\_-  
ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _-  
UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >  
&__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename  
_Traits >  
std::basic\_ostream< _CharT, _Traits > & std::operator<< (std::basic\_-  
ostream< _CharT, _Traits > &__os, const discard_block_engine< _-  
RandomNumberEngine, __p, __r > &__x)`

- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::bernoulli_distribution &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::cauchy_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::weibull_distribution< _RealType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`

- ```
ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType >
&__x)
```
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &, const std::extreme_value_distribution< _`
`RealType > &)`
 - `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType >`
`&__x)`
 - `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename`
`_Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _`
`UIntType, __w, __s, __r > &__x)`
 - `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &, const std::exponential_distribution< _RealType`
`> &)`
 - `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _`
`RealType > &__x)`
 - `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _`
`RealType > &__x)`
 - `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType`
`> &)`
 - `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _`
`RealType > &)`
 - `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType >`
`&__x)`
 - `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_`
`ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType`
`> &__x)`
 - `template<typename _RealType >`
`bool std::operator== (const std::normal_distribution< _RealType > &__d1,`
`const std::normal_distribution< _RealType > &__d2)`

- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::exponential_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::extreme_value_distribution< _RealType > &)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > & __is, lognormal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > & __is, chi_squared_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > & __is, fisher_f_distribution< _RealType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > & __is, negative_binomial_distribution< _IntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > & __is, poisson_distribution< _IntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > & __is, discrete_distribution< _IntType > & __x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename
_Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > & __is, subtract_with_carry_engine< _UIntType,
__w, __s, __r > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > &, std::geometric_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > &, std::weibull_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > & __is, piecewise_linear_distribution< _RealType
> & __x)`

- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream<
_CharT, _Traits > &, std::cauchy_distribution< _RealType > &)`

6.252.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [random.tcc](#).

6.253 random_number.h File Reference

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- class [__gnu_parallel::RandomNumber](#)
Random number generator, based on the Mersenne twister.

Namespaces

- namespace [__gnu_parallel](#)

6.253.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random_number.h](#).

6.254 random_shuffle.h File Reference

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct [__gnu_parallel::__DRandomShufflingGlobalData<_RAIter>](#)
Data known to every thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).
- struct [__gnu_parallel::__DRSSorterPU<_RAIter, _RandomNumberGenerator>](#)
Local data for a thread participating in [__gnu_parallel::__parallel_random_shuffle\(\)](#).

Namespaces

- namespace [__gnu_parallel](#)

Typedefs

- typedef unsigned short [__gnu_parallel::_BinIndex](#)

Functions

- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle](#) (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle_drs](#) (_RAIter __begin, _RAIter __end, typename std::iterator_traits<_RAIter>::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__parallel_random_shuffle_drs_pu](#) (_DRSSorterPU<_RAIter, _RandomNumberGenerator> *__pus)
- template<typename _RandomNumberGenerator>
int [__gnu_parallel::__random_number_pow2](#) (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Tp>
_Tp [__gnu_parallel::__round_up_to_pow2](#) (_Tp __x)
- template<typename _RAIter, typename _RandomNumberGenerator>
void [__gnu_parallel::__sequential_random_shuffle](#) (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)

6.254.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [random_shuffle.h](#).

6.255 range_access.h File Reference

Namespaces

- namespace [std](#)

Functions

- `template<class _Container >`
`auto std::begin (_Container &__cont)-> decltype(__cont.begin())`
- `template<class _Container >`
`auto std::begin (const _Container &__cont)-> decltype(__cont.begin())`
- `template<class _Tp, size_t _Nm>`
`_Tp * std::begin (_Tp(&__arr)[_Nm])`
- `template<class _Container >`
`auto std::end (const _Container &__cont)-> decltype(__cont.end())`
- `template<class _Container >`
`auto std::end (_Container &__cont)-> decltype(__cont.end())`
- `template<class _Tp, size_t _Nm>`
`_Tp * std::end (_Tp(&__arr)[_Nm])`

6.255.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [range_access.h](#).

6.256 ratio File Reference

Classes

- struct [std::ratio< _Num, _Den >](#)
Provides compile-time rational arithmetic.

- struct `std::ratio_add<_R1, _R2>`
ratio_add
- struct `std::ratio_divide<_R1, _R2>`
ratio_divide
- struct `std::ratio_equal<_R1, _R2>`
ratio_equal
- struct `std::ratio_multiply<_R1, _R2>`
ratio_multiply
- struct `std::ratio_not_equal<_R1, _R2>`
ratio_not_equal
- struct `std::ratio_subtract<_R1, _R2>`
ratio_subtract

Namespaces

- namespace `std`

6.256.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [ratio](#).

6.257 `rb_tree` File Reference

Classes

- struct `__gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc>`

Namespaces

- namespace `__gnu_cxx`

6.257.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rb_tree](#).

6.258 rc_string_base.h File Reference

Classes

- class [__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>](#)

Namespaces

- namespace [__gnu_cxx](#)

6.258.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [rc_string_base.h](#).

6.259 regex File Reference

6.259.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [regex](#).

6.260 regex.h File Reference

Classes

- class [std::basic_regex<_Ch_type, _Rx_traits>](#)
- class [std::match_results<_Bi_iter, _Allocator>](#)

The results of a match or search operation.

- class `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`
- struct `std::regex_traits< _Ch_type >`

Describes aspects of a regular expression.

- class `std::sub_match< _BiIter >`

Namespaces

- namespace `std`

Typedefs

- typedef `match_results< const char * >` **`std::cmatch`**
- typedef `regex_iterator< const char * >` **`std::cregex_iterator`**
- typedef `regex_token_iterator< const char * >` **`std::cregex_token_iterator`**
- typedef `sub_match< const char * >` **`std::csub_match`**
- typedef `basic_regex< char >` **`std::regex`**
- typedef `match_results< string::const_iterator >` **`std::smatch`**
- typedef `regex_iterator< string::const_iterator >` **`std::sregex_iterator`**
- typedef `regex_token_iterator< string::const_iterator >` **`std::sregex_token_iterator`**
- typedef `sub_match< string::const_iterator >` **`std::ssub_match`**
- typedef `match_results< const wchar_t * >` **`std::wcmatch`**
- typedef `regex_iterator< const wchar_t * >` **`std::wcregex_iterator`**
- typedef `regex_token_iterator< const wchar_t * >` **`std::wcregex_token_iterator`**
- typedef `sub_match< const wchar_t * >` **`std::wcs_sub_match`**
- typedef `basic_regex< wchar_t >` **`std::wregex`**
- typedef `match_results< wstring::const_iterator >` **`std::wsmatch`**
- typedef `regex_iterator< wstring::const_iterator >` **`std::wsregex_iterator`**
- typedef `regex_token_iterator< wstring::const_iterator >` **`std::wsregex_token_iterator`**
- typedef `sub_match< wstring::const_iterator >` **`std::wssub_match`**

Functions

- template<typename `_Bi_iter` >
const `sub_match< _Bi_iter >` & **`std::__unmatched_sub`** ()
- template<typename `_BiIter` >
bool **`std::operator!=`** (const `sub_match< _BiIter >` & `__lhs`, const `sub_match< _BiIter >` & `__rhs`)

- `template<typename _Bi_iter >`
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Allocator >`
`bool std::operator!= (const match_results< _Bi_iter, _Allocator > &__m1, const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _BiIter >`
`bool std::operator< (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator< (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`

- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream<`
`_Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const basic_-`
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_-`
`alloc > &__rhs)`
- `template<typename _Bilter >`
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match<`
`_Bilter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const`
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<= (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const basic_-`
`string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_-`
`alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const basic_string< typename iterator_traits< _Bi_iter`
`>::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter >`
`&__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`bool std::operator== (const match_results< _Bi_iter, _Allocator > &__m1,`
`const match_results< _Bi_iter, _Allocator > &__m2)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const`
`*__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_-`
`traits< _Bi_iter >::value_type const &__rhs)`

- `template<typename _BiIter >`
`bool std::operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator> (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`
`bool std::operator> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator>= (const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _BiIter >`
`bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc > &__rhs)`

- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename`
`iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const`
`&__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _`
`Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Allocator >`
`void std::swap (match_results< _Bi_iter, _Allocator > &__lhs, match_results<`
`_Bi_iter, _Allocator > &__rhs)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits`
`>`
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter,`
`_Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_`
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type _`
`__flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Allocator, typename _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type`
`*, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_`
`type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc`
`> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _`
`Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_`
`constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_`
`type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_`
`constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _`
`Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_`
`allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Allocator, typename _Ch_type, typename _Rx_traits`
`>`

```

bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, match_results< _-
Bi_iter, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits >
&__re, regex_constants::match_flag_type __flags=regex_constants::match_
default)
• template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >
bool std::regex\_search (_Bi_iter __first, _Bi_iter __last, const basic_
regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type _-
__flags=regex_constants::match_default)
• template<typename _Ch_type, class _Allocator, class _Rx_traits >
bool std::regex\_search (const _Ch_type *__s, match_results< const _Ch_
type *, _Allocator > &__m, const basic_regex< _Ch_type, _Rx_traits > &_-
__e, regex_constants::match_flag_type __f=regex_constants::match_default)
• template<typename _Ch_type, typename _Rx_traits >
bool std::regex\_search (const _Ch_type *__s, const basic_regex< _Ch_
type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
constants::match_default)
• template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename
_Rx_traits >
bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _String_
allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_
constants::match_flag_type __flags=regex_constants::match_default)
• template<typename _Ch_traits, typename _Ch_alloc, typename _Allocator, typename _Ch_
type, typename _Rx_traits >
bool std::regex\_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc
> &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _-
Ch_alloc >::const_iterator, _Allocator > &__m, const basic_regex< _-
Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_
constants::match_default)
• template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type
>
_Out_iter std::regex\_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter _-
last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string<
_Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_
constants::match_default)
• template<typename _Rx_traits, typename _Ch_type >
basic_string< _Ch_type > std::regex\_replace (const basic_string< _Ch_
type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const
basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type _-
flags=regex_constants::match_default)

```

6.260.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex.h](#).

6.261 regex_compiler.h File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _InIter, typename _TraitsT >
_AutomatonPtr std::regex::compile(const _InIter &__b, const _InIter &__e, _TraitsT &__t, regex_constants::syntax_option_type __f)`

6.261.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_compiler.h](#).

6.262 regex_constants.h File Reference

Constant definitions for the std regex library.

Namespaces

- namespace [std](#)
- namespace [std::regex_constants](#)

5.1 Regular Expression Syntax Options

- `enum std::regex_constants::__syntax_option {
 _S_icode, _S_nosubs, _S_optimize, _S_collate,
 _S_ECMAScript, _S_basic, _S_extended, _S_awk,
 _S_grep, _S_egrep, _S_syntax_last }`
- `typedef unsigned int std::regex_constants::syntax_option_type`
- `static const syntax_option_type std::regex_constants::icase`
- `static const syntax_option_type std::regex_constants::nosubs`
- `static const syntax_option_type std::regex_constants::optimize`
- `static const syntax_option_type std::regex_constants::collate`

- static const syntax_option_type [std::regex_constants::ECMAScript](#)
- static const syntax_option_type [std::regex_constants::basic](#)
- static const syntax_option_type [std::regex_constants::extended](#)
- static const syntax_option_type [std::regex_constants::awk](#)
- static const syntax_option_type [std::regex_constants::grep](#)
- static const syntax_option_type [std::regex_constants::egrep](#)

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum [std::regex_constants::__match_flag](#) {
[_S_not_bol](#), [_S_not_eol](#), [_S_not_bow](#), [_S_not_eow](#),
[_S_any](#), [_S_not_null](#), [_S_continuous](#), [_S_prev_avail](#),
[_S_sed](#), [_S_no_copy](#), [_S_first_only](#), [_S_match_flag_last](#) }
- typedef [std::bitset](#)< [_S_match_flag_last](#) > [std::regex_constants::match_flag_type](#)
- static const match_flag_type [std::regex_constants::match_default](#)
- static const match_flag_type [std::regex_constants::match_not_bol](#)
- static const match_flag_type [std::regex_constants::match_not_eol](#)
- static const match_flag_type [std::regex_constants::match_not_bow](#)
- static const match_flag_type [std::regex_constants::match_not_eow](#)
- static const match_flag_type [std::regex_constants::match_any](#)
- static const match_flag_type [std::regex_constants::match_not_null](#)
- static const match_flag_type [std::regex_constants::match_continuous](#)
- static const match_flag_type [std::regex_constants::match_prev_avail](#)
- static const match_flag_type [std::regex_constants::format_default](#)
- static const match_flag_type [std::regex_constants::format_sed](#)
- static const match_flag_type [std::regex_constants::format_no_copy](#)
- static const match_flag_type [std::regex_constants::format_first_only](#)

6.262.1 Detailed Description

Constant definitions for the std regex library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_constants.h](#).

6.263 `regex_cursor.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _FwdIterT > _SpecializedCursor< _FwdIterT > std::__regex::__cursor (const _FwdIterT &__b, const _FwdIterT __e)`

6.263.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_cursor.h](#).

6.264 `regex_error.h` File Reference

Error and exception objects for the std regex library.

Classes

- class [std::regex_error](#)
A regular expression exception class.
The regular expression library throws objects of this class on error.

Namespaces

- namespace [std](#)
- namespace [std::regex_constants](#)

Functions

- `void std::__throw_regex_error (regex_constants::error_type __ecode)`

5.3 Error Types

- enum `std::regex_constants::error_type` {
`_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`,
`_S_error_brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`,
`_S_error_range`, `_S_error_space`, `_S_error_badrepeat`, `_S_error_ -`
`complexity`,
`_S_error_stack`, `_S_error_last` }
- static const error_type `std::regex_constants::error_collate` (`_S_error_collate`)
- static const error_type `std::regex_constants::error_ctype` (`_S_error_ctype`)
- static const error_type `std::regex_constants::error_escape` (`_S_error_escape`)
- static const error_type `std::regex_constants::error_backref` (`_S_error_backref`)
- static const error_type `std::regex_constants::error_brack` (`_S_error_brack`)
- static const error_type `std::regex_constants::error_paren` (`_S_error_paren`)
- static const error_type `std::regex_constants::error_brace` (`_S_error_brace`)
- static const error_type `std::regex_constants::error_badbrace` (`_S_error_ -`
`badbrace`)
- static const error_type `std::regex_constants::error_range` (`_S_error_range`)
- static const error_type `std::regex_constants::error_space` (`_S_error_space`)
- static const error_type `std::regex_constants::error_badrepeat` (`_S_error_ -`
`badrepeat`)
- static const error_type `std::regex_constants::error_complexity` (`_S_error_ -`
`complexity`)
- static const error_type `std::regex_constants::error_stack` (`_S_error_stack`)

6.264.1 Detailed Description

Error and exception objects for the std regex library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_error.h](#).

6.265 regex_grep_matcher.h File Reference

Namespaces

- namespace `std`

Typedefs

- typedef `std::stack< _StateIdT, std::vector< _StateIdT > >` `std::__regex:: -`
`StateStack`

6.265.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_grep_matcher.h](#).

6.266 `regex_grep_matcher.tcc` File Reference

Namespaces

- namespace [std](#)

6.266.1 Detailed Description

Definition in file [regex_grep_matcher.tcc](#).

6.267 `regex_nfa.h` File Reference

Namespaces

- namespace [std](#)

Typedefs

- typedef [std::shared_ptr](#)< `_Automaton` > `std::__regex::_AutomatonPtr`
- typedef `std::function`< `bool`(const `_PatternCursor` &)> `std::__regex::_Matcher`
- typedef `int` `std::__regex::_StateIdT`
- typedef [std::set](#)< `_StateIdT` > `std::__regex::_StateSet`
- typedef `std::function`< `void`(const `_PatternCursor` &, `_Results` &)> `std::__regex::_Tagger`

Enumerations

- enum `_Opcode` {
`_S_opcode_unknown`, `_S_opcode_alternative`, `_S_opcode_subexpr_begin`,
`_S_opcode_subexpr_end`,
`_S_opcode_match`, `_S_opcode_accept` }

Functions

- `bool std::__regex::_AnyMatcher (const _PatternCursor &)`

Variables

- `static const _StateIdT std::__regex::_S_invalid_state_id`

6.267.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [regex_nfa.h](#).

6.268 `regex_nfa.tcc` File Reference

Namespaces

- namespace [std](#)

6.268.1 Detailed Description

Definition in file [regex_nfa.tcc](#).

6.269 `rope` File Reference

Classes

- class [__gnu_cxx::rope<_CharT, _Alloc>](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [__gnu_cxx::__detail](#)
- namespace [std](#)
- namespace [std::tr1](#)

Defines

- `#define __GC_CONST`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOCS(__a)`
- `#define __STATIC_IF_SGI_ALLOC`
- `#define __STL_FREE_STRING(__s, __l, __a)`
- `#define __STL_ROPE_FROM_UNOWNED_CHAR_PTR(__s, __size, __a)`

Typedefs

- `typedef rope< char > __gnu_cxx::crope`
- `typedef rope< wchar_t > __gnu_cxx::wrope`

Enumerations

- `enum { _S_max_rope_depth }`
- `enum _Tag { _S_leaf, _S_concat, _S_substringfn, _S_function }`

Functions

- `crope::reference __gnu_cxx::__mutable_reference_at (crope &__c, size_t __i)`
- `template<typename _ForwardIterator, typename _Tp >
void __gnu_cxx::__Destroy_const (_ForwardIterator __first, _ForwardIterator
__last, allocator< _Tp >)`
- `template<typename _ForwardIterator, typename _Allocator >
void __gnu_cxx::__Destroy_const (_ForwardIterator __first, _ForwardIterator
__last, _Allocator __alloc)`
- `template<class _CharT >
void __gnu_cxx::__S_cond_store_eos (_CharT &)`
- `void __gnu_cxx::__S_cond_store_eos (char &__c)`
- `void __gnu_cxx::__S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >
_CharT __gnu_cxx::__S_eos (_CharT *)`
- `bool __gnu_cxx::__S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >
bool __gnu_cxx::__S_is_basic_char_type (_CharT *)`
- `bool __gnu_cxx::__S_is_basic_char_type (char *)`
- `template<class _CharT >
bool __gnu_cxx::__S_is_one_byte_char_type (_CharT *)`

- `bool __gnu_cxx::S_is_one_byte_char_type (char *)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const rope< _CharT, _Alloc > &__x, const`
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc`
`> &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator!= (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _Rope_-`
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __n, const`
`_Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`
`> &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`
`> &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc`
`> &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (const _-`
`Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (ptrdiff_t __-`
`n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`
`&__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`
`&__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc >`
`&__left, _CharT __right)`

- `template<class _CharT, class _Alloc >`
`ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc`
`> &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_`
`iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &_`
`__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _`
`Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const`
`rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<<`
`(std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc >`
`&__r)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const`
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc`
`> &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const`
`rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const`
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc >`
`&__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x,`
`const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const`
`rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _-`
`Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _CharT, class _Alloc >`
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc`
`> &__y)`

Variables

- `rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn<`
`_CharT, _Alloc >)`

6.269.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [rope](#).

6.270 ropeimpl.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Functions

- `template<class _CharT, class _Traits >`
`void __gnu_cxx::_Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`
`bool __gnu_cxx::_Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::_Rope_is_simple (wchar_t *)`
- `bool __gnu_cxx::_Rope_is_simple (char *)`
- `template<class _Rope_iterator >`
`void __gnu_cxx::_Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

6.270.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [ropeimpl.h](#).

6.271 safe_base.h File Reference

Classes

- class [__gnu_debug::_Safe_iterator_base](#)
Basic functionality for a safe iterator.
- class [__gnu_debug::_Safe_sequence_base](#)

Base class that supports tracking of iterators that reference a sequence.

Namespaces

- namespace [__gnu_debug](#)

6.271.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_base.h](#).

6.272 [safe_iterator.h](#) File Reference

Classes

- struct [__gnu_debug::_BeforeBeginHelper<_Sequence>](#)
- class [__gnu_debug::_Safe_iterator<_Iterator, _Sequence>](#)

Safe iterator wrapper.

Namespaces

- namespace [__gnu_debug](#)

Functions

- `template<typename _Iterator>
_Siter_base<_Iterator>::iterator_type __gnu_debug::__base (_Iterator __it)`
- `bool __gnu_debug::__check_singular_aux (const _Safe_iterator_base *__x)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>
bool __gnu_debug::operator!= (const _Safe_iterator<_IteratorL, _Sequence> &__lhs, const _Safe_iterator<_IteratorR, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence>
bool __gnu_debug::operator!= (const _Safe_iterator<_Iterator, _Sequence> &__lhs, const _Safe_iterator<_Iterator, _Sequence> &__rhs)`
- `template<typename _Iterator, typename _Sequence>
_Safe_iterator<_Iterator, _Sequence> __gnu_debug::operator+ (typename _Safe_iterator<_Iterator, _Sequence>::difference_type __n, const _Safe_iterator<_Iterator, _Sequence> &__i)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`_Safe_iterator< _IteratorL, _Sequence >::difference_type __gnu_`
`debug::operator- (const _Safe_iterator< _IteratorL, _Sequence > &__lhs,`
`const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`_Safe_iterator< _Iterator, _Sequence >::difference_type __gnu_`
`debug::operator- (const _Safe_iterator< _Iterator, _Sequence > &__lhs,`
`const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator< (const _Safe_iterator< _Iterator, _Sequence >`
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator< (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator<= (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator<= (const _Safe_iterator< _Iterator, _Sequence`
`> &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_iterator< _Iterator, _Sequence >`
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator> (const _Safe_iterator< _Iterator, _Sequence >`
`&__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator> (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator>= (const _Safe_iterator< _IteratorL, _Sequence`
`> &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator>= (const _Safe_iterator< _Iterator, _Sequence`
`> &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)`

6.272.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_iterator.h](#).

6.273 `safe_iterator.tcc` File Reference

Namespaces

- namespace `__gnu_debug`

6.273.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_iterator.tcc](#).

6.274 `safe_sequence.h` File Reference

Classes

- class `__gnu_debug::_After_nth_from<_Iterator>`
- class `__gnu_debug::_Not_equal_to<_Type>`
- class `__gnu_debug::_Safe_sequence<_Sequence>`

Base class for constructing a safe sequence type that tracks iterators that reference it.

Namespaces

- namespace `__gnu_debug`

6.274.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [safe_sequence.h](#).

6.275 `search.h` File Reference

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace `__gnu_parallel`

Functions

- `template<typename _RAIter, typename _DifferenceTp >`
`void __gnu_parallel::__calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred >`
`__RAIter1 __gnu_parallel::__search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)`

6.275.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [search.h](#).

6.276 set File Reference

6.276.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [set](#).

6.277 set File Reference

6.277.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set](#).

6.278 set File Reference

6.278.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set](#).

6.279 set.h File Reference

Classes

- class [std::__debug::set< _Key, _Compare, _Allocator >](#)

Class [std::set](#) with safety/checking/debug instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__debug](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__debug::operator!= (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__debug::operator< (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__debug::operator<= (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__debug::operator== (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__debug::operator> (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator > &_
_lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _
Key, _Compare, _Allocator > &__y)`

6.279.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/set.h](#).

6.280 set.h File Reference

Classes

- class [std::__profile::set< _Key, _Compare, _Allocator >](#)
Class [std::set](#) wrapper with performance instrumentation.

Namespaces

- namespace [std](#)
- namespace [std::__profile](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator!= (const set< _Key, _Compare, _Allocator >
&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator< (const set< _Key, _Compare, _Allocator > &_
__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator<= (const set< _Key, _Compare, _Allocator >
&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator== (const set< _Key, _Compare, _Allocator >
&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator> (const set< _Key, _Compare, _Allocator > &_
__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool std::__profile::operator>= (const set< _Key, _Compare, _Allocator >
&__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
void std::__profile::swap (set< _Key, _Compare, _Allocator > &__x, set< _
Key, _Compare, _Allocator > &__y)`

6.280.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/set.h](#).

6.281 set_operations.h File Reference

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<typename _Iter, typename _OutputIterator >
_OutputIterator __gnu_parallel::__copy_tail (std::pair< _Iter, _Iter > __b,
std::pair< _Iter, _Iter > __e, _OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __gnu_parallel::__parallel_set_difference (_Iter __begin1,
_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _-
Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __gnu_parallel::__parallel_set_intersection (_Iter __begin1,
_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _-
Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >
_OutputIterator __gnu_parallel::__parallel_set_operation (_Iter __begin1,
_Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _-
Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __gnu_parallel::__parallel_set_symmetric_difference (_Iter
__begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __-
result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >
_OutputIterator __gnu_parallel::__parallel_set_union (_Iter __begin1, _Iter
__end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __-
comp)`

6.281.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [set_operations.h](#).

6.282 settings.h File Reference

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

Classes

- struct `__gnu_parallel::_Settings`
class `_Settings` /// Run-time settings for the parallel mode including all tunable parameters.

Namespaces

- namespace `__gnu_parallel`

Defines

- `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

6.282.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

6.282.2 parallelization_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and __off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. It is a tristate variable corresponding to:

a. `force_sequential`, meaning the sequential algorithm is executed. b. `force_parallel`, meaning the parallel algorithm is executed. c. `heuristic`

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

Definition in file [settings.h](#).

6.282.3 Define Documentation

6.282.3.1 `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

Determine at compile(?) -time if the parallel variant of an algorithm should be called.

Parameters

`__c` A condition that is convertible to bool that is overruled by `__gnu_parallel::_Settings::algorithm_strategy`. Usually a decision based on the input size.

Definition at line 95 of file [settings.h](#).

6.283 `shared_ptr.h` File Reference

Classes

- class [std::enable_shared_from_this<_Tp>](#)
Base class allowing use of member function `shared_from_this`.
- struct [std::hash<shared_ptr<_Tp>>](#)
`std::hash` specialization for `shared_ptr`.

- struct `std::owner_less< shared_ptr< _Tp > >`
Partial specialization of `owner_less` for `shared_ptr`.
- struct `std::owner_less< weak_ptr< _Tp > >`
Partial specialization of `owner_less` for `weak_ptr`.
- class `std::shared_ptr< _Tp >`
A smart pointer with reference-counted copy semantics.
- class `std::weak_ptr< _Tp >`
A smart pointer with weak semantics.

Namespaces

- namespace `std`

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
`shared_ptr< _Tp > std::allocate_shared (_Alloc __a, _Args &&...__args)`
- template<typename _Tp, typename _Tp1 >
`shared_ptr< _Tp > std::const_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- template<typename _Tp, typename _Tp1 >
`shared_ptr< _Tp > std::dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- template<typename _Del, typename _Tp, _Lock_policy _Lp>
`_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p)`
- template<typename _Tp, typename... _Args>
`shared_ptr< _Tp > std::make_shared (_Args &&...__args)`
- template<typename _Tp >
`bool std::operator!= (nullptr_t, const shared_ptr< _Tp > &__b)`
- template<typename _Tp1, typename _Tp2 >
`bool std::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`
- template<typename _Tp >
`bool std::operator!= (const shared_ptr< _Tp > &__a, nullptr_t)`
- template<typename _Tp1, typename _Tp2 >
`bool std::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b)`

- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch,`
`_Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp >`
`bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t)`
- `template<typename _Tp >`
`bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__b)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _-`
`Tp2 > &__b)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r)`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b)`
- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b)`

6.283.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [shared_ptr.h](#).

6.284 shared_ptr_base.h File Reference

Classes

- struct [std::hash< __shared_ptr< _Tp, _Lp > >](#)
std::hash specialization for __shared_ptr.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp, _Lock_policy _Lp, typename _Alloc, typename... _Args>`
`__shared_ptr< _Tp, _Lp > std::__allocate_shared (_Alloc __a, _Args &&... _-`
`_args)`

- `template<_Lock_policy _Lp, typename _Tp1, typename _Tp2 >`
`void std::enable_shared_from_this_helper (const __shared_count< _Lp >`
`&, const __enable_shared_from_this< _Tp1, _Lp > *, const _Tp2 *)`
- `template<typename _Tp1, typename _Tp2 >`
`void std::enable_shared_from_this_helper (const __shared_count<> &,`
`const enable_shared_from_this< _Tp1 > *, const _Tp2 *)`
- `template<_Lock_policy _Lp>`
`void std::enable_shared_from_this_helper (const __shared_count< _Lp >`
`&,...)`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args>`
`__shared_ptr< _Tp, _Lp > std::make_shared (_Args &&...__args)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1,`
`_Lp > &__r)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _`
`_Tp1, _Lp > &__r)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __-`
`shared_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator< (const __shared_ptr< _Tp1, _Lp > &__a, const __-`
`shared_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator<= (const __shared_ptr< _Tp, _Lp > &__a, const __-`
`shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __-`
`shared_ptr< _Tp2, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator> (const __shared_ptr< _Tp, _Lp > &__a, const __shared_`
`ptr< _Tp, _Lp > &__b)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator>= (const __shared_ptr< _Tp, _Lp > &__a, const __-`
`shared_ptr< _Tp, _Lp > &__b)`

- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1,`
`_Lp > &__r)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void std::swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp >`
`&__b)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void std::swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp >`
`&__b)`

6.284.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [shared_ptr_base.h](#).

6.285 `slice_array.h` File Reference

Classes

- class [std::slice](#)
Class defining one-dimensional subset of an array.
- class [std::slice_array< _Tp >](#)
Reference to one-dimensional subset of an array.

Namespaces

- namespace [std](#)

Defines

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

6.285.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [slice_array.h](#).

6.286 slist File Reference

Classes

- class [__gnu_cxx::slist<_Tp, _Alloc>](#)

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Functions

- [_Slist_node_base * __gnu_cxx::__slist_make_link](#) ([_Slist_node_base * __prev_node](#), [_Slist_node_base * __new_node](#))
- [_Slist_node_base * __gnu_cxx::__slist_previous](#) ([_Slist_node_base * __head](#), [const _Slist_node_base * __node](#))
- [const _Slist_node_base * __gnu_cxx::__slist_previous](#) ([const _Slist_node_base * __head](#), [const _Slist_node_base * __node](#))
- [_Slist_node_base * __gnu_cxx::__slist_reverse](#) ([_Slist_node_base * __node](#))
- [size_t __gnu_cxx::__slist_size](#) ([_Slist_node_base * __node](#))
- [void __gnu_cxx::__slist_splice_after](#) ([_Slist_node_base * __pos](#), [_Slist_node_base * __before_first](#), [_Slist_node_base * __before_last](#))
- [void __gnu_cxx::__slist_splice_after](#) ([_Slist_node_base * __pos](#), [_Slist_node_base * __head](#))
- [template<class _Tp, class _Alloc> bool __gnu_cxx::operator!=](#) ([const slist<_Tp, _Alloc> &_SL1](#), [const slist<_Tp, _Alloc> &_SL2](#))
- [template<class _Tp, class _Alloc> bool __gnu_cxx::operator<](#) ([const slist<_Tp, _Alloc> &_SL1](#), [const slist<_Tp, _Alloc> &_SL2](#))
- [template<class _Tp, class _Alloc> bool __gnu_cxx::operator<=](#) ([const slist<_Tp, _Alloc> &_SL1](#), [const slist<_Tp, _Alloc> &_SL2](#))
- [template<class _Tp, class _Alloc> bool __gnu_cxx::operator==](#) ([const slist<_Tp, _Alloc> &_SL1](#), [const slist<_Tp, _Alloc> &_SL2](#))
- [template<class _Tp, class _Alloc> bool __gnu_cxx::operator>](#) ([const slist<_Tp, _Alloc> &_SL1](#), [const slist<_Tp, _Alloc> &_SL2](#))
- [template<class _Tp, class _Alloc> bool __gnu_cxx::operator>=](#) ([const slist<_Tp, _Alloc> &_SL1](#), [const slist<_Tp, _Alloc> &_SL2](#))

- `template<class _Tp, class _Alloc >`
`void __gnu_cxx::swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &_`
`__y)`

6.286.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file [slist](#).

6.287 sort.h File Reference

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Namespaces

- namespace [__gnu_parallel](#)

Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, _Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _-`
`Compare __comp, multiway_mergesort_sampling_tag __parallelism)`

- template<bool __stable, typename _RAIter, typename _Compare >
void [__gnu_parallel::__parallel_sort](#) (_RAIter __begin, _RAIter __end, _-
Compare __comp, multiway_mergesort_exact_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare >
void [__gnu_parallel::__parallel_sort](#) (_RAIter __begin, _RAIter __end, _-
Compare __comp, multiway_mergesort_tag __parallelism)

6.287.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [sort.h](#).

6.288 sso_string_base.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

6.288.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [sso_string_base.h](#).

6.289 sstream File Reference

Classes

- class [std::basic_istream<_CharT, _Traits, _Alloc>](#)
Controlling input for std::string.
*This class supports reading from objects of type [std::basic_string](#), using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as *sb*.*
- class [std::basic_ostream<_CharT, _Traits, _Alloc>](#)
Controlling output for std::string.
*This class supports writing to objects of type [std::basic_string](#), using the inherited functions from [std::basic_ostream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as *sb*.*

- class [std::basic_stringbuf<_CharT, _Traits, _Alloc>](#)

The actual work of input and output (for [std::string](#)).

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a [std::basic_string](#). (Paraphrased from [27.7.1]/1.).

- class [std::basic_stringstream<_CharT, _Traits, _Alloc>](#)

Controlling input and output for [std::string](#).

*This class supports reading from and writing to objects of type [std::basic_string](#), using the inherited functions from [std::basic_istream](#). To control the associated sequence, an instance of [std::basic_stringbuf](#) is used, which this page refers to as *sb*.*

Namespaces

- namespace [std](#)

6.289.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [sstream](#).

6.290 sstream.tcc File Reference

Namespaces

- namespace [std](#)

6.290.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [sstream.tcc](#).

6.291 stack File Reference

6.291.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stack](#).

6.292 standard_policies.hpp File Reference

Namespaces

- namespace [__gnu_pbds](#)

Enumerations

- enum { [default_store_hash](#) }

6.292.1 Detailed Description

Contains standard policies for containers.

Definition in file [standard_policies.hpp](#).

6.293 stdatomic.h File Reference

6.293.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdatomic.h](#).

6.294 stdexcept File Reference

Classes

- class [std::domain_error](#)
- class [std::invalid_argument](#)
- class [std::length_error](#)
- class [std::logic_error](#)

One of two subclasses of exception.

- class [std::out_of_range](#)
- class [std::overflow_error](#)
- class [std::range_error](#)
- class [std::runtime_error](#)

One of two subclasses of exception.

- class [std::underflow_error](#)

Namespaces

- namespace [std](#)

6.294.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [stdexcept](#).

6.295 stdio_filebuf.h File Reference

Classes

- class [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#)

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE's and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., [stdio_filebuf<char>](#).*

Namespaces

- namespace [__gnu_cxx](#)

6.295.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [stdio_filebuf.h](#).

6.296 `stdio_sync_filebuf.h` File Reference

Classes

- class `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`
*Provides a layer of compatibility for C.
This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.*

Namespaces

- namespace `__gnu_cxx`

6.296.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file `stdio_sync_filebuf.h`.

6.297 `std_algo.h` File Reference

Namespaces

- namespace `std`

Enumerations

- enum { `_S_threshold` }
- enum { `_S_chunk_size` }

Functions

- `template<typename _RandomAccessIterator, typename _Distance>`
`void std::__chunk_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Distance __chunk_size)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare>`
`void std::__chunk_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)`

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _-`
`OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_RandomAccessIterator __first, _Size __n, _-`
`OutputIterator __result, random_access_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__final_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__final_insertion_sort (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Tp >`
`_RandomAccessIterator std::find (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, const _Tp &__val, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp`
`&__val, input_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _`
`BinaryPredicate >`
`_BidirectionalIterator1 std::find_end (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`
`_BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_-`
`iterator_tag, _BinaryPredicate __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _-`
`ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2`
`__last2, forward_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _-`
`ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2`
`__last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2 >`
`_BidirectionalIterator1 std::find_end (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`
`_BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_-`
`iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`
`_RandomAccessIterator std::find_if (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`

- `template<typename _RandomAccessIterator, typename _Predicate >`
`_RandomAccessIterator std::__find_if_not` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __last`, `_Predicate __pred`, `random_access_iterator_tag`)
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::__find_if_not` (`_InputIterator __first`, `_InputIterator __last`, `_`
`Predicate __pred`, `input_iterator_tag`)
- `template<typename _EuclideanRingElement >`
`_EuclideanRingElement std::__gcd` (`_EuclideanRingElement __m`, `_`
`EuclideanRingElement __n`)
- `template<typename _RandomAccessIterator >`
`void std::__heap_select` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __middle`, `_RandomAccessIterator __last`)
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__heap_select` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __middle`, `_RandomAccessIterator __last`, `_Compare`
`__comp`)
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`
`_ForwardIterator std::__inplace_stable_partition` (`_ForwardIterator __first`, `_`
`ForwardIterator __last`, `_Predicate __pred`, `_Distance __len`)
- `template<typename _RandomAccessIterator >`
`void std::__inplace_stable_sort` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __last`)
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__inplace_stable_sort` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __last`, `_Compare __comp`)
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__insertion_sort` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __last`, `_Compare __comp`)
- `template<typename _RandomAccessIterator >`
`void std::__insertion_sort` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __last`)
- `template<typename _RandomAccessIterator, typename _Size >`
`void std::__introsselect` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __nth`, `_RandomAccessIterator __last`, `_Size __`
`depth_limit`)
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void std::__introsselect` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __nth`, `_RandomAccessIterator __last`, `_Size __`
`depth_limit`, `_Compare __comp`)
- `template<typename _RandomAccessIterator, typename _Size >`
`void std::__introsort_loop` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __last`, `_Size __depth_limit`)
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void std::__introsort_loop` (`_RandomAccessIterator __first`, `_`
`RandomAccessIterator __last`, `_Size __depth_limit`, `_Compare __comp`)

- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer >`
`void std::__merge_adaptive (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename`
`_Compare >`
`void std::__merge_adaptive (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1,`
`_Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare`
`__comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`BidirectionalIterator3 >`
`_BidirectionalIterator3 std::__merge_backward (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`BidirectionalIterator3, typename _Compare >`
`_BidirectionalIterator3 std::__merge_backward (_BidirectionalIterator1 __-`
`first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _-`
`BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __-`
`comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename`
`_Distance >`
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-`
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`
`__step_size)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename`
`_Distance, typename _Compare >`
`void std::__merge_sort_loop (_RandomAccessIterator1 __first, _-`
`RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance`
`__step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer >`
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`
`void std::__merge_sort_with_buffer (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance >`
`void std::__merge_without_buffer (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`
`void std::__merge_without_buffer (_BidirectionalIterator __first, _-`
`BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance`
`__len1, _Distance __len2, _Compare __comp)`

- `template<typename _Iterator, typename _Compare >`
`void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _-`
`Compare __comp)`
- `template<typename _Iterator >`
`void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator _-`
`__last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`
`_BidirectionalIterator std::__partition (_BidirectionalIterator __first, _-`
`BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last,`
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _-`
`ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __-`
`middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __-`
`middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _-`
`Distance >`
`_BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _-`
`BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __-`
`len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_-`
`size)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator _-`
`__last, _Integer __count, const _Tp &__val, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _Tp >`
`_RandomAccessIter std::__search_n (_RandomAccessIter __first, _-`
`RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random_-`
`access_iterator_tag)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _-`
`BinaryPredicate >`
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator _-`
`__last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred,`
`std::forward_iterator_tag)`

- `template<typename _RandomAccessIter , typename _Integer , typename _Tp , typename _BinaryPredicate >`
`_RandomAccessIter std::__search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag)`
- `template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _Distance >`
`_ForwardIterator std::__stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Distance >`
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator , typename _Pointer , typename _Distance , typename _Compare >`
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::__unguarded_linear_insert (_RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`void std::__unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator , typename _Tp >`
`_RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)`
- `template<typename _RandomAccessIterator , typename _Tp , typename _Compare >`
`_RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`_RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator , typename _OutputIterator >`
`_OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)`

- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::__unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator std::__unique_copy (_InputIterator __first, _InputIterator __-`
`last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator`
`__last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_`
`iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::__unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag,`
`output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::__unique_copy (_InputIterator __first, _InputIterator _-`
`__last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_`
`iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator`
`__last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator`
`__result)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __-`
`first, _InputIterator __last, const _Tp &__value)`

- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator`
`__first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator`
`__first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator`
`__first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp`
`&__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1`
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1`
`__last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _-`
`BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1,`
`_ForwardIterator __first2, _ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1,`
`_ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _-`
`Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function`
`__f)`
- `template<typename _ForwardIterator, typename _Generator >`
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator`
`__gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator`
`__gen)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator _-`
`__middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator _-`
`__middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate`
`__pred)`
- `template<typename _ForwardIterator >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare`
`__comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator`
`__last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator`
`__last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _-`
`InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`

- `template<typename _Tp, typename _Compare >`
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __-`
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`
`RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __-`
`first, _InputIterator __last, _RandomAccessIterator __result_first, _-`
`RandomAccessIterator __result_last)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, type-`
`name _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator`
`__first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __-`
`out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator`
`__last, _Predicate __pred)`
- `template<typename _BidirectionalIterator >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::random_shuffle (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __-`
`last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __-`
`last, _Predicate __pred)`

- `template<typename _ForwardIterator, typename _Tp >`
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp`
`&__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::replace_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _`
`Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __`
`last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _`
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator std::reverse_copy (_BidirectionalIterator __first, _`
`BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _`
`ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __`
`middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate`
`>`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __`
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate`
`__predicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __`
`last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _`
`BinaryPredicate >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __`
`last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __`
`last, _Integer __count, const _Tp &__val)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result,`
`_Compare __comp)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __-`
`last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1`
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`
`result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1`
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __-`
`result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`
`OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _-`
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`
`OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _Compare >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _-`
`Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _RandomAccessIterator >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last,`
`_Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _-`
`ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`

- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _-`
`OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, type-`
`name _BinaryOperation >`
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_-`
`op)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last,`
`_BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last,`
`_OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator`
`__last, const _Tp &__val, _Compare __comp)`

6.297.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_algo.h](#).

6.298 `stl_algo.h` File Reference

Namespaces

- namespace [std](#)

Defines

- `#define _GLIBCXX_MOVE3(_Tp, _Up, _Vp)`
- `#define _GLIBCXX_MOVE_BACKWARD3(_Tp, _Up, _Vp)`

Functions

- `template<bool _IsMove, typename _II, typename _OI >
_OI std::__copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type std::__copy_move_a2
(_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type std::__copy_move_a2
(const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT >
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT > >
>, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >
_OI std::__copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >
_BI2 std::__copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >
_BI2 std::__copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _II1, typename _II2 >
bool std::__equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >
__gnu_cxx::__enable_if< !__is_scalar< _Tp >::__value, void >::__type std::__fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp >
__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type std::__fill_a (_Tp *__first, _Tp *__last, const _Tp &__c)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >
__gnu_cxx::__enable_if< !__is_scalar< _Tp >::__value, _OutputIterator >::__type std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Size, typename _Tp >
__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, _Tp * >::__type std::__fill_n_a (_Tp *__first, _Size __n, const _Tp &__c)`
- `template<typename _II1, typename _II2 >
bool std::__lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`

- long long **std::__lg** (long long __n)
- long **std::__lg** (long __n)
- template<typename _Size >
_Size **std::__lg** (_Size __n)
- int **std::__lg** (int __n)
- template<typename _Iterator >
_Miter_base< _Iterator >::iterator_type **std::__miter_base** (_Iterator __it)
- template<typename _Iterator >
_Niter_base< _Iterator >::iterator_type **std::__niter_base** (_Iterator __it)
- template<typename _II, typename _OI >
_OI **std::copy** (_II __first, _II __last, _OI __result)
- template<typename _BI1, typename _BI2 >
_BI2 **std::copy_backward** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _II1, typename _II2 >
bool **std::equal** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >
bool **std::equal** (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _ForwardIterator, typename _Tp >
void **std::fill** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _OI, typename _Size, typename _Tp >
_OI **std::fill_n** (_OI __first, _Size __n, const _Tp &__value)
- template<typename _ForwardIterator1, typename _ForwardIterator2 >
void **std::iter_swap** (_ForwardIterator1 __a, _ForwardIterator2 __b)
- template<typename _II1, typename _II2, typename _Compare >
bool **std::lexicographical_compare** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _II1, typename _II2 >
bool **std::lexicographical_compare** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _ForwardIterator, typename _Tp >
_ForwardIterator **std::lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _Tp, typename _Compare >
const _Tp & **std::max** (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
const _Tp & **std::max** (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare >
const _Tp & **std::min** (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
const _Tp & **std::min** (const _Tp &__a, const _Tp &__b)

- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_`
`pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1,`
`_InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _II, typename _OI >`
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _`
`ForwardIterator1 __last1, _ForwardIterator2 __first2)`

6.298.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_algobase.h](#).

6.299 `std_bvector.h` File Reference

Classes

- struct [std::hash<::vector< bool, _Alloc > >](#)
[std::hash](#) specialization for `vector<bool>`.
- class [std::vector< bool, _Alloc >](#)
A specialization of `vector` for booleans which offers fixed time access to individual elements in any order.

Namespaces

- namespace [std](#)

Typedefs

- typedef unsigned long [std::_Bit_type](#)

Enumerations

- enum { **_S_word_bit** }

Functions

- void **std::fill_bvector** (_Bit_iterator __first, _Bit_iterator __last, bool __x)
- void **std::fill** (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)
- _Bit_iterator **std::operator+** (ptrdiff_t __n, const _Bit_iterator &__x)
- _Bit_const_iterator **std::operator+** (ptrdiff_t __n, const _Bit_const_iterator &__x)
- ptrdiff_t **std::operator-** (const _Bit_iterator_base &__x, const _Bit_iterator_base &__y)

6.299.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_bvector.h](#).

6.300 stl_construct.h File Reference

Namespaces

- namespace [std](#)

Functions

- template<typename _T1, typename... _Args>
void [std::_Construct](#) (_T1 *__p, _Args &&...__args)
- template<typename _ForwardIterator, typename _Tp>
void **std::Destroy** (_ForwardIterator __first, _ForwardIterator __last, allocator<_Tp> &)
- template<typename _ForwardIterator, typename _Allocator>
void **std::Destroy** (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)
- template<typename _ForwardIterator>
void [std::_Destroy](#) (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _Tp>
void [std::_Destroy](#) (_Tp *__pointer)

6.300.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_construct.h](#).

6.301 `std_deque.h` File Reference

Classes

- class [std::_Deque_base< _Tp, _Alloc >](#)
- struct [std::_Deque_iterator< _Tp, _Ref, _Ptr >](#)
A deque::iterator.
- class [std::deque< _Tp, _Alloc >](#)
A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Namespaces

- namespace [std](#)

Defines

- `#define` [_GLIBCXX_DEQUE_BUF_SIZE](#)

Functions

- `size_t` [std::__deque_buf_size](#) (`size_t __size`)
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * >` [std::copy](#) (`_Deque_iterator< _Tp, const _Tp &, const _Tp * > __first`, `_Deque_iterator< _Tp, const _Tp &, const _Tp * > __last`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __result`)
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * >` [std::copy](#) (`_Deque_iterator< _Tp, _Tp &, _Tp * > __first`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __last`, `_Deque_iterator< _Tp, _Tp &, _Tp * > __result`)

- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_`
`iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp`
`* > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _`
`Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _Tp &__value)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp,`
`const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const _Tp &, const`
`_Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp, _Tp`
`&, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_`
`iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`
`iterator< _Tp, const _Tp &, const _Tp * > __first, _Deque_iterator< _Tp, const`
`_Tp &, const _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_`
`iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp`
`* > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool std::operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr > std::operator+ (ptrdiff_t __n, const _`
`Deque_iterator< _Tp, _Ref, _Ptr > &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type std::operator- (const _`
`Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref,`
`_Ptr > &__y)`

- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`
`>`
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type std::operator- (const`
`_Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _`
`RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`
`>`
`bool std::operator< (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`
`>`
`bool std::operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`
`_Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`
`_Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`
`>`
`bool std::operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`
`_Alloc > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR>`
`>`
`bool std::operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const`
`_Deque_iterator< _Tp, _Ref, _Ptr > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR`
`>`
`bool std::operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x,`
`const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp,`
`_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`

6.301.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_deque.h](#).

6.301.2 Define Documentation

6.301.2.1 `#define _GLIBCXX_DEQUE_BUF_SIZE`

This function controls the size of memory nodes.

Parameters

size The size of an element.

Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 82 of file `stl_deque.h`.

6.302 std_function.h File Reference

Classes

- struct `std::binary_function< _Arg1, _Arg2, _Result >`
- class `std::binary_negate< _Predicate >`
One of the [negation functors](#).
- class `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun_ref_t< _Ret, _Tp >`
One of the [adaptors for member /// pointers](#).
- class `std::const_mem_fun_t< _Ret, _Tp >`
One of the [adaptors for member /// pointers](#).
- struct `std::divides< _Tp >`
One of the [math functors](#).
- struct `std::equal_to< _Tp >`
One of the [comparison functors](#).
- struct `std::greater< _Tp >`
One of the [comparison functors](#).
- struct `std::greater_equal< _Tp >`
One of the [comparison functors](#).
- struct `std::less< _Tp >`
One of the [comparison functors](#).
- struct `std::less_equal< _Tp >`
One of the [comparison functors](#).
- struct `std::logical_and< _Tp >`
One of the [Boolean operations functors](#).
- struct `std::logical_not< _Tp >`

One of the [Boolean operations functors](#).

- struct [std::logical_or< _Tp >](#)
One of the [Boolean operations functors](#).
- class [std::mem_fun1_ref_t< _Ret, _Tp, _Arg >](#)
One of the [adaptors for member /// pointers](#).
- class [std::mem_fun1_t< _Ret, _Tp, _Arg >](#)
One of the [adaptors for member /// pointers](#).
- class [std::mem_fun_ref_t< _Ret, _Tp >](#)
One of the [adaptors for member /// pointers](#).
- class [std::mem_fun_t< _Ret, _Tp >](#)
One of the [adaptors for member /// pointers](#).
- struct [std::minus< _Tp >](#)
One of the [math functors](#).
- struct [std::modulus< _Tp >](#)
One of the [math functors](#).
- struct [std::multiplies< _Tp >](#)
One of the [math functors](#).
- struct [std::negate< _Tp >](#)
One of the [math functors](#).
- struct [std::not_equal_to< _Tp >](#)
One of the [comparison functors](#).
- struct [std::plus< _Tp >](#)
One of the [math functors](#).
- class [std::pointer_to_binary_function< _Arg1, _Arg2, _Result >](#)
One of the [adaptors for function pointers](#).
- class [std::pointer_to_unary_function< _Arg, _Result >](#)
One of the [adaptors for function pointers](#).
- struct [std::unary_function< _Arg, _Result >](#)
- class [std::unary_negate< _Predicate >](#)

One of the *[negation functors](#)*.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Ret, typename _Tp >`
`mem_fun_t< _Ret, _Tp > std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp >`
`mem_fun_ref_t< _Ret, _Tp > std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Predicate >`
`unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`
`binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`
- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))`

6.302.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_function.h](#).

6.303 `std_heap.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __-`
`holeIndex, _Distance __len, _Tp __value)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _`
`Compare >`
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __-`
`holeIndex, _Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`bool std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`
`bool std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _`
`Distance __n)`
- `template<typename _RandomAccessIterator >`
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __-`
`n, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __-`
`n)`
- `template<typename _RandomAccessIterator >`
`void std::__pop_heap (_RandomAccessIterator __first, _`
`RandomAccessIterator __last, _RandomAccessIterator __result)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__pop_heap (_RandomAccessIterator __first, _`
`RandomAccessIterator __last, _RandomAccessIterator __result, _Compare`
`__comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _`
`Compare >`
`void std::__push_heap (_RandomAccessIterator __first, _Distance __-`
`holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp >`
`void std::__push_heap (_RandomAccessIterator __first, _Distance __-`
`holeIndex, _Distance __topIndex, _Tp __value)`
- `template<typename _RandomAccessIterator >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __-`
`last, _Compare __comp)`

- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _-`
`RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator`
`__last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator _-`
`__last)`

6.303.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_heap.h](#).

6.304 `std_iterator.h` File Reference

Classes

- class [std::back_insert_iterator](#)< `_Container` >

Turns assignment into insertion.

- class [std::front_insert_iterator](#)< _Container >

Turns assignment into insertion.

- class [std::insert_iterator](#)< _Container >

Turns assignment into insertion.

- class [std::move_iterator](#)< _Iterator >
- class [std::reverse_iterator](#)< _Iterator >

Namespaces

- namespace [__gnu_cxx](#)
- namespace [std](#)

Defines

- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

Functions

- `template<typename _Container >`
`back_insert_iterator< _Container > std::back_inserter (_Container &__x)`
- `template<typename _Container >`
`front_insert_iterator< _Container > std::front_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator >`
`insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::make_move_iterator (const _Iterator &__i)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`

- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator!= (const __normal_iterator< _Iterator, _Container`
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator, _Container > __gnu_cxx::operator+ (typename`
`__normal_iterator< _Iterator, _Container >::difference_type __n, const __-`
`normal_iterator< _Iterator, _Container > &__i)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator`
`>::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _-`
`Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator, _Container >::difference_type __gnu_-`
`cxx::operator- (const __normal_iterator< _Iterator, _Container > &__lhs,`
`const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const move_iterator< _IteratorL > &__x, const move_-`
`iterator< _IteratorR > &__y)-> decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >::difference_type std::operator- (const reverse_-`
`iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_-`
`iterator< _IteratorR > &__y)-> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`auto __gnu_cxx::operator- (const __normal_iterator< _IteratorL, _Container`
`> &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)->`
`decltype(__lhs.base()-__rhs.base())`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_-`
`iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_-`
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator< (const __normal_iterator< _IteratorL, _Container`
`> &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator< (const __normal_iterator< _Iterator, _Container`
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`

- `template<typename _Iterator >`
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const`
`reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator<= (const __normal_iterator< _IteratorL, _`
`Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &_`
`rhs)`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator<= (const __normal_iterator< _Iterator, _Container`
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator== (const __normal_iterator< _Iterator, _Container`
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _Iterator >`
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_`
`iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator== (const __normal_iterator< _IteratorL, _`
`Container > &__lhs, const __normal_iterator< _IteratorR, _Container >`
`&__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_`
`iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator> (const __normal_iterator< _IteratorL, _Container`
`> &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator> (const __normal_iterator< _Iterator, _Container`
`> &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const`
`reverse_iterator< _IteratorR > &__y)`

- `template<typename _Iterator >`
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`

6.304.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

Definition in file [std_iterator.h](#).

6.305 `std_iterator_base_funcs.h` File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _InputIterator, typename _Distance >`
`void std::advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`
`void std::advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`

- `template<typename _RandomAccessIterator, typename _Distance>`
`void std::advance (_RandomAccessIterator &__i, _Distance __n, random_`
`access_iterator_tag)`
- `template<typename _RandomAccessIterator>`
`iterator_traits<_RandomAccessIterator>::difference_type std::distance (_`
`RandomAccessIterator __first, _RandomAccessIterator __last, random_access_`
`iterator_tag)`
- `template<typename _InputIterator>`
`iterator_traits<_InputIterator>::difference_type std::distance (_`
`InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _InputIterator, typename _Distance>`
`void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator>`
`iterator_traits<_InputIterator>::difference_type std::distance (_InputIterator _`
`__first, _InputIterator __last)`
- `template<typename _ForwardIterator>`
`_ForwardIterator std::next (_ForwardIterator __x, typename iterator_traits<_`
`ForwardIterator>::difference_type __n=1)`
- `template<typename _BidirectionalIterator>`
`_BidirectionalIterator std::prev (_BidirectionalIterator __x, typename iterator_`
`traits<_BidirectionalIterator>::difference_type __n=1)`

6.305.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file contains all of the general iterator-related utility functions, such as [distance\(\)](#) and [advance\(\)](#).

Definition in file [stl_iterator_base_funcs.h](#).

6.306 stl_iterator_base_types.h File Reference

Classes

- struct [std::__iterator_traits<_Iterator, bool>](#)
Traits class for iterators.
- struct [std::bidirectional_iterator_tag](#)
Bidirectional iterators support a superset of forward iterator /// operations.
- struct [std::forward_iterator_tag](#)

Forward iterators support a superset of input iterator operations.

- struct `std::input_iterator_tag`
Marking input iterators.
- struct `std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference>`
Common iterator class.
- struct `std::iterator_traits<_Tp*>`
Partial specialization for pointer types.
- struct `std::iterator_traits<const _Tp*>`
Partial specialization for const pointer types.
- struct `std::output_iterator_tag`
Marking output iterators.
- struct `std::random_access_iterator_tag`
Random-access iterators support a superset of bidirectional /// iterator operations.

Namespaces

- namespace `std`

Functions

- template<typename `_Iter`>
`iterator_traits<_Iter>::iterator_category` `std::__iterator_category` (`const _Iter` &)

6.306.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

Definition in file `std_iterator_base_types.h`.

6.307 `std_list.h` File Reference

Classes

- class `std::_List_base< _Tp, _Alloc >`
See `bits/stl_deque.h`'s `_Deque_base` for an explanation.
- struct `std::_List_const_iterator< _Tp >`
A `list::const_iterator`.
- struct `std::_List_iterator< _Tp >`
A `list::iterator`.
- struct `std::_List_node< _Tp >`
An actual node in the list.
- struct `std::_List_node_base`
Common part of a node in the list.
- class `std::list< _Tp, _Alloc >`
A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Namespaces

- namespace `std`

Functions

- `template<typename _Val >`
`bool std::operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`

- `template<typename _Val >`
`bool std::operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`

6.307.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_list.h](#).

6.308 stl_map.h File Reference

Classes

- class [std::map< _Key, _Tp, _Compare, _Alloc >](#)
A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`
`bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`
`bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`
`bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`
`bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`
`bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const`
`map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`
`void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp,`
`_Compare, _Alloc > &__y)`

6.308.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_map.h](#).

6.309 `stl_multimap.h` File Reference

Classes

- class [std::multimap< _Key, _Tp, _Compare, _Alloc >](#)

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x,`
`const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap<`
`_Key, _Tp, _Compare, _Alloc > &__y)`

6.309.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_multimap.h](#).

6.310 `std_multiset.h` File Reference

Classes

- class `std::multiset< _Key, _Compare, _Alloc >`
A standard container made up of elements, which can be retrieved in logarithmic time.

Namespaces

- namespace `std`

Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`

6.310.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_multiset.h](#).

6.311 [stl_numeric.h](#) File Reference

Namespaces

- namespace [std](#)

Functions

- `template<typename _InputIterator, typename _Tp >`
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`

- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`
`_Tp std::accumulate` (`_InputIterator __first`, `_InputIterator __last`, `_Tp __init`, `_BinaryOperation __binary_op`)
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::adjacent_difference` (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator __result`)
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::adjacent_difference` (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator __result`, `_BinaryOperation __binary_op`)
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`
`_Tp std::inner_product` (`_InputIterator1 __first1`, `_InputIterator1 __last1`, `_InputIterator2 __first2`, `_Tp __init`, `_BinaryOperation1 __binary_op1`, `_BinaryOperation2 __binary_op2`)
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`
`_Tp std::inner_product` (`_InputIterator1 __first1`, `_InputIterator1 __last1`, `_InputIterator2 __first2`, `_Tp __init`)
- `template<typename _ForwardIterator, typename _Tp >`
`void std::iota` (`_ForwardIterator __first`, `_ForwardIterator __last`, `_Tp __value`)
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::partial_sum` (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator __result`, `_BinaryOperation __binary_op`)
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::partial_sum` (`_InputIterator __first`, `_InputIterator __last`, `_OutputIterator __result`)

6.311.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_numeric.h](#).

6.312 `std_pair.h` File Reference

Classes

- `struct std::pair<_T1, _T2>`
pair holds two objects of arbitrary type.

Namespaces

- namespace [std](#)

Functions

- `template<class _T1, class _T2 >`
`pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type >` [std::make_pair](#) (`_T1 &&__x, _T2 &&__y`)
- `template<class _T1, class _T2 >`
`bool` [std::operator!=](#) (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2 >`
`bool` [std::operator<](#) (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2 >`
`bool` [std::operator<=](#) (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2 >`
`bool` [std::operator==](#) (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2 >`
`bool` [std::operator>](#) (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2 >`
`bool` [std::operator>=](#) (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2 >`
`void` [std::swap](#) (`pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y`)

Variables

- static const `piecewise_construct_t` [std::piecewise_construct](#)

6.312.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_pair.h](#).

6.313 `std_queue.h` File Reference

Classes

- class `std::priority_queue<_Tp, _Sequence, _Compare >`
A standard container automatically sorting its contents.
- class `std::queue<_Tp, _Sequence >`
A standard container giving FIFO behavior.

Namespaces

- namespace `std`

Functions

- `template<typename _Tp, typename _Seq >`
`bool std::operator!= (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator< (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator<= (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator== (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator> (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator>= (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Sequence, typename _Compare >`
`void std::swap (priority_queue<_Tp, _Sequence, _Compare > &__x, priority_queue<_Tp, _Sequence, _Compare > &__y)`
- `template<typename _Tp, typename _Seq >`
`void std::swap (queue<_Tp, _Seq > &__x, queue<_Tp, _Seq > &__y)`

6.313.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_queue.h](#).

6.314 std_raw_storage_iter.h File Reference

Classes

- class [std::raw_storage_iterator](#)< [_OutputIterator](#), [_Tp](#) >

Namespaces

- namespace [std](#)

6.314.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_raw_storage_iter.h](#).

6.315 std_relops.h File Reference

Namespaces

- namespace [std](#)
- namespace [std::rel_ops](#)

Functions

- [template](#)<class [_Tp](#) >
bool [std::rel_ops::operator!=](#) (const [_Tp](#) &__x, const [_Tp](#) &__y)
- [template](#)<class [_Tp](#) >
bool [std::rel_ops::operator<=](#) (const [_Tp](#) &__x, const [_Tp](#) &__y)
- [template](#)<class [_Tp](#) >
bool [std::rel_ops::operator>](#) (const [_Tp](#) &__x, const [_Tp](#) &__y)
- [template](#)<class [_Tp](#) >
bool [std::rel_ops::operator>=](#) (const [_Tp](#) &__x, const [_Tp](#) &__y)

6.315.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std_utility.h`. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html>, or http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads

Short summary: the `rel_ops` operators should be avoided for the present.

Definition in file [std_relops.h](#).

6.316 `std_set.h` File Reference

Classes

- class [std::set< _Key, _Compare, _Alloc >](#)
A standard container made up of unique keys, which can be retrieved in logarithmic time.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`

- `template<typename _Key , typename _Compare , typename _Alloc >`
`bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set<`
`_Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`
`bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set<`
`_Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`
`void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare,`
`_Alloc > &__y)`

6.316.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_set.h](#).

6.317 `stl_stack.h` File Reference

Classes

- class [std::stack< _Tp, _Sequence >](#)
A standard container giving FILO behavior.

Namespaces

- namespace [std](#)

Functions

- `template<typename _Tp , typename _Seq >`
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq`
`> &__y)`
- `template<typename _Tp , typename _Seq >`
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq`
`> &__y)`
- `template<typename _Tp , typename _Seq >`
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq`
`> &__y)`

- `template<typename _Tp, typename _Seq >`
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`void std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y)`

6.317.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std_stack.h](#).

6.318 `std_tempbuf.h` File Reference

Classes

- class [std::_Temporary_buffer<_ForwardIterator, _Tp >](#)

Namespaces

- namespace [std](#)

Functions

- `template<typename _ForwardIterator, typename _Tp >`
`void std::__uninitialized_construct_buf (_ForwardIterator __first, _ForwardIterator __last, _Tp &__value)`
- `template<typename _Tp >`
`pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len)`
- `template<typename _Tp >`
`void std::return_temporary_buffer (_Tp *__p)`

6.318.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl_tempbuf.h](#).

6.319 stl_tree.h File Reference

Namespaces

- namespace [std](#)

Enumerations

- enum **_Rb_tree_color** { **_S_red**, **_S_black** }

Functions

- **std::__attribute__** ((__pure__)) **_Rb_tree_node_base** * **_Rb_tree_increment**(**_Rb_tree_node_base** * __x) throw ()
- void **std::__Rb_tree_insert_and_rebalance** (const bool __insert_left, **_Rb_tree_node_base** * __x, **_Rb_tree_node_base** * __p, **_Rb_tree_node_base** & __header) throw ()
- **_Rb_tree_node_base** * **std::__Rb_tree_rebalance_for_erase** (**_Rb_tree_node_base** * const __z, **_Rb_tree_node_base** & __header) throw ()
- template<typename **_Key** , typename **_Val** , typename **_KeyOfValue** , typename **_Compare** , typename **_Alloc** >
bool **std::operator!=** (const **_Rb_tree**< **_Key**, **_Val**, **_KeyOfValue**, **_Compare**, **_Alloc** > & __x, const **_Rb_tree**< **_Key**, **_Val**, **_KeyOfValue**, **_Compare**, **_Alloc** > & __y)
- template<typename **_Val** >
bool **std::operator!=** (const **_Rb_tree_iterator**< **_Val** > & __x, const **_Rb_tree_const_iterator**< **_Val** > & __y)
- template<typename **_Key** , typename **_Val** , typename **_KeyOfValue** , typename **_Compare** , typename **_Alloc** >
bool **std::operator**< (const **_Rb_tree**< **_Key**, **_Val**, **_KeyOfValue**, **_Compare**, **_Alloc** > & __x, const **_Rb_tree**< **_Key**, **_Val**, **_KeyOfValue**, **_Compare**, **_Alloc** > & __y)
- template<typename **_Key** , typename **_Val** , typename **_KeyOfValue** , typename **_Compare** , typename **_Alloc** >
bool **std::operator**<= (const **_Rb_tree**< **_Key**, **_Val**, **_KeyOfValue**, **_Compare**,

- ```

_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc
> &__y)

```
- `template<typename _Val >`  
`bool std::operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_`  
`tree_const_iterator< _Val > &__y)`
  - `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`  
`name _Alloc >`  
`bool std::operator== (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare,`  
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`  
`> &__y)`
  - `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`  
`name _Alloc >`  
`bool std::operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare,`  
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`  
`> &__y)`
  - `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`  
`name _Alloc >`  
`bool std::operator>= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare,`  
`_Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc`  
`> &__y)`
  - `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, type-`  
`name _Alloc >`  
`void std::swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >`  
`&__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
  - `const _Rb_tree_node_base *__root std::throw ()`

### 6.319.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_tree.h](#).

## 6.320 `std_uninitialized.h` File Reference

### Namespaces

- namespace `std`

### Functions

- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`

```

_FwdIterator std::__uninitialized_copy_a (_InputIterator __first, _-
InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)
• template<typename _InputIterator, typename _ForwardIterator, typename _Tp>
_FwdIterator std::__uninitialized_copy_a (_InputIterator __first, _-
InputIterator __last, _ForwardIterator __result, allocator<_Tp> &__alloc)
• template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-
name _Allocator>
_FwdIterator std::__uninitialized_copy_move (_InputIterator1 __first1, _-
InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-
ForwardIterator __result, _Allocator &__alloc)
• template<typename _InputIterator, typename _Size, typename _ForwardIterator>
_FwdIterator std::__uninitialized_copy_n (_InputIterator __first, _Size _-
n, _ForwardIterator __result, input_iterator_tag)
• template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator>
_FwdIterator std::__uninitialized_copy_n (_RandomAccessIterator __-
first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)
• template<typename _ForwardIterator>
void std::__uninitialized_default (_ForwardIterator __first, _ForwardIterator
__last)
• template<typename _ForwardIterator, typename _Allocator>
void std::__uninitialized_default_a (_ForwardIterator __first, _-
ForwardIterator __last, _Allocator &__alloc)
• template<typename _ForwardIterator, typename _Tp>
void std::__uninitialized_default_a (_ForwardIterator __first, _-
ForwardIterator __last, allocator<_Tp> &__alloc)
• template<typename _ForwardIterator, typename _Size>
void std::__uninitialized_default_n (_ForwardIterator __first, _Size __n)
• template<typename _ForwardIterator, typename _Size, typename _Allocator>
void std::__uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _-
Allocator &__alloc)
• template<typename _ForwardIterator, typename _Size, typename _Tp>
void std::__uninitialized_default_n_a (_ForwardIterator __first, _Size __n,
allocator<_Tp> &__alloc)
• template<typename _ForwardIterator, typename _Tp, typename _Tp2>
void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __-
last, const _Tp &__x, allocator<_Tp2> &__alloc)
• template<typename _ForwardIterator, typename _Tp, typename _Allocator>
void std::__uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __-
last, const _Tp &__x, _Allocator &__alloc)
• template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _-
Allocator>
_FwdIterator std::__uninitialized_fill_move (_ForwardIterator __result, _-
ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator
__last, _Allocator &__alloc)

```



- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`  
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const`  
`_Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`  
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const`  
`_Tp &__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`  
`_ForwardIterator std::__uninitialized_move_a (_InputIterator __first, _-`  
`InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, type-`  
`name _Allocator >`  
`_ForwardIterator std::__uninitialized_move_copy (_InputIterator1 __first1, _-`  
`InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _-`  
`ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _-`  
`Allocator >`  
`void std::__uninitialized_move_fill (_InputIterator __first1, _InputIterator __-`  
`last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _-`  
`Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator`  
`__last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _-`  
`ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &_-`  
`__x)`

### 6.320.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [std\\_uninitialized.h](#).

## 6.321 `std_vector.h` File Reference

### Classes

- struct [std::\\_Vector\\_base< \\_Tp, \\_Alloc >](#)

See [bits/stl\\_deque.h's `\_Deque\_base`](#) for an explanation.

- class [std::vector](#)< [\\_Tp](#), [\\_Alloc](#) >

*A standard container which offers fixed time access to individual elements in any order.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const vector< \_Tp, \_Alloc > &__x, const vector< \_Tp, \_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const vector< \_Tp, \_Alloc > &__x, const vector< \_Tp, \_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const vector< \_Tp, \_Alloc > &__x, const vector< \_Tp, \_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const vector< \_Tp, \_Alloc > &__x, const vector< \_Tp, \_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const vector< \_Tp, \_Alloc > &__x, const vector< \_Tp, \_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const vector< \_Tp, \_Alloc > &__x, const vector< \_Tp, \_Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (vector< \_Tp, \_Alloc > &__x, vector< \_Tp, \_Alloc > &__y)`

### 6.321.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stl\\_vector.h](#).

## 6.322 stream\_iterator.h File Reference

### Classes

- class [std::istream\\_iterator](#)< \_Tp, \_CharT, \_Traits, \_Dist >  
*Provides input iterator semantics for streams.*
- class [std::ostream\\_iterator](#)< \_Tp, \_CharT, \_Traits >  
*Provides output iterator semantics for streams.*

### Namespaces

- namespace [std](#)

### Functions

- template<class \_Tp, class \_CharT, class \_Traits, class \_Dist >  
bool [std::operator!=](#) (const istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > &\_  
\_x, const istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > &\_\_y)
- template<typename \_Tp, typename \_CharT, typename \_Traits, typename \_Dist >  
bool [std::operator==](#) (const istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > &\_  
\_x, const istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > &\_\_y)

### 6.322.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stream\\_iterator.h](#).

## 6.323 streambuf File Reference

### Classes

- class [std::basic\\_streambuf](#)< \_CharT, \_Traits >  
*The actual work of input and output (interface).  
This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.*

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits >`  
`*, basic_streambuf< _CharT, _Traits > *, bool &)`
- `template<>`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< wchar_t > *__-`  
`sbin, basic_streambuf< wchar_t > *__sbout, bool &__ineof)`
- `template<>`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< char > *__sbin,`  
`basic_streambuf< char > *__sbout, bool &__ineof)`

### 6.323.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [streambuf](#).

## 6.324 streambuf.tcc File Reference

## Namespaces

- namespace [std](#)

## Functions

- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs (basic_streambuf< _CharT, _Traits > *_-`  
`__sbin, basic_streambuf< _CharT, _Traits > *__sbout)`
- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs_eof (basic_streambuf< _CharT, _Traits >`  
`*, basic_streambuf< _CharT, _Traits > *, bool &)`

### 6.324.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [streambuf.tcc](#).

## 6.325 streambuf\_iterator.h File Reference

### Classes

- class [std::istreambuf\\_iterator< \\_CharT, \\_Traits >](#)  
*Provides input iterator semantics for streambufs.*
- class [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >](#)  
*Provides output iterator semantics for streambufs.*

### Namespaces

- namespace [std](#)

### Functions

- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy_move_a2 (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy_move_a2 (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)`
- `template<typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)`
- `template<typename _CharT >  
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`

- `template<typename _CharT, typename _Traits >`  
`bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a,`  
`const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _CharT, typename _Traits >`  
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a,`  
`const istreambuf_iterator< _CharT, _Traits > &__b)`

### 6.325.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [streambuf\\_iterator.h](#).

## 6.326 string File Reference

### 6.326.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [string](#).

## 6.327 string File Reference

### Classes

- class [\\_\\_gnu\\_debug::basic\\_string< \\_CharT, \\_Traits, \\_Allocator >](#)  
*Class [std::basic\\_string](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [\\_\\_gnu\\_debug](#)

### Typedefs

- `typedef basic_string< char > \_\_gnu\_debug::string`
- `typedef basic_string< wchar_t > \_\_gnu\_debug::wstring`

## Functions

- `template<typename _CharT, typename _Traits, typename _Allocator >  
std::basic_istream< _CharT, _Traits > & __gnu_debug::getline (std::basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Allocator > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
std::basic_istream< _CharT, _Traits > & __gnu_debug::getline (std::basic_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Allocator > & __str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator!= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator!= (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator!= (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, _Allocator > & __lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
basic_string< _CharT, _Traits, _Allocator > __gnu_debug::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator< (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator< (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
bool __gnu_debug::operator< (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __rhs)`

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_debug::operator<<`  
`(std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT,`  
`_Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const _CharT *__lhs, const basic_string< _`  
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _`  
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator<= (const basic_string< _CharT, _Traits, _`  
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _`  
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const basic_string< _CharT, _Traits, _`  
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator== (const _CharT *__lhs, const basic_string< _`  
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _`  
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const _CharT *__lhs, const basic_string< _`  
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator> (const basic_string< _CharT, _Traits, _`  
`Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic_string< _`  
`CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _`  
`Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _`  
`Allocator > &__lhs, const _CharT *__rhs)`



- `template<typename _CharT, typename _Traits, typename _Allocator >  
std::basic_istream< _CharT, _Traits > & __gnu_debug::operator>>  
(std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits,  
_Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >  
void __gnu_debug::swap (basic_string< _CharT, _Traits, _Allocator > &__-  
lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)`

### 6.327.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/string](#).

## 6.328 stringfwd.h File Reference

### Namespaces

- namespace [std](#)

### Typedefs

- `typedef basic_string< char > std::string`
- `typedef basic_string< char16_t > std::u16string`
- `typedef basic_string< char32_t > std::u32string`
- `typedef basic_string< wchar_t > std::wstring`

### 6.328.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [stringfwd.h](#).

## 6.329 system\_error File Reference

### Classes

- class [std::error\\_category](#)  
*error\_category*

- struct `std::error_code`  
*error\_code*
- struct `std::error_condition`  
*error\_condition*
- struct `std::hash< error_code >`  
*std::hash specialization for error\_code.*
- struct `std::is_error_code_enum< _Tp >`  
*is\_error\_code\_enum*
- struct `std::is_error_condition_enum< _Tp >`  
*is\_error\_condition\_enum*
- class `std::system_error`  
*Thrown to indicate error code of underlying system.*

## Namespaces

- namespace `std`

## Functions

- `_GLIBCXX_CONST` `const error_category & std::generic_category ()` `throw ()`
- `error_code` `std::make_error_code (errc __e)`
- `error_condition` `std::make_error_condition (errc __e)`
- `bool` `std::operator!= (const error_code &__lhs, const error_condition &__rhs)`
- `bool` `std::operator!= (const error_condition &__lhs, const error_condition &__rhs)`
- `bool` `std::operator!= (const error_code &__lhs, const error_code &__rhs)`
- `bool` `std::operator!= (const error_condition &__lhs, const error_code &__rhs)`
- `bool` `std::operator< (const error_code &__lhs, const error_code &__rhs)`
- `bool` `std::operator< (const error_condition &__lhs, const error_condition &__rhs)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)`

- bool **std::operator==** (const error\_condition &\_\_lhs, const error\_condition &\_\_rhs)
- bool **std::operator==** (const error\_code &\_\_lhs, const error\_condition &\_\_rhs)
- bool **std::operator==** (const error\_code &\_\_lhs, const error\_code &\_\_rhs)
- bool **std::operator==** (const error\_condition &\_\_lhs, const error\_code &\_\_rhs)
- \_GLIBCXX\_CONST const error\_category & **std::system\_category** () throw ()

## Variables

- error\_code **std::make\_error\_code** (errc)
- error\_condition **std::make\_error\_condition** (errc)

### 6.329.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [system\\_error](#).

## 6.330 tag\_and\_trait.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::associative\\_container\\_tag](#)  
*Basic associative-container.*
- struct [\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#)  
*Basic hash.*
- struct [\\_\\_gnu\\_pbds::basic\\_tree\\_tag](#)  
*Basic tree.*
- struct [\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#)  
*Binary-heap (array-based).*
- struct [\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#)  
*Binomial-heap.*
- struct [\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#)  
*Collision-chaining hash.*

- struct [\\_\\_gnu\\_pbds::container\\_tag](#)  
*Base data structure tag.*
- struct [\\_\\_gnu\\_pbds::container\\_traits< Cntnr >](#)  
*container\_traits*
- struct [\\_\\_gnu\\_pbds::gp\\_hash\\_tag](#)  
*General-probing hash.*
- struct [\\_\\_gnu\\_pbds::list\\_update\\_tag](#)  
*List-update.*
- struct [\\_\\_gnu\\_pbds::null\\_mapped\\_type](#)  
*A mapped-policy indicating that an associative container is a set.*
- struct [\\_\\_gnu\\_pbds::ov\\_tree\\_tag](#)  
*Ordered-vector tree.*
- struct [\\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#)  
*Pairing-heap.*
- struct [\\_\\_gnu\\_pbds::pat\\_trie\\_tag](#)  
*PATRICIA trie.*
- struct [\\_\\_gnu\\_pbds::priority\\_queue\\_tag](#)  
*Basic priority-queue.*
- struct [\\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)  
*Red-black tree.*
- struct [\\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)  
*Redundant-counter binomial-heap.*
- struct [\\_\\_gnu\\_pbds::sequence\\_tag](#)  
*Basic sequence.*
- struct [\\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)  
*Splay tree.*
- struct [\\_\\_gnu\\_pbds::string\\_tag](#)  
*Basic string container, inclusive of strings, ropes, etc.*

- struct [\\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)  
*Thin heap.*
- struct [\\_\\_gnu\\_pbds::tree\\_tag](#)  
*tree.*
- struct [\\_\\_gnu\\_pbds::trie\\_tag](#)  
*trie.*

## Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

## Typedefs

- typedef void [\\_\\_gnu\\_pbds::trivial\\_iterator\\_difference\\_type](#)

### 6.330.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

Definition in file [tag\\_and\\_trait.hpp](#).

## 6.331 tags.h File Reference

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

## Classes

- struct [\\_\\_gnu\\_parallel::balanced\\_quicksort\\_tag](#)  
*Forces parallel sorting using balanced quicksort at compile time.*
- struct [\\_\\_gnu\\_parallel::balanced\\_tag](#)  
*Recommends parallel execution using dynamic load-balancing at compile time.*
- struct [\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag](#)  
*Selects the constant block size variant for `std::find()`.*

- struct [\\_\\_gnu\\_parallel::default\\_parallel\\_tag](#)  
*Recommends parallel execution using the default parallel algorithm.*
- struct [\\_\\_gnu\\_parallel::equal\\_split\\_tag](#)  
*Selects the equal splitting variant for `std::find()`.*
- struct [\\_\\_gnu\\_parallel::exact\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*
- struct [\\_\\_gnu\\_parallel::find\\_tag](#)  
*Base class for `std::find()` variants.*
- struct [\\_\\_gnu\\_parallel::growing\\_blocks\\_tag](#)  
*Selects the growing block size variant for `std::find()`.*
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_exact\\_tag](#)  
*Forces parallel sorting using multiway mergesort with exact splitting at compile time.*
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_sampling\\_tag](#)  
*Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.*
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_tag](#)  
*Forces parallel sorting using multiway mergesort at compile time.*
- struct [\\_\\_gnu\\_parallel::omp\\_loop\\_static\\_tag](#)  
*Recommends parallel execution using OpenMP static load-balancing at compile time.*
- struct [\\_\\_gnu\\_parallel::omp\\_loop\\_tag](#)  
*Recommends parallel execution using OpenMP dynamic load-balancing at compile time.*
- struct [\\_\\_gnu\\_parallel::parallel\\_tag](#)  
*Recommends parallel execution at compile time, optionally using a user-specified number of threads.*
- struct [\\_\\_gnu\\_parallel::quicksort\\_tag](#)  
*Forces parallel sorting using unbalanced quicksort at compile time.*
- struct [\\_\\_gnu\\_parallel::sampling\\_tag](#)  
*Forces parallel merging with exact splitting, at compile time.*

- struct `__gnu_parallel::sequential_tag`  
*Forces sequential execution at compile time.*
- struct `__gnu_parallel::unbalanced_tag`  
*Recommends parallel execution using static load-balancing at compile time.*

## Namespaces

- namespace `__gnu_parallel`

### 6.331.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [tags.h](#).

## 6.332 `tgmath.h` File Reference

### 6.332.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tgmath.h](#).

## 6.333 `thread` File Reference

### Classes

- struct `std::hash< thread::id >`  
*`std::hash` specialization for `thread::id`.*
- class `std::thread`  
*`thread`*
- class `std::thread::id`  
*`thread::id`*

## Namespaces

- namespace [std](#)
- namespace [std::this\\_thread](#)

## Functions

- `thread::id` [std::this\\_thread::get\\_id](#) ()
- `bool` **`std::operator!=`** (`thread::id __x`, `thread::id __y`)
- `template<class _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`  
`CharT, _Traits > &__out, thread::id __id)`
- `bool` **`std::operator<=`** (`thread::id __x`, `thread::id __y`)
- `bool` **`std::operator>`** (`thread::id __x`, `thread::id __y`)
- `bool` **`std::operator>=`** (`thread::id __x`, `thread::id __y`)
- `template<typename _Rep, typename _Period >`  
`void std::this\_thread::sleep\_for (const chrono::duration< _Rep, _Period > &_`  
`_rtime)`
- `template<typename _Clock, typename _Duration >`  
`void std::this\_thread::sleep\_until (const chrono::time_point< _Clock, _Duration`  
`> &__atime)`
- `void` **`std::swap`** (`thread &__x`, `thread &__y`)
- `void` [std::this\\_thread::yield](#) ()

### 6.333.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [thread](#).

## 6.334 `throw_allocator.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::annotate\\_base](#)  
*Base class for checking address and label information about allocations. Create a [std::map](#) between the allocated address (void\*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.*
- struct [\\_\\_gnu\\_cxx::condition\\_base](#)  
*Base struct for condition policy.*



- struct [`\_\_gnu\_cxx::forced\_error`](#)  
*Thrown by exception safety machinery.*
- struct [`\_\_gnu\_cxx::limit\_condition`](#)  
*Base class for incremental control and throw.*
- struct [`\_\_gnu\_cxx::limit\_condition::always\_adjustor`](#)  
*Always enter the condition.*
- struct [`\_\_gnu\_cxx::limit\_condition::limit\_adjustor`](#)  
*Enter the *n*th condition.*
- struct [`\_\_gnu\_cxx::limit\_condition::never\_adjustor`](#)  
*Never enter the condition.*
- struct [`\_\_gnu\_cxx::random\_condition`](#)  
*Base class for random probability control and throw.*
- struct [`\_\_gnu\_cxx::random\_condition::always\_adjustor`](#)  
*Always enter the condition.*
- struct [`\_\_gnu\_cxx::random\_condition::group\_adjustor`](#)  
*Group condition.*
- struct [`\_\_gnu\_cxx::random\_condition::never\_adjustor`](#)  
*Never enter the condition.*
- class [`\_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >`](#)  
*Allocator class with logging and exception generation control. Intended to be used as an `allocator_type` in templated code.*  
*Note: Deallocate not allowed to throw.*
- struct [`\_\_gnu\_cxx::throw\_allocator\_limit< \_Tp >`](#)  
*Allocator throwing via limit condition.*
- struct [`\_\_gnu\_cxx::throw\_allocator\_random< \_Tp >`](#)  
*Allocator throwing via random condition.*
- struct [`\_\_gnu\_cxx::throw\_value\_base< \_Cond >`](#)  
*Class with exception generation control. Intended to be used as a `value_type` in templated code.*
- struct [`\_\_gnu\_cxx::throw\_value\_limit`](#)

*Type throwing via limit condition.*

- struct [\\_\\_gnu\\_cxx::throw\\_value\\_random](#)

*Type throwing via random condition.*

- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)

*Explicit specialization of [std::hash](#) for [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#).*

- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_random >](#)

*Explicit specialization of [std::hash](#) for [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#).*

## Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

## Functions

- void [\\_\\_gnu\\_cxx::throw\\_forced\\_error](#) ()
- template<typename \_Tp, typename \_Cond >  
bool [\\_\\_gnu\\_cxx::operator!=](#) (const throw\_allocator\_base< \_Tp, \_Cond > &, const throw\_allocator\_base< \_Tp, \_Cond > &)
- template<typename \_Cond >  
throw\_value\_base< \_Cond > [\\_\\_gnu\\_cxx::operator\\*](#) (const throw\_value\_base< \_Cond > &\_\_a, const throw\_value\_base< \_Cond > &\_\_b)
- template<typename \_Cond >  
throw\_value\_base< \_Cond > [\\_\\_gnu\\_cxx::operator+](#) (const throw\_value\_base< \_Cond > &\_\_a, const throw\_value\_base< \_Cond > &\_\_b)
- template<typename \_Cond >  
throw\_value\_base< \_Cond > [\\_\\_gnu\\_cxx::operator-](#) (const throw\_value\_base< \_Cond > &\_\_a, const throw\_value\_base< \_Cond > &\_\_b)
- template<typename \_Cond >  
bool [\\_\\_gnu\\_cxx::operator<](#) (const throw\_value\_base< \_Cond > &\_\_a, const throw\_value\_base< \_Cond > &\_\_b)
- [std::ostream](#) & [\\_\\_gnu\\_cxx::operator<<](#) ([std::ostream](#) &os, const annotate\_base &\_\_b)
- template<typename \_Tp, typename \_Cond >  
bool [\\_\\_gnu\\_cxx::operator==](#) (const throw\_allocator\_base< \_Tp, \_Cond > &, const throw\_allocator\_base< \_Tp, \_Cond > &)
- template<typename \_Cond >  
bool [\\_\\_gnu\\_cxx::operator==](#) (const throw\_value\_base< \_Cond > &\_\_a, const throw\_value\_base< \_Cond > &\_\_b)

- `template<typename _Cond >`  
`void __gnu_cxx::swap` (`throw_value_base< _Cond > &__a`, `throw_value_base< _Cond > &__b`)

### 6.334.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (`throw_value`, `throw_allocator`) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type `forced_exception_error`.

Definition in file [throw\\_allocator.h](#).

## 6.335 `time_members.h` File Reference

### Namespaces

- namespace [std](#)

### 6.335.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [time\\_members.h](#).

## 6.336 `tree_policy.hpp` File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 6.336.1 Detailed Description

Contains tree-related policies.

Definition in file [tree\\_policy.hpp](#).

## 6.337 tree\_trace\_base.hpp File Reference

### 6.337.1 Detailed Description

Contains tree-related policies.

Definition in file [tree\\_trace\\_base.hpp](#).

## 6.338 trie\_policy.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- `#define PB_DS_BASE_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 6.338.1 Detailed Description

Contains trie-related policies.

Definition in file [trie\\_policy.hpp](#).

## 6.339 tuple File Reference

### Classes

- struct [std::\\_Build\\_index\\_tuple<\\_Num>](#)

*Builds an `_Index_tuple`<0, 1, 2, ..., `_Num-1`>.*

- struct `std::_Index_tuple<_Indexes>`
- struct `std::_Tuple_impl<_Idx>`
- struct `std::_Tuple_impl<_Idx, _Head, _Tail...>`
- class `std::tuple<_Elements>`  
*tuple*
- class `std::tuple<_T1>`  
*tuple (1-element).*
- class `std::tuple<_T1, _T2>`  
*tuple (2-element), with construction and assignment from a pair.*
- struct `std::tuple_element<0, tuple<_Head, _Tail...>>`
- struct `std::tuple_element<__i, tuple<_Head, _Tail...>>`
- struct `std::tuple_size<tuple<_Elements...>>`  
*class tuple\_size*

## Namespaces

- namespace `std`

## Functions

- template<std::size\_t \_\_i, typename \_Head, typename... \_Tail>  
`__add_ref<_Head>::type std::__get_helper(_Tuple_impl<__i, _Head, _Tail...> &__t)`
- template<std::size\_t \_\_i, typename \_Head, typename... \_Tail>  
`__add_c_ref<_Head>::type std::__get_helper(const _Tuple_impl<__i, _Head, _Tail...> &__t)`
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
`tuple<_TElements..., _UElements...> std::__tuple_cat_helper(tuple<_TElements...> &&__t, const __index_holder<_TIdx...> &, tuple<_UElements...> &&__u, const __index_holder<_UIdx...> &)`
- template<typename... \_TElements, std::size\_t... \_TIdx, typename... \_UElements, std::size\_t... \_UIdx>  
`tuple<_TElements..., _UElements...> std::__tuple_cat_helper(const tuple<_TElements...> &&__t, const __index_holder<_TIdx...> &, const tuple<_UElements...> &&__u, const __index_holder<_UIdx...> &)`

- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`  
`tuple< _TElements..., _UElements...> std::__tuple_cat_helper (tuple< _TElements...> &&__t, const __index_holder< _TIdx...> &, const tuple< _UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<typename... _TElements, std::size_t... _TIdx, typename... _UElements, std::size_t... _UIdx>`  
`tuple< _TElements..., _UElements...> std::__tuple_cat_helper (const tuple< _TElements...> &&__t, const __index_holder< _TIdx...> &, tuple< _UElements...> &&__u, const __index_holder< _UIdx...> &)`
- `template<typename... _Elements>`  
`tuple< _Elements &&...> std::forward_as_tuple (_Elements &&...__args)`
- `template<std::size_t __i, typename... _Elements>`  
`__add_c_ref< typename tuple_element< __i, tuple< _Elements...> >::type>::type std::get (const tuple< _Elements...> &__t)`
- `template<std::size_t __i, typename... _Elements>`  
`__add_ref< typename tuple_element< __i, tuple< _Elements...> >::type>::type std::get (tuple< _Elements...> &__t)`
- `template<typename... _Elements>`  
`tuple< typename __decay_and_strip< _Elements >::__type...> std::make_tuple (_Elements &&...__args)`
- `template<typename... _TElements, typename... _UElements>`  
`bool std::operator!= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`bool std::operator< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`bool std::operator<= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`bool std::operator== (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`bool std::operator> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`bool std::operator>= (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _Elements>`  
`void std::swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y)`
- `template<typename... _Elements>`  
`tuple< _Elements &&...> std::tie (_Elements &&...__args)`

- `template<typename... _TElements, typename... _UElements>  
tuple< _TElements..., _UElements...> std::tuple_cat (tuple< _TElements...>  
&&__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>  
tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-  
TElements...> &__t, tuple< _UElements...> &&__u)`
- `template<typename... _TElements, typename... _UElements>  
tuple< _TElements..., _UElements...> std::tuple_cat (const tuple< _-  
TElements...> &__t, const tuple< _UElements...> &__u)`
- `template<typename... _TElements, typename... _UElements>  
tuple< _TElements..., _UElements...> std::tuple_cat (tuple< _TElements...>  
&&__t, const tuple< _UElements...> &__u)`

## Variables

- `const _Swallow_assign std::ignore`

### 6.339.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [tuple](#).

## 6.340 type\_traits File Reference

### Classes

- struct [std::\\_\\_declval\\_protector< \\_Tp >](#)  
*declval*
- struct [std::add\\_lvalue\\_reference< \\_Tp >](#)  
*add\_lvalue\_reference*
- struct [std::add\\_rvalue\\_reference< \\_Tp >](#)  
*add\_rvalue\_reference*
- struct [std::aligned\\_storage< \\_Len, \\_Align >](#)  
*Alignment type.*
- struct [std::conditional< \\_Cond, \\_Iftrue, \\_Iffalse >](#)  
*conditional*

- class `std::decay< _Tp >`  
*decay*
- struct `std::enable_if< bool, _Tp >`  
*enable\_if*
- struct `std::has_nothrow_copy_assign< _Tp >`  
*has\_nothrow\_copy\_assign*
- struct `std::has_nothrow_copy_constructor< _Tp >`  
*has\_nothrow\_copy\_constructor*
- struct `std::has_nothrow_default_constructor< _Tp >`  
*has\_nothrow\_default\_constructor*
- struct `std::has_trivial_copy_assign< _Tp >`  
*has\_trivial\_copy\_assign*
- struct `std::has_trivial_copy_constructor< _Tp >`  
*has\_trivial\_copy\_constructor*
- struct `std::has_trivial_default_constructor< _Tp >`  
*has\_trivial\_default\_constructor*
- struct `std::has_trivial_destructor< _Tp >`  
*has\_trivial\_destructor*
- struct `std::is_base_of< _Base, _Derived >`  
*is\_base\_of*
- struct `std::is_constructible< _Tp, _Args >`  
*is\_constructible*
- struct `std::is_convertible< _From, _To >`  
*is\_convertible*
- struct `std::is_explicitly_convertible< _From, _To >`  
*is\_explicitly\_convertible*
- struct `std::is_lvalue_reference< typename >`  
*is\_lvalue\_reference*



- struct `std::is_nothrow_constructible< _Tp, _Args >`  
*is\_nothrow\_constructible*
- struct `std::is_pod< _Tp >`  
*is\_pod*
- struct `std::is_reference< _Tp >`  
*is\_reference*
- struct `std::is_rvalue_reference< typename >`  
*is\_rvalue\_reference*
- struct `std::is_signed< _Tp >`  
*is\_signed*
- struct `std::is_standard_layout< _Tp >`  
*is\_standard\_layout*
- struct `std::is_trivial< _Tp >`  
*is\_trivial*
- struct `std::is_unsigned< _Tp >`  
*is\_unsigned*
- struct `std::make_signed< _Tp >`  
*make\_signed*
- struct `std::make_unsigned< _Tp >`  
*make\_unsigned*
- struct `std::remove_reference< _Tp >`  
*remove\_reference*

## Namespaces

- namespace `std`

## Defines

- `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`

## Functions

- `template<typename _Tp >`  
`add_rvalue_reference< _Tp >::type` **std::declval** () noexcept

### 6.340.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [type\\_traits](#).

## 6.341 `type_traits` File Reference

### Classes

- struct `std::tr1::__is_member_pointer_helper< _Tp >`  
*is\_member\_pointer*
- struct `std::tr1::add_const< _Tp >`  
*add\_const*
- struct `std::tr1::add_cv< _Tp >`  
*add\_cv*
- struct `std::tr1::add_pointer< _Tp >`  
*add\_pointer*
- struct `std::tr1::add_volatile< _Tp >`  
*add\_volatile*
- struct `std::tr1::alignment_of< _Tp >`  
*alignment\_of*
- struct `std::tr1::extent< typename, _Uint >`  
*extent*
- struct `std::tr1::has_virtual_destructor< _Tp >`  
*has\_virtual\_destructor*
- struct `std::tr1::integral_constant< _Tp, __v >`  
*integral\_constant*

- struct `std::tr1::is_abstract< _Tp >`  
*is\_abstract*
- struct `std::tr1::is_arithmetic< _Tp >`  
*is\_arithmetic*
- struct `std::tr1::is_array< typename >`  
*is\_array*
- struct `std::tr1::is_class< _Tp >`  
*is\_class*
- struct `std::tr1::is_compound< _Tp >`  
*is\_compound*
- struct `std::tr1::is_const< typename >`  
*is\_const*
- struct `std::tr1::is_empty< _Tp >`  
*is\_empty*
- struct `std::tr1::is_enum< _Tp >`  
*is\_enum*
- struct `std::tr1::is_floating_point< _Tp >`  
*is\_floating\_point*
- struct `std::tr1::is_function< typename >`  
*is\_function*
- struct `std::tr1::is_fundamental< _Tp >`  
*is\_fundamental*
- struct `std::tr1::is_integral< _Tp >`  
*is\_integral*
- struct `std::tr1::is_member_function_pointer< _Tp >`  
*is\_member\_function\_pointer*
- struct `std::tr1::is_member_object_pointer< _Tp >`  
*is\_member\_object\_pointer*

- struct `std::tr1::is_object< _Tp >`  
*is\_object*
- struct `std::tr1::is_pointer< _Tp >`  
*is\_pointer*
- struct `std::tr1::is_polymorphic< _Tp >`  
*is\_polymorphic*
- struct `std::tr1::is_same< typename, typename >`  
*is\_same*
- struct `std::tr1::is_scalar< _Tp >`  
*is\_scalar*
- struct `std::tr1::is_union< _Tp >`  
*is\_union*
- struct `std::tr1::is_void< _Tp >`  
*is\_void*
- struct `std::tr1::is_volatile< typename >`  
*is\_volatile*
- struct `std::tr1::rank< typename >`  
*rank*
- struct `std::tr1::remove_all_extents< _Tp >`  
*remove\_all\_extents*
- struct `std::tr1::remove_const< _Tp >`  
*remove\_const*
- struct `std::tr1::remove_cv< _Tp >`  
*remove\_cv*
- struct `std::tr1::remove_extent< _Tp >`  
*remove\_extent*
- struct `std::tr1::remove_pointer< _Tp >`  
*remove\_pointer*

- struct [std::tr1::remove\\_volatile< \\_Tp >](#)  
*remove\_volatile*

## Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

## Defines

- #define **\_DEFINE\_SPEC**(\_Order, \_Trait, \_Type, \_Value)
- #define **\_DEFINE\_SPEC\_0\_HELPER**
- #define **\_DEFINE\_SPEC\_1\_HELPER**
- #define **\_DEFINE\_SPEC\_2\_HELPER**

## Typedefs

- typedef integral\_constant< bool, false > [std::tr1::false\\_type](#)
- typedef integral\_constant< bool, true > [std::tr1::true\\_type](#)

### 6.341.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/type\\_traits](#).

## 6.342 type\_traits.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)

### Functions

- template<typename \_Type >  
bool [\\_\\_gnu\\_cxx::\\_\\_is\\_null\\_pointer](#) (\_Type \*\_\_ptr)
- template<typename \_Type >  
bool [\\_\\_gnu\\_cxx::\\_\\_is\\_null\\_pointer](#) (\_Type)

### 6.342.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [type\\_traits.h](#).

## 6.343 type\_utils.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### Defines

- #define `PB_DS_STATIC_ASSERT(UNIQUE, E)`

### Typedefs

- typedef [std::tr1::integral\\_constant](#)< int, 0 > `__gnu_pbds::detail::false_type`
- typedef [std::tr1::integral\\_constant](#)< int, 1 > `__gnu_pbds::detail::true_type`

### 6.343.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

Definition in file [type\\_utils.hpp](#).

## 6.344 typeinfo File Reference

### Classes

- class [std::bad\\_cast](#)  
*Thrown during incorrect typecasting.  
If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.*
- class [std::bad\\_typeid](#)  
*Thrown when a `NULL` pointer in a `typeid` expression is used.*
- class [std::type\\_info](#)

*Part of RTTI.*

## Namespaces

- namespace [std](#)

## Defines

- #define `__GXX_MERGED_TYPEINFO_NAMES`
- #define `__GXX_TYPEINFO_EQUALITY_INLINE`

## Functions

- `size_t std::_Hash_bytes` (const void \*\_\_ptr, size\_t \_\_len, size\_t \_\_seed)

### 6.344.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [typeinfo](#).

## 6.345 typelist.h File Reference

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [\\_\\_gnu\\_cxx::typelist](#)

### Defines

- #define `_GLIBCXX_TYPELIST_CHAIN1(X0)`
- #define `_GLIBCXX_TYPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- #define `_GLIBCXX_TYPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- #define `_GLIBCXX_TYPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- #define `_GLIBCXX_TYPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`

- `#define _GLIBCXX_TYPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TYPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

## Functions

- `template<typename Fn, typename Typelist >`  
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV >`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist >`  
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV >`  
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Gn, typename Typelist >`  
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`

### 6.345.1 Detailed Description

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

Definition in file [typelist.h](#).

## 6.346 types.h File Reference

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

## Namespaces

- namespace [\\_\\_gnu\\_parallel](#)



## Typedefs

- typedef int64\_t [\\_\\_gnu\\_parallel::\\_CASable](#)
- typedef uint64\_t [\\_\\_gnu\\_parallel::\\_SequenceIndex](#)
- typedef uint16\_t [\\_\\_gnu\\_parallel::\\_ThreadIndex](#)

## Enumerations

- enum [\\_\\_gnu\\_parallel::\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum [\\_\\_gnu\\_parallel::\\_FindAlgorithm](#) { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum [\\_\\_gnu\\_parallel::\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_\\_gnu\\_parallel::\\_Parallelism](#) {  
[\\_\\_gnu\\_parallel::sequential](#), [\\_\\_gnu\\_parallel::parallel\\_unbalanced](#), [\\_\\_gnu\\_parallel::parallel\\_balanced](#), [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop](#),  
[\\_\\_gnu\\_parallel::parallel\\_omp\\_loop\\_static](#), [\\_\\_gnu\\_parallel::parallel\\_taskqueue](#) }
- enum [\\_\\_gnu\\_parallel::\\_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [\\_\\_gnu\\_parallel::\\_SortAlgorithm](#) { **MWMS**, **QS**, **QS\_BALANCED** }
- enum [\\_\\_gnu\\_parallel::\\_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

## Variables

- static const int [\\_\\_gnu\\_parallel::\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_\\_gnu\\_parallel::\\_CASable\\_mask](#)

### 6.346.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [types.h](#).

## 6.347 types\_traits.hpp File Reference

### Namespaces

- namespace [\\_\\_gnu\\_pbds](#)

### 6.347.1 Detailed Description

Contains a traits class of types used by containers.

Definition in file [types\\_traits.hpp](#).

## 6.348 [unique\\_copy.h](#) File Reference

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

### Namespaces

- namespace [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_unique\_copy (_Iter __first, _Iter __`  
`__last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_unique\_copy (_Iter __first, _Iter __`  
`__last, _OutputIterator __result)`

### 6.348.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [unique\\_copy.h](#).

## 6.349 [unique\\_ptr.h](#) File Reference

### Classes

- struct [std::default\\_delete<\\_Tp>](#)  
*Primary template, [default\\_delete](#).*
- struct [std::default\\_delete<\\_Tp\[\]>](#)  
*Specialization, [default\\_delete](#).*

- struct `std::hash< unique_ptr< _Tp, _Dp > >`  
*std::hash specialization for `unique_ptr`.*
- class `std::unique_ptr< _Tp, _Dp >`  
20.7.12.2 *`unique_ptr` for single objects.*
- class `std::unique_ptr< _Tp[ ], _Dp >`  
20.7.12.3 *`unique_ptr` for array objects with a runtime length*

## Namespaces

- namespace `std`

## Functions

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`

- `template<typename _Tp , typename _Dp >`  
`void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &_`  
`__y)`

### 6.349.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unique\\_ptr.h](#).

## 6.350 unordered\_map File Reference

### 6.350.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered\\_map](#).

## 6.351 unordered\_map File Reference

### Classes

- class [std::\\_\\_debug::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Class [std::unordered\\_map](#) with safety/checking/debug instrumentation.*
- class [std::\\_\\_debug::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Class [std::unordered\\_multimap](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc`  
`>`

```

bool std::__debug::operator!= (const unordered_map< _Key, _Tp, _Hash, _-
Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
> &__y)
• template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc
>
bool std::__debug::operator!= (const unordered_multimap< _Key, _Tp, _-
Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash,
_Pred, _Alloc > &__y)
• template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc
>
bool std::__debug::operator== (const unordered_multimap< _Key, _Tp, _-
Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash,
_Pred, _Alloc > &__y)
• template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc
>
bool std::__debug::operator== (const unordered_map< _Key, _Tp, _Hash, _-
Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
> &__y)
• template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc
>
void std::__debug::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _-
Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__-
_y)
• template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc
>
void std::__debug::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
> &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)

```

### 6.351.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered\\_map](#).

## 6.352 unordered\_map File Reference

### Classes

- class [std::\\_\\_profile::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Class [std::unordered\\_map](#) wrapper with performance instrumentation.*
- class [std::\\_\\_profile::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)

Class [`std::unordered\_multimap`](#) wrapper with performance instrumentation.

## Namespaces

- namespace [`std`](#)
- namespace [`std::\_\_profile`](#)

## Defines

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

## Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
>  
bool std::__profile::operator!= (const unordered_map< _Key, _Tp, _Hash, _-  
Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc  
> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
>  
bool std::__profile::operator!= (const unordered_multimap< _Key, _Tp, _-  
Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash,  
_Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
>  
bool std::__profile::operator== (const unordered_multimap< _Key, _Tp, _-  
Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash,  
_Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
>  
bool std::__profile::operator== (const unordered_map< _Key, _Tp, _Hash, _-  
Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc  
> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>  
>  
void std::__profile::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _-  
Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &_-  
_y)`

- `template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc  
>  
void std::__profile::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc  
> &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`

### 6.352.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered\\_map](#).

## 6.353 unordered\_map.h File Reference

### Classes

- class [std::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)  
*A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.*
- class [std::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >](#)  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.*

### Namespaces

- namespace [std](#)

### Functions

- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc , bool __cache_hash_  
code>  
bool std::operator!= (const __unordered_map< _Key, _Tp, _Hash, _Pred, _  
Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _  
Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >  
bool std::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc  
> &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc , bool __cache_hash_  
code>  
bool std::operator!= (const __unordered_multimap< _Key, _Tp, _Hash, _Pred,`

```

 __Alloc, __cache_hash_code > &__x, const __unordered_multimap< _Key, _
 Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
 bool std::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred,
 _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
 > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
 bool std::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
 > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_
 code>
 bool std::operator== (const __unordered_map< _Key, _Tp, _Hash, _Pred, _-
 Alloc, __cache_hash_code > &__x, const __unordered_map< _Key, _Tp, _-
 Hash, _Pred, _Alloc, __cache_hash_code > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_
 code>
 bool std::operator== (const __unordered_multimap< _Key, _Tp, _Hash, _-
 Pred, _Alloc, __cache_hash_code > &__x, const __unordered_multimap< _-
 Key, _Tp, _Hash, _Pred, _Alloc, __cache_hash_code > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
 bool std::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred,
 _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
 > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_
 code>
 void std::swap (__unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc, __-
 cache_hash_code > &__x, __unordered_multimap< _Key, _Tp, _Hash, _Pred,
 _Alloc, __cache_hash_code > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc, bool __cache_hash_
 code>
 void std::swap (__unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc, __-
 cache_hash_code > &__x, __unordered_map< _Key, _Tp, _Hash, _Pred, _-
 Alloc, __cache_hash_code > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
 void std::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x,
 unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
• template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >
 void std::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &-
 __x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)

```

### 6.353.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.



Definition in file [unordered\\_map.h](#).

## 6.354 unordered\_set File Reference

### 6.354.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [unordered\\_set](#).

## 6.355 unordered\_set File Reference

### Classes

- class [std::\\_\\_debug::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Class [std::unordered\\_multiset](#) with safety/checking/debug instrumentation.*
- class [std::\\_\\_debug::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Class [std::unordered\\_set](#) with safety/checking/debug instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

### Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__debug::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__debug::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool std::__debug::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__debug::operator== (const unordered_set< _Value, _Hash, _Pred,`  
`_Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void std::__debug::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc`  
`> &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void std::__debug::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &_`  
`__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

### 6.355.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/unordered\\_set](#).

## 6.356 unordered\_set File Reference

### Classes

- class [std::\\_\\_profile::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Unordered\_multiset wrapper with performance instrumentation.*
- class [std::\\_\\_profile::unordered\\_set< \\_Key, \\_Hash, \\_Pred, \\_Alloc >](#)  
*Unordered\_set wrapper with performance instrumentation.*

### Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

### Defines

- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_BASE`
- `#define _GLIBCXX_STD_BASE`
- `#define _GLIBCXX_STD_BASE`

## Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool std::__profile::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void std::__profile::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void std::__profile::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`

### 6.356.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/unordered\\_set](#).

## 6.357 unordered\_set.h File Reference

### Classes

- class [std::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.*
- class [std::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >](#)  
*A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.*

## Namespaces

- namespace [std](#)

## Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool std::operator!= (const __unordered_set< _Value, _Hash, _Pred, _Alloc,`  
`__cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred,`  
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool std::operator!= (const __unordered_multiset< _Value, _Hash, _Pred, _-`  
`_Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _-`  
`_Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc`  
`> &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc >`  
`&__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool std::operator== (const __unordered_set< _Value, _Hash, _Pred, _Alloc,`  
`__cache_hash_code > &__x, const __unordered_set< _Value, _Hash, _Pred,`  
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`bool std::operator== (const __unordered_multiset< _Value, _Hash, _Pred, _-`  
`_Alloc, __cache_hash_code > &__x, const __unordered_multiset< _Value, _-`  
`_Hash, _Pred, _Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool std::operator== (const unordered_multiset< _Value, _Hash, _Pred, _-`  
`_Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &_-`  
`__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`void std::swap (__unordered_multiset< _Value, _Hash, _Pred, _Alloc, __-`  
`cache_hash_code > &__x, __unordered_multiset< _Value, _Hash, _Pred, _-`  
`_Alloc, __cache_hash_code > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc, bool __cache_hash_code>`  
`void std::swap (__unordered_set< _Value, _Hash, _Pred, _Alloc, __cache_-`  
`hash_code > &__x, __unordered_set< _Value, _Hash, _Pred, _Alloc, __-`  
`cache_hash_code > &__y)`

- `template<class _Value , class _Hash , class _Pred , class _Alloc >`  
`void std::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x,`  
`unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value , class _Hash , class _Pred , class _Alloc >`  
`void std::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x,`  
`unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`

### 6.357.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [unordered\\_set.h](#).

## 6.358 utility File Reference

### 6.358.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [utility](#).

## 6.359 utility File Reference

### Namespaces

- namespace [std](#)
- namespace [std::tr1](#)

### Functions

- `template<std::size_t _Int, class _Tp1 , class _Tp2 >`  
`tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::tr1::get`  
`(std::pair< _Tp1, _Tp2 > &__in)`
- `template<std::size_t _Int, class _Tp1 , class _Tp2 >`  
`const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::tr1::get`  
`(const std::pair< _Tp1, _Tp2 > &__in)`

### 6.359.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [tr1\\_impl/utility](#).

## 6.360 valarray File Reference

### Classes

- class [std::valarray<\\_Tp>](#)  
*Smart array designed to support numeric processing.*

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

### Functions

- `template<class _Tp>`  
`_Tp * std::begin (valarray<_Tp> &__va)`
- `template<class _Tp>`  
`const _Tp * std::begin (const valarray<_Tp> &__va)`
- `template<class _Tp>`  
`_Tp * std::end (valarray<_Tp> &__va)`
- `template<class _Tp>`  
`const _Tp * std::end (const valarray<_Tp> &__va)`
- `template<typename _Tp>`  
`_Expr<_BinClos<__not_equal_to, _ValArray, _Constant, _Tp, _Tp>, type-name __fun<__not_equal_to, _Tp>::result_type> std::operator!= (const valarray<_Tp> &__v, const _Tp &__t)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const _`  
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __modulus, _Tp >::result_type > std::operator% (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const _Tp`  
`&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_and, _Tp >::result_type > std::operator& (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const _`  
`Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __logical_and, _Tp >::result_type > std::operator&& (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > std::operator* (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`__fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __plus, _Tp >::result_type > std::operator+ (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`__fun< __minus, _Tp >::result_type > std::operator- (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __minus, _Tp >::result_type > std::operator- (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`__fun< __divides, _Tp >::result_type > std::operator/ (const valarray< _Tp >`  
`&__v, const _Tp &__t)`



- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename _-`  
`_fun< __divides, _Tp >::result_type > std::operator/ (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename _-`  
`_fun< __less, _Tp >::result_type > std::operator< (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename _-`  
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`  
`&__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename _-`  
`_fun< __less, _Tp >::result_type > std::operator< (const valarray< _Tp >`  
`&__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __shift_left, _Tp >::result_type > std::operator<< (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __less_equal, _Tp >::result_type > std::operator<= (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _-`  
`Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __equal_to, _Tp >::result_type > std::operator== (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename _`  
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`  
`> &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename _`  
`__fun< __greater, _Tp >::result_type > std::operator> (const _Tp &__t, const`  
`valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`_Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __greater_equal, _Tp >::result_type > std::operator>= (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`  
`_Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray<`  
`_Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`  
`valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const _Tp`  
`&__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, type-`  
`name __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const`  
`valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __bitwise_or, _Tp >::result_type > std::operator| (const _Tp &__t,`  
`const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`  
`Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _`  
`Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename`  
`__fun< __logical_or, _Tp >::result_type > std::operator|| (const _Tp &__t,`  
`const valarray< _Tp > &__v)`

### 6.360.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [valarray](#).

## 6.361 valarray\_after.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- #define **\_DEFINE\_EXPR\_BINARY\_FUNCTION**(\_Fun, \_UFun)
- #define **\_DEFINE\_EXPR\_BINARY\_OPERATOR**(\_Op, \_Name)
- #define **\_DEFINE\_EXPR\_UNARY\_FUNCTION**(\_Name, \_UName)
- #define **\_DEFINE\_EXPR\_UNARY\_OPERATOR**(\_Op, \_Name)

### Functions

- template<class \_Dom >  
\_Expr< \_UnClos< \_Abs, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**std::abs** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Abs, \_ValArray, \_Tp >, \_Tp > **std::abs** (const valarray< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Acos, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**std::acos** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Acos, \_ValArray, \_Tp >, \_Tp > **std::acos** (const valarray< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Asin, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**std::asin** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Asin, \_ValArray, \_Tp >, \_Tp > **std::asin** (const valarray< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< \_Atan, \_Expr, \_Dom >, typename \_Dom::value\_type >  
**std::atan** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
\_Expr< \_UnClos< \_Atan, \_ValArray, \_Tp >, \_Tp > **std::atan** (const valarray< \_Tp > &\_\_v)
- template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_Atan2, \_Expr, \_Expr, \_Dom1, \_Dom2 >, type-  
name \_Dom1::value\_type > **std::atan2** (const \_Expr< \_Dom1, typename \_-  
Dom1::value\_type > &\_\_e1, const \_Expr< \_Dom2, typename \_Dom2::value\_-  
type > &\_\_e2)

- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename _Dom::value_type > std::atan2 (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _-`  
`Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _-`  
`Dom >, typename _Dom::value_type > std::atan2 (const valarray< typename`  
`_Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_`  
`type >, typename _Dom::value_type > std::atan2 (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type,`  
`_Dom >, typename _Dom::value_type > std::atan2 (const typename _-`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp >`  
`std::atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp >`  
`std::atan2 (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp >`  
`std::atan2 (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type >`  
`std::cos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray<`  
`_Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type >`  
`std::cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray<`  
`_Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type >`  
`std::exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray<`  
`_Tp > &__v)`

- `template<typename _Tp >`  
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > std::log (const valarray<`  
`_Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type >`  
`std::log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type >`  
`std::log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > std::log10 (const`  
`valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename`  
`_Dom::value_type >::result_type > std::operator!= (const typename _-`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`  
`&__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename`  
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`  
`Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type`  
`> &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename`  
`_Dom::value_type >::result_type > std::operator!= (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename`  
`_Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __not_equal_to, typename _Dom1::value_type >::result_type >`  
`std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename`  
`_Dom::value_type >, typename __fun< __not_equal_to, typename _-`  
`Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value -`  
`type >::result_type > std::operator% (const _Expr< _Dom, typename _-`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __modulus, typename _Dom1::value_type >::result_type >`  
`std::operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__`  
`v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`  
`>::result_type > std::operator% (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __modulus, typename _Dom::value_`  
`type >::result_type > std::operator% (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __modulus, typename _Dom::value_type`  
`>::result_type > std::operator% (const valarray< typename _Dom::value_`  
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename`  
`_Dom::value_type >, typename __fun< __bitwise_and, typename _`  
`Dom::value_type >::result_type > std::operator& (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`  
`>::result_type > std::operator& (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename`  
`_Dom::value_type >, typename __fun< __bitwise_and, typename _`  
`Dom::value_type >::result_type > std::operator& (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type`  
`> &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type`  
`>::result_type > std::operator& (const valarray< typename _Dom::value_type`  
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __bitwise_and, typename _Dom1::value_type >::result_type >`

**std::operator&** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)

- template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_\_logical\_and, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_logical\_and, typename \_Dom1::value\_type >::result\_type >  
**std::operator&&** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_logical\_and, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_logical\_and, typename \_Dom::value\_type >::result\_type > **std::operator&&** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_logical\_and, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_logical\_and, typename \_Dom::value\_type >::result\_type > **std::operator&&** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_logical\_and, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_logical\_and, typename \_Dom::value\_type >::result\_type > **std::operator&&** (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_logical\_and, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_logical\_and, typename \_Dom::value\_type >::result\_type > **std::operator&&** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_\_multiplies, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_multiplies, typename \_Dom1::value\_type >::result\_type >  
**std::operator\*** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_multiplies, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_multiplies, typename \_Dom::value\_type >::result\_type > **std::operator\*** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_multiplies, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_multiplies, typename \_Dom::value\_type >::result\_type > **std::operator\*** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_multiplies, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_multiplies, typename \_Dom::value\_type >::result\_type > **std::operator\*** (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)



Dom::value\_type >, typename \_\_fun< \_\_multiplies, typename \_Dom::value\_type >::result\_type > **std::operator\*** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)

- template<class \_Dom >  
\_Expr< \_BinClos< \_\_multiplies, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_multiplies, typename \_Dom::value\_type >::result\_type > **std::operator\*** (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_plus, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **std::operator+** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_plus, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **std::operator+** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_plus, \_ValArray, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **std::operator+** (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_\_plus, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_plus, typename \_Dom1::value\_type >::result\_type > **std::operator+** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_plus, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_plus, typename \_Dom::value\_type >::result\_type > **std::operator+** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_minus, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_minus, typename \_Dom::value\_type >::result\_type > **std::operator-** (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_minus, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_minus, typename \_Dom::value\_type >::result\_type > **std::operator-** (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)

- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type,`  
`_Dom >, typename __fun< __minus, typename _Dom::value_type >::result_`  
`type > std::operator- (const valarray< typename _Dom::value_type > &__v,`  
`const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __`  
`fun< __minus, typename _Dom1::value_type >::result_type > std::operator-`  
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __minus, typename _Dom::value_`  
`type >::result_type > std::operator- (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __`  
`fun< __divides, typename _Dom1::value_type >::result_type > std::operator/`  
`(const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr<`  
`_Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __divides, typename _Dom::value_`  
`type >::result_type > std::operator/ (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type,`  
`_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_`  
`type > std::operator/ (const typename _Dom::value_type &__t, const _Expr<`  
`_Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __divides, typename _Dom::value_`  
`type >::result_type > std::operator/ (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`  
`__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type,`  
`_Dom >, typename __fun< __divides, typename _Dom::value_type >::result_`  
`type > std::operator/ (const valarray< typename _Dom::value_type > &__v,`  
`const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun<`  
`__less, typename _Dom1::value_type >::result_type > std::operator< (const`  
`_Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2,`  
`typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_`  
`type >, typename __fun< __less, typename _Dom::value_type >::result_type`  
`> std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__v,`  
`const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _`  
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`  
`> std::operator< (const typename _Dom::value_type &__t, const _Expr< _`  
`Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_`  
`type >, typename __fun< __less, typename _Dom::value_type >::result_type`  
`> std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__e,`  
`const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _`  
`Dom >, typename __fun< __less, typename _Dom::value_type >::result_type`  
`> std::operator< (const valarray< typename _Dom::value_type > &__v, const`  
`_Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __shift_left, typename _Dom1::value_type >::result_type >`  
`std::operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &_`  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`  
`type >::result_type > std::operator<< (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`  
`>::result_type > std::operator<< (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_`  
`type >::result_type > std::operator<< (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`  
`__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type`  
`>::result_type > std::operator<< (const valarray< typename _Dom::value_`  
`type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __less_equal, typename _Dom1::value_type >::result_type >`  
`std::operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &_`  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &_w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`  
`type >::result_type > std::operator<= (const _Expr< _Dom, typename _`  
`Dom::value_type > &_v, const typename _Dom::value_type &_t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_`  
`type >::result_type > std::operator<= (const _Expr< _Dom, typename _`  
`Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`  
`>::result_type > std::operator<= (const typename _Dom::value_type &_t,`  
`const _Expr< _Dom, typename _Dom::value_type > &_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __less_equal, typename _Dom::value_type`  
`>::result_type > std::operator<= (const valarray< typename _Dom::value_`  
`type > &_v, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_`  
`type >::result_type > std::operator== (const _Expr< _Dom, typename _`  
`Dom::value_type > &_e, const valarray< typename _Dom::value_type > &_`  
`_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`  
`>::result_type > std::operator== (const valarray< typename _Dom::value_`  
`type > &_v, const _Expr< _Dom, typename _Dom::value_type > &_e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type`  
`>::result_type > std::operator== (const typename _Dom::value_type &_t,`  
`const _Expr< _Dom, typename _Dom::value_type > &_v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __equal_to, typename _Dom1::value_type >::result_type >`

**std::operator==** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_  
\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)

- template<class \_Dom >  
\_Expr< \_BinClos< \_\_equal\_to, \_Expr, \_Constant, \_Dom, typename \_  
Dom::value\_type >, typename \_\_fun< \_\_equal\_to, typename \_Dom::value\_  
type >::result\_type > **std::operator==** (const \_Expr< \_Dom, typename \_  
Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater, \_Expr, \_Constant, \_Dom, typename \_  
Dom::value\_type >, typename \_\_fun< \_\_greater, typename \_Dom::value\_  
type >::result\_type > **std::operator>** (const \_Expr< \_Dom, typename \_  
Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater, \_Expr, \_ValArray, \_Dom, typename \_  
Dom::value\_type >, typename \_\_fun< \_\_greater, typename \_Dom::value\_  
type >::result\_type > **std::operator>** (const \_Expr< \_Dom, typename \_  
Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_  
\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater, \_ValArray, \_Expr, typename \_Dom::value\_type,  
\_Dom >, typename \_\_fun< \_\_greater, typename \_Dom::value\_type >::result\_  
type > **std::operator>** (const valarray< typename \_Dom::value\_type > &\_\_v,  
const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- template<class \_Dom1, class \_Dom2 >  
\_Expr< \_BinClos< \_\_greater, \_Expr, \_Expr, \_Dom1, \_Dom2 >, type-  
name \_\_fun< \_\_greater, typename \_Dom1::value\_type >::result\_type >  
**std::operator>** (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_  
\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater, \_Constant, \_Expr, typename \_Dom::value\_type,  
\_Dom >, typename \_\_fun< \_\_greater, typename \_Dom::value\_type >::result\_  
type > **std::operator>** (const typename \_Dom::value\_type &\_\_t, const \_Expr<  
\_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater\_equal, \_Expr, \_ValArray, \_Dom, typename  
\_Dom::value\_type >, typename \_\_fun< \_\_greater\_equal, typename \_  
Dom::value\_type >::result\_type > **std::operator>=** (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_  
type > &\_\_v)
- template<class \_Dom >  
\_Expr< \_BinClos< \_\_greater\_equal, \_Constant, \_Expr, typename \_  
Dom::value\_type, \_Dom >, typename \_\_fun< \_\_greater\_equal, typename \_  
Dom::value\_type >::result\_type > **std::operator>=** (const typename \_  
Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type >  
&\_\_v)

- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _-`  
`Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _-`  
`Dom::value_type >::result_type > std::operator>=` (const valarray< typename  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type`  
`> &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __greater_equal, typename _Dom1::value_type >::result_type >`  
`std::operator>=` (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename`  
`_Dom::value_type >, typename __fun< __greater_equal, typename _-`  
`Dom::value_type >::result_type > std::operator>=` (const \_Expr< \_Dom,  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &_  
_t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name __fun< __shift_right, typename _Dom1::value_type >::result_type >`  
`std::operator>>` (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_  
`_v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value`  
`-type >::result_type > std::operator>>` (const \_Expr< \_Dom, typename \_  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value`  
`-type, _Dom >, typename __fun< __shift_right, typename _Dom::value`  
`-type >::result_type > std::operator>>` (const typename \_Dom::value\_type &\_\_t,  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _-`  
`Dom::value_type >, typename __fun< __shift_right, typename _Dom::value`  
`-type >::result_type > std::operator>>` (const \_Expr< \_Dom, typename \_  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_  
_v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value`  
`-type, _Dom >, typename __fun< __shift_right, typename _Dom::value`  
`-type >::result_type > std::operator>>` (const valarray< typename \_Dom::value  
`-type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value`  
`-type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value`

>::result\_type > **std::operator**<sup>^</sup> (const valarray< typename \_Dom::value\_type > &\_\_v, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)

- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_xor, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **std::operator**<sup>^</sup> (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom1, class \_Dom2 >  
 \_Expr< \_BinClos< \_\_bitwise\_xor, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_bitwise\_xor, typename \_Dom1::value\_type >::result\_type > **std::operator**<sup>^</sup> (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_xor, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **std::operator**<sup>^</sup> (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_xor, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_bitwise\_xor, typename \_Dom::value\_type >::result\_type > **std::operator**<sup>^</sup> (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom1, class \_Dom2 >  
 \_Expr< \_BinClos< \_\_bitwise\_or, \_Expr, \_Expr, \_Dom1, \_Dom2 >, typename \_\_fun< \_\_bitwise\_or, typename \_Dom1::value\_type >::result\_type > **std::operator**| (const \_Expr< \_Dom1, typename \_Dom1::value\_type > &\_\_v, const \_Expr< \_Dom2, typename \_Dom2::value\_type > &\_\_w)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_or, \_Expr, \_Constant, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **std::operator**| (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v, const typename \_Dom::value\_type &\_\_t)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_or, \_Expr, \_ValArray, \_Dom, typename \_Dom::value\_type >, typename \_\_fun< \_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **std::operator**| (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e, const valarray< typename \_Dom::value\_type > &\_\_v)
- template<class \_Dom >  
 \_Expr< \_BinClos< \_\_bitwise\_or, \_Constant, \_Expr, typename \_Dom::value\_type, \_Dom >, typename \_\_fun< \_\_bitwise\_or, typename \_Dom::value\_type >::result\_type > **std::operator**| (const typename \_Dom::value\_type &\_\_t, const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_v)

- `template<class _Dom >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __bitwise_or, typename _Dom::value_type`  
`>::result_type > std::operator| (const valarray< typename _Dom::value_type`  
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`  
`>::result_type > std::operator|| (const valarray< typename _Dom::value_type`  
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _`  
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_`  
`type >::result_type > std::operator|| (const _Expr< _Dom, typename _`  
`Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _`  
`Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_`  
`type >::result_type > std::operator|| (const _Expr< _Dom, typename _`  
`Dom::value_type > &__e, const valarray< typename _Dom::value_type > &_`  
`__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_`  
`type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type`  
`>::result_type > std::operator|| (const typename _Dom::value_type &__t,`  
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, type`  
`name __fun< __logical_or, typename _Dom1::value_type >::result_type >`  
`std::operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`  
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _`  
`Dom >, typename _Dom::value_type > std::pow (const valarray< typename`  
`_Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type,`  
`_Dom >, typename _Dom::value_type > std::pow (const typename _`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow`  
`(const valarray< _Tp > &__v, const _Tp &__t)`



- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_`  
`type >, typename _Dom::value_type > std::pow (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow`  
`(const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_`  
`type >, typename _Dom::value_type > std::pow (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type`  
`> &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, type-`  
`name _Dom1::value_type > std::pow (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_`  
`type > &__e2)`
- `template<typename _Tp >`  
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow`  
`(const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type >`  
`std::sin (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray<`  
`_Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type >`  
`std::sinh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray<`  
`_Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray<`  
`_Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type >`  
`std::sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray<`  
`_Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type >`  
`std::tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom >`  
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type >`  
`std::tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray<`  
`_Tp > &__v)`

### 6.361.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray\\_after.h](#).

## 6.362 valarray\_array.h File Reference

### Namespaces

- namespace [std](#)

### Defines

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

### Functions

- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`  
`restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t`  
`__s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__-`  
`restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _-`  
`Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _-`  
`Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`  
`__b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t`  
`> __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__-`  
`restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const`  
`size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s,`  
`_Tp *__restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __-`  
`b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp >`  
`__b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t`  
`__n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const`  
`size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n,`  
`size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp`  
`*__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`

- `template<typename _Tp >`  
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void * std::__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`  
`_Tp *__restrict std::__valarray_get_storage (size_t __n)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_min (const _Ta &__a)`
- `template<typename _Tp >`  
`_Tp std::__valarray_product (const _Tp *__f, const _Tp *__l)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_and (_Array< _Tp > __a, size_t __-`  
`n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_and (_Array< _Tp > __a, size_t __-`  
`n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_and (_Array< _Tp > __a, size_t __-`  
`n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___bitwise_and (_Array< _Tp > __a, const _-`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_and (_Array< _Tp > __a, size_t __-`  
`n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_and (_Array< _Tp > __a, _Array<`  
`_Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___bitwise_and (_Array< _Tp > __a, size_t __-`  
`s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_and (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, _Array<`  
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, const _-`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, _Array< _-`  
`Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array<`  
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, const _`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`  
`_Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`  
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array<`  
`bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp`  
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`  
`> __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const`  
`_Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t`  
`> __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr<`  
`_Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp`  
`> __b, size_t __n, size_t __s)`



- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`  
`> __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool`  
`> __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< _`  
`Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`  
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, const _-`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array<`  
`bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __s,`  
`const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool >`  
`__m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __s, const`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, const`  
`_Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp >`  
`__b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, const Expr< _-`  
`Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t >`  
`__i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t >`  
`__i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _-`  
`Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t`  
`__s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool >`  
`__m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, const _`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array<`  
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array<`  
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp`  
`> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array<`  
`bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, _Array<`  
`_Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, _Array<`  
`bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, _Array<`  
`size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, const _`  
`Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`  
`_Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, _Array<`  
`size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`  
`size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, size_t __s,`  
`const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::_Array_augmented__shift_right (_Array< _Tp > __a, size_t __n,`  
`const _Tp &__t)`

### 6.362.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray\\_array.h](#).

## 6.363 valarray\_array.tcc File Reference

### Namespaces

- namespace [std](#)

### Functions

- `template<typename _Tp >  
void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >  
void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp >  
void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >  
void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >  
void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom >  
void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >  
void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp >  
void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >  
void std::__valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >  
void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >  
void std::__valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`

### 6.363.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray\\_array.tcc](#).

## 6.364 valarray\_before.h File Reference

### Namespaces

- namespace [std](#)

### 6.364.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [valarray\\_before.h](#).

## 6.365 vector File Reference

### 6.365.1 Detailed Description

This is a Standard C++ Library header.

Definition in file [vector](#).

## 6.366 vector File Reference

### Classes

- class [std::\\_\\_debug::vector< \\_Tp, \\_Allocator >](#)  
*Class [std::vector](#) with safety/checking/debug instrumentation.*
- struct [std::hash< \\_\\_debug::vector< bool, \\_Alloc > >](#)  
*[std::hash](#) specialization for [vector<bool>](#).*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_debug](#)

## Functions

- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator!= (const vector< _Tp, _Alloc > &__lhs, const  
vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator< (const vector< _Tp, _Alloc > &__lhs, const  
vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator<= (const vector< _Tp, _Alloc > &__lhs, const  
vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator== (const vector< _Tp, _Alloc > &__lhs, const  
vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator> (const vector< _Tp, _Alloc > &__lhs, const  
vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator>= (const vector< _Tp, _Alloc > &__lhs, const  
vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
void std::__debug::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc  
> &__rhs)`

### 6.366.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file [debug/vector](#).

## 6.367 vector File Reference

### Classes

- struct [std::hash< \\_\\_profile::vector< bool, \\_Alloc > >](#)  
*[std::hash](#) specialization for `vector<bool>`.*

## Namespaces

- namespace [std](#)
- namespace [std::\\_\\_profile](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__profile::operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__profile::swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)`

### 6.367.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file [profile/vector](#).



## 6.368 vector.tcc File Reference

### Namespaces

- namespace [std](#)

### 6.368.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vector.tcc](#).

## 6.369 vstring.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>](#)

*Template class [\\_\\_versa\\_string](#).*

*Data structure managing sequences of characters and character-like objects.*

### Namespaces

- namespace [\\_\\_gnu\\_cxx](#)
- namespace [std](#)

### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

```

bool __gnu_cxx::operator!= (const _CharT *__lhs, const __versa_string< _-
_CharT, _Traits, _Alloc, _Base > &__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _-
_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-
rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _-
_Base > &__lhs, const _CharT *__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__-
rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_-
string< _CharT, _Traits, _Alloc, _Base > &__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const
__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const
_CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-
rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_-
_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _-
_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-
rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _-
_Base > &__lhs, const _CharT *__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>

```

- ```
bool __gnu_cxx::operator< (const _CharT *__lhs, const __versa_string< _-
_CharT, _Traits, _Alloc, _Base > &__rhs)
```
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _-`
`CharT, _Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits,`
`_Alloc, _Base > &__str)`
 - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc,`
`_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-`
`__rhs)`
 - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc,`
`_Base > &__lhs, const _CharT *__rhs)`
 - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator<= (const _CharT *__lhs, const __versa_string< _-`
`CharT, _Traits, _Alloc, _Base > &__rhs)`
 - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator== (const _CharT *__lhs, const __versa_string< _-`
`CharT, _Traits, _Alloc, _Base > &__rhs)`
 - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _-`
`Base > &__lhs, const _CharT *__rhs)`
 - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc,`
`_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-`
`__rhs)`
 - `template<typename _CharT, template< typename, typename, typename > class _Base>`
`__enable_if< std::__is_char< _CharT >::__value, bool >::__type __gnu_-`
`cxx::operator== (const __versa_string< _CharT, std::char_traits< _CharT >,`
`std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT,`
`std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
 - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _-`
`Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_-`
`rhs)`
 - `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`

```

bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _-
Base > &__lhs, const _CharT *__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator> (const _CharT *__lhs, const __versa_string< _-
CharT, _Traits, _Alloc, _Base > &__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc,
_Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-
rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc,
_Base > &__lhs, const _CharT *__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (const _CharT *__lhs, const __versa_string< _-
CharT, _Traits, _Alloc, _Base > &__rhs)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,
_Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base
> &__str)
• double __gnu_cxx::stod (const __vstring &__str, std::size_t *__idx=0)
• float __gnu_cxx::stof (const __vstring &__str, std::size_t *__idx=0)
• int __gnu_cxx::stoi (const __vstring &__str, std::size_t *__idx=0, int __-
base=10)
• long __gnu_cxx::stol (const __vstring &__str, std::size_t *__idx=0, int __-
base=10)
• long double __gnu_cxx::stold (const __vstring &__str, std::size_t *__idx=0)
• long long __gnu_cxx::stoll (const __vstring &__str, std::size_t *__idx=0, int
__base=10)
• unsigned long __gnu_cxx::stoul (const __vstring &__str, std::size_t *__idx=0,
int __base=10)
• unsigned long long __gnu_cxx::stoull (const __vstring &__str, std::size_t *__-
idx, int __base=10)
• template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__-
lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
• __vstring __gnu_cxx::to_string (long long __val)
• __vstring __gnu_cxx::to_string (unsigned __val)
• __vstring __gnu_cxx::to_string (int __val)

```

- `__vstring __gnu_cxx::to_string` (long `__val`)
- `__vstring __gnu_cxx::to_string` (unsigned long `__val`)
- `__vstring __gnu_cxx::to_string` (float `__val`)
- `__vstring __gnu_cxx::to_string` (unsigned long long `__val`)
- `__vstring __gnu_cxx::to_string` (long double `__val`)
- `__vstring __gnu_cxx::to_string` (double `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned `__val`)
- `__wvstring __gnu_cxx::to_wstring` (unsigned long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long long `__val`)
- `__wvstring __gnu_cxx::to_wstring` (long double `__val`)
- `__wvstring __gnu_cxx::to_wstring` (double `__val`)
- `__wvstring __gnu_cxx::to_wstring` (float `__val`)
- `__wvstring __gnu_cxx::to_wstring` (int `__val`)

6.369.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file [vstring.h](#).

6.370 `vstring.tcc` File Reference

Namespaces

- namespace `__gnu_cxx`
- namespace `std`

Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & std::getline` (`basic_istream< _CharT, _-`
`Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`
`&__str, _CharT __delim`)
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+` (`_-`
`CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs`)

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`
`_CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__-`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__-`
`rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const`
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_-`
`string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT,`
`_Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base`
`> &__str)`

6.370.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring.tcc](#).

6.371 vstring_fwd.h File Reference

Namespaces

- namespace [__gnu_cxx](#)

Typedefs

- typedef `__versa_string< char, std::char_traits< char >, std::allocator< char >,`
`__rc_string_base > __gnu_cxx::__rc_string`
- typedef `__vstring __gnu_cxx::__sso_string`

- `typedef __versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base > __gnu_cxx::__u16rc_string`
- `typedef __u16vstring __gnu_cxx::__u16sso_string`
- `typedef __versa_string< char16_t > __gnu_cxx::__u16vstring`
- `typedef __versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base > __gnu_cxx::__u32rc_string`
- `typedef __u32vstring __gnu_cxx::__u32sso_string`
- `typedef __versa_string< char32_t > __gnu_cxx::__u32vstring`
- `typedef __versa_string< char > __gnu_cxx::__vstring`
- `typedef __versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base > __gnu_cxx::__wrc_string`
- `typedef __wvstring __gnu_cxx::__wsso_string`
- `typedef __versa_string< wchar_t > __gnu_cxx::__wvstring`

6.371.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring_fwd.h](#).

6.372 `vstring_util.h` File Reference

Namespaces

- namespace [__gnu_cxx](#)

6.372.1 Detailed Description

This file is a GNU extension to the Standard C++ Library. This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vstring_util.h](#).

6.373 `workstealing.h` File Reference

Parallelization of embarrassingly parallel execution by means of work-stealing.

Classes

- struct [__gnu_parallel::__Job<_DifferenceTp>](#)
One __job for a certain thread.

Namespaces

- namespace [__gnu_parallel](#)

Defines

- #define `_GLIBCXX_JOB_VOLATILE`

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
[__gnu_parallel::__for_each_template_random_access_workstealing](#) (_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits<_RAIter>::difference_type __bound)

6.373.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing. Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file [workstealing.h](#).

Index

~_LoserTreeBase
 __gnu_parallel::_LoserTreeBase, 1180
 ~_RestrictedBoundedConcurrentQueue
 __gnu_parallel::_RestrictedBoundedConcurrentQueue, 1214
 ~_Safe_sequence_base
 __gnu_debug::_Safe_sequence_base, 1048
 ~__versa_string
 __gnu_cxx::__versa_string, 806
 ~auto_ptr
 std::auto_ptr, 1510
 ~basic_filebuf
 std::basic_filebuf, 1529
 ~basic_fstream
 std::basic_fstream, 1570
 ~basic_ifstream
 std::basic_ifstream, 1633
 ~basic_ios
 std::basic_ios, 1684
 ~basic_iostream
 std::basic_iostream, 1720
 ~basic_istream
 std::basic_istream, 1781
 ~basic_istreamstream
 std::basic_istreamstream, 1835
 ~basic_ofstream
 std::basic_ofstream, 1888
 ~basic_ostream
 std::basic_ostream, 1931
 ~basic_ostringstream
 std::basic_ostringstream, 1976
 ~basic_regex
 std::basic_regex, 2013
 ~basic_streambuf
 std::basic_streambuf, 2025
 ~basic_string
 std::basic_string, 2053
 ~basic_stringstream
 std::basic_stringstream, 2144
 ~collate
 std::collate, 2275
 ~ctype
 std::ctype< char >, 2316
 std::ctype< wchar_t >, 2331
 ~deque
 std::deque, 2394
 ~facet
 std::locale::facet, 2640
 ~forward_list
 std::forward_list, 2456
 ~gslice
 numeric_arrays, 91
 ~ios_base
 std::ios_base, 2557
 ~locale
 std::locale, 2633
 ~match_results
 std::match_results, 2684
 ~messages
 std::messages, 2701
 ~money_get
 std::money_get, 2714
 ~money_put
 std::money_put, 2719
 ~moneypunct
 std::moneypunct, 2727
 ~num_get
 std::num_get, 2806
 ~num_put
 std::num_put, 2826
 ~numpunct

- std::numpunct, [2872](#)
- ~sentry
 - std::basic_ostream::sentry, [1963](#)
- ~stdio_filebuf
 - __gnu_cxx::stdio_filebuf, [958](#)
- ~temporary_buffer
 - __gnu_cxx::temporary_buffer, [1012](#)
- ~time_get
 - std::time_get, [3043](#)
- ~time_put
 - std::time_put, [3065](#)
- ~type_info
 - std::type_info, [3116](#)
- ~vector
 - std::vector, [3155](#)
- /mnt/share/src/ Directory Reference, [339](#)
- /mnt/share/src/gcc.svn-trunk/ Directory Reference, [325](#)
- /mnt/share/src/gcc.svn-trunk/libstdc++-v3/ Directory Reference, [331](#)
- /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/ Directory Reference, [321](#)
- /mnt/share/src/gcc.svn-trunk/libstdc++-v3/doc/doxygen/ Directory Reference, [322](#)
- /mnt/share/src/gcc.svn-trunk/libstdc++-v3/libsupc++/ Directory Reference, [332](#)
- __gnu_parallel
 - parallel_balanced, [407](#)
 - parallel_omp_loop, [408](#)
 - parallel_omp_loop_static, [408](#)
 - parallel_taskqueue, [408](#)
 - parallel_unbalanced, [407](#)
 - sequential, [407](#)
- _AlgorithmStrategy
 - __gnu_parallel, [407](#)
- _BALLOC_ALIGN_BYTES
 - bitmap_allocator.h, [3247](#)
- _BinIndex
 - __gnu_parallel, [406](#)
- _Bit_scan_forward
 - __gnu_cxx, [366](#)
- _CASable
 - __gnu_parallel, [406](#)
- _CASable_bits
 - __gnu_parallel, [454](#)
- _CASable_mask
 - __gnu_parallel, [454](#)
- _Construct
 - std, [619](#)
- _DRandomShufflingGlobalData
 - __gnu_parallel::_-DRandomShufflingGlobalData, [1149](#)
- _Destroy
 - std, [620](#)
- _FindAlgorithm
 - __gnu_parallel, [407](#)
- _Find_first
 - SGIextensions, [17](#)
- _Find_next
 - SGIextensions, [17](#)
- _GLIBCXX_ASSERTIONS
 - compiletime_settings.h, [3280](#)
- _GLIBCXX_ATOMIC_PROPERTY
 - atomics, [210](#)
- _GLIBCXX_BAL_QUICKSORT
 - features.h, [3328](#)
- _GLIBCXX_CALL
 - compiletime_settings.h, [3280](#)
- _GLIBCXX_DEBUG_VERIFY
 - macros.h, [3389](#)
- _GLIBCXX_DEPRECATED_ATTR
 - binders, [127](#)
- _GLIBCXX_DEQUE_BUF_SIZE
 - stl_deque.h, [3530](#)
- _GLIBCXX_FIND_CONSTANT_-SIZE_BLOCKS
 - features.h, [3328](#)
- _GLIBCXX_FIND_EQUAL_SPLIT
 - features.h, [3328](#)
- _GLIBCXX_FIND_GROWING_-BLOCKS
 - features.h, [3328](#)
- _GLIBCXX_HAS_NESTED_TYPE
 - metaprogramming, [124](#)
- _GLIBCXX_MERGESORT
 - features.h, [3329](#)
- _GLIBCXX_PARALLEL_CONDITION
 - settings.h, [3498](#)

-
- `_GLIBCXX_PARALLEL_LENGTH`
 multiway_merge.h, 3410
 - `_GLIBCXX_QUICKSORT`
 features.h, 3329
 - `_GLIBCXX_RANDOM_SHUFFLE_-`
 CONSIDER_L1
 compiletime_settings.h, 3280
 - `_GLIBCXX_RANDOM_SHUFFLE_-`
 CONSIDER_TLB
 compiletime_settings.h, 3280
 - `_GLIBCXX_SCALE_DOWN_FPU`
 compiletime_settings.h, 3281
 - `_GLIBCXX_TREE_DYNAMIC_-`
 BALANCING
 features.h, 3329
 - `_GLIBCXX_TREE_FULL_COPY`
 features.h, 3329
 - `_GLIBCXX_TREE_INITIAL_-`
 SPLITTING
 features.h, 3330
 - `_GLIBCXX_VERBOSE_LEVEL`
 compiletime_settings.h, 3281
 - `_GLIBCXX_VOLATILE`
 partition.h, 3434
 queue.h, 3454
 - `_GuardedIterator`
 __gnu_parallel::_GuardedIterator,
 1158
 - `_LoserTreeBase`
 __gnu_parallel::_LoserTreeBase,
 1180
 - `_M_allocate_and_copy`
 std::vector, 3155
 - `_M_allocate_single_object`
 __gnu_cxx::bitmap_allocator, 876
 - `_M_attach`
 __gnu_debug::_Safe_iterator, 1030
 __gnu_debug::_Safe_iterator_base,
 1041
 - `_M_attach_single`
 __gnu_debug::_Safe_iterator, 1031
 __gnu_debug::_Safe_iterator_base,
 1041
 - `_M_attached_to`
 __gnu_debug::_Safe_iterator, 1031
 - __gnu_debug::_Safe_iterator_base,
 1041
 - `_M_before_dereferenceable`
 __gnu_debug::_Safe_iterator, 1031
 - `_M_begin`
 __gnu_parallel::_Piece, 1201
 - `_M_bin_proc`
 __gnu_parallel::_-
 DRandomShufflingGlobalData,
 1149
 - `_M_bins_begin`
 __gnu_parallel::_DRSSorterPU,
 1152
 - `_M_buf`
 __gnu_cxx::enc_filebuf, 906
 __gnu_cxx::stdio_filebuf, 979
 std::basic_filebuf, 1549
 - `_M_buf_locale`
 __gnu_cxx::enc_filebuf, 906
 __gnu_cxx::stdio_filebuf, 979
 __gnu_cxx::stdio_sync_filebuf,
 1006
 std::basic_filebuf, 1549
 std::basic_streambuf, 2041
 std::basic_stringbuf, 2126
 - `_M_buf_size`
 __gnu_cxx::enc_filebuf, 906
 __gnu_cxx::stdio_filebuf, 979
 std::basic_filebuf, 1549
 - `_M_can_compare`
 __gnu_debug::_Safe_iterator, 1031
 __gnu_debug::_Safe_iterator_base,
 1041
 - `_M_clear`
 __gnu_cxx::free_list, 916
 - `_M_comp`
 __gnu_parallel::_LoserTree, 1174
 __gnu_parallel::_LoserTree< false,
 _Tp, _Compare >, 1178
 __gnu_parallel::_LoserTreeBase,
 1181
 - `_M_const_iterators`
 __gnu_debug::_Safe_sequence,
 1046
 __gnu_debug::_Safe_sequence_-
 base, 1049
-

- __gnu_debug::basic_string, 1102
- std::__debug::bitset, 1339
- std::__debug::deque, 1345
- std::__debug::list, 1350
- std::__debug::map, 1356
- std::__debug::multimap, 1361
- std::__debug::multiset, 1366
- std::__debug::set, 1372
- std::__debug::unordered_map, 1377
- std::__debug::unordered_multimap, 1381
- std::__debug::unordered_multiset, 1386
- std::__debug::unordered_set, 1391
- std::__debug::vector, 1396
- _M_create_node
 - std::list, 2611
- _M_create_pback
 - __gnu_cxx::enc_filebuf, 889
 - __gnu_cxx::stdio_filebuf, 959
 - std::basic_filebuf, 1529
- _M_data
 - std::List_node, 1460
- _M_deallocate_single_object
 - __gnu_cxx::bitmap_allocator, 876
- _M_dereferenceable
 - __gnu_debug::_Safe_iterator, 1032
- _M_destroy_pback
 - __gnu_cxx::enc_filebuf, 889
 - __gnu_cxx::stdio_filebuf, 959
 - std::basic_filebuf, 1529
- _M_detach
 - __gnu_debug::_Safe_iterator, 1032
 - __gnu_debug::_Safe_iterator_base, 1041
- _M_detach_all
 - __gnu_debug::_Safe_sequence, 1044
 - __gnu_debug::_Safe_sequence_base, 1048
 - __gnu_debug::basic_string, 1058
 - std::__debug::bitset, 1338
 - std::__debug::deque, 1343
 - std::__debug::list, 1349
 - std::__debug::map, 1354
 - std::__debug::multimap, 1360
 - std::__debug::multiset, 1365
 - std::__debug::set, 1370
 - std::__debug::unordered_map, 1375
 - std::__debug::unordered_multimap, 1380
 - std::__debug::unordered_multiset, 1384
 - std::__debug::unordered_set, 1389
 - std::__debug::vector, 1395
- _M_detach_single
 - __gnu_debug::_Safe_iterator, 1032
 - __gnu_debug::_Safe_iterator_base, 1041
- _M_detach_singular
 - __gnu_debug::_Safe_sequence, 1044
 - __gnu_debug::_Safe_sequence_base, 1048
 - __gnu_debug::basic_string, 1058
 - std::__debug::bitset, 1338
 - std::__debug::deque, 1343
 - std::__debug::list, 1349
 - std::__debug::map, 1354
 - std::__debug::multimap, 1360
 - std::__debug::multiset, 1365
 - std::__debug::set, 1370
 - std::__debug::unordered_map, 1375
 - std::__debug::unordered_multimap, 1380
 - std::__debug::unordered_multiset, 1384
 - std::__debug::unordered_set, 1389
 - std::__debug::vector, 1395
- _M_dist
 - __gnu_parallel::_DRandomShufflingGlobalData, 1149
- _M_elements_leftover
 - __gnu_parallel::_QSBThreadLocal, 1210
- _M_end
 - __gnu_parallel::_Piece, 1201
- _M_ext_buf
 - __gnu_cxx::enc_filebuf, 906
 - __gnu_cxx::stdio_filebuf, 980
 - std::basic_filebuf, 1549

-
- `_M_ext_buf_size`
 - `__gnu_cxx::enc_filebuf`, 906
 - `__gnu_cxx::stdio_filebuf`, 980
 - `std::basic_filebuf`, 1550
 - `_M_ext_next`
 - `__gnu_cxx::enc_filebuf`, 907
 - `__gnu_cxx::stdio_filebuf`, 980
 - `std::basic_filebuf`, 1550
 - `_M_fill_initialize`
 - `std::deque`, 2394
 - `_M_finish_iterator`
 - `__gnu_parallel::__accumulate_-selector`, 1105
 - `__gnu_parallel::__adjacent_difference_selector`, 1106
 - `__gnu_parallel::__count_if_selector`, 1113
 - `__gnu_parallel::__count_selector`, 1115
 - `__gnu_parallel::__fill_selector`, 1117
 - `__gnu_parallel::__for_each_-selector`, 1123
 - `__gnu_parallel::__generate_-selector`, 1124
 - `__gnu_parallel::__generic_for_each_selector`, 1127
 - `__gnu_parallel::__identity_selector`, 1128
 - `__gnu_parallel::__inner_product_selector`, 1131
 - `__gnu_parallel::__replace_if_selector`, 1140
 - `__gnu_parallel::__replace_selector`, 1143
 - `__gnu_parallel::__transform1_selector`, 1144
 - `__gnu_parallel::__transform2_selector`, 1146
 - `_M_first`
 - `__gnu_parallel::__Job`, 1165
 - `_M_first_insert`
 - `__gnu_parallel::__LoserTree`, 1174
 - `__gnu_parallel::__LoserTree< _Tp, _Compare >`, 1178
 - `__gnu_parallel::__LoserTreeBase`, 1181
 - `_M_gcount`
 - `std::basic_fstream`, 1615
 - `std::basic_ifstream`, 1668
 - `std::basic_iostream`, 1764
 - `std::basic_istream`, 1815
 - `std::basic_istreamstream`, 1870
 - `std::basic_stringstream`, 2188
 - `_M_get`
 - `__gnu_cxx::free_list`, 916
 - `_M_get_distance`
 - `__gnu_debug::_Safe_iterator`, 1032
 - `_M_get_mutex`
 - `__gnu_debug::_Safe_iterator`, 1032
 - `__gnu_debug::_Safe_iterator_base`, 1041
 - `__gnu_debug::_Safe_sequence`, 1045
 - `__gnu_debug::_Safe_sequence_base`, 1049
 - `__gnu_debug::basic_string`, 1059
 - `std::__debug::bitset`, 1338
 - `std::__debug::deque`, 1343
 - `std::__debug::list`, 1349
 - `std::__debug::map`, 1355
 - `std::__debug::multimap`, 1360
 - `std::__debug::multiset`, 1365
 - `std::__debug::set`, 1371
 - `std::__debug::unordered_map`, 1375
 - `std::__debug::unordered_multimap`, 1380
 - `std::__debug::unordered_multiset`, 1385
 - `std::__debug::unordered_set`, 1389
 - `std::__debug::vector`, 1395
 - `_M_get_result`
 - `std::__basic_future`, 1315
 - `std::future`, 2488
 - `std::future< _Res & >`, 2492
 - `std::future< void >`, 2495
 - `std::shared_future`, 2998
 - `std::shared_future< _Res & >`, 3001
 - `std::shared_future< void >`, 3004
 - `_M_getloc`
 - `std::basic_fstream`, 1571
 - `std::basic_ifstream`, 1634
 - `std::basic_ios`, 1685
-

- std::basic_iostream, 1720
- std::basic_istream, 1782
- std::basic_istream, 1836
- std::basic_ofstream, 1888
- std::basic_ostream, 1931
- std::basic_ostringstream, 1976
- std::basic_stringstream, 2144
- std::ios_base, 2557
- _M_global
 - __gnu_parallel::_QSBThreadLocal, 1210
- _M_in_beg
 - __gnu_cxx::enc_filebuf, 907
 - __gnu_cxx::stdio_filebuf, 980
 - __gnu_cxx::stdio_sync_filebuf, 1006
 - std::basic_filebuf, 1550
 - std::basic_streambuf, 2042
 - std::basic_stringbuf, 2126
- _M_in_cur
 - __gnu_cxx::enc_filebuf, 907
 - __gnu_cxx::stdio_filebuf, 981
 - __gnu_cxx::stdio_sync_filebuf, 1006
 - std::basic_filebuf, 1550
 - std::basic_streambuf, 2042
 - std::basic_stringbuf, 2126
- _M_in_end
 - __gnu_cxx::enc_filebuf, 907
 - __gnu_cxx::stdio_filebuf, 981
 - __gnu_cxx::stdio_sync_filebuf, 1007
 - std::basic_filebuf, 1551
 - std::basic_streambuf, 2042
 - std::basic_stringbuf, 2126
- _M_incrementable
 - __gnu_debug::_Safe_iterator, 1033
- _M_initial
 - __gnu_parallel::_QSBThreadLocal, 1210
- _M_initialize_map
 - std::_Deque_base, 1442
 - std::deque, 2395
- _M_insert
 - __gnu_cxx::free_list, 916
- _M_invalidate
 - __gnu_debug::_Safe_iterator, 1033
- _M_invalidate_all
 - __gnu_debug::_Safe_sequence, 1045
 - __gnu_debug::_Safe_sequence_base, 1049
 - __gnu_debug::basic_string, 1059
 - std::_debug::bitset, 1338
 - std::_debug::deque, 1344
 - std::_debug::list, 1349
 - std::_debug::map, 1355
 - std::_debug::multimap, 1360
 - std::_debug::multiset, 1365
 - std::_debug::set, 1371
 - std::_debug::unordered_map, 1376
 - std::_debug::unordered_multimap, 1380
 - std::_debug::unordered_multiset, 1385
 - std::_debug::unordered_set, 1390
 - std::_debug::vector, 1395
- _M_invalidate_if
 - __gnu_debug::_Safe_sequence, 1045
 - __gnu_debug::basic_string, 1059
 - std::_debug::deque, 1344
 - std::_debug::list, 1350
 - std::_debug::map, 1355
 - std::_debug::multimap, 1360
 - std::_debug::multiset, 1365
 - std::_debug::set, 1371
 - std::_debug::unordered_map, 1376
 - std::_debug::unordered_multimap, 1380
 - std::_debug::unordered_multiset, 1385
 - std::_debug::unordered_set, 1390
 - std::_debug::vector, 1396
- _M_invalidate_single
 - __gnu_debug::_Safe_iterator, 1033
- _M_is_before_begin
 - __gnu_debug::_Safe_iterator, 1034
- _M_is_begin
 - __gnu_debug::_Safe_iterator, 1034
- _M_is_end
 - __gnu_debug::_Safe_iterator, 1034

- `_M_iterators`
 - `__gnu_debug::_Safe_sequence`, 1046
 - `__gnu_debug::_Safe_sequence_base`, 1050
 - `__gnu_debug::basic_string`, 1102
 - `std::__debug::bitset`, 1339
 - `std::__debug::deque`, 1345
 - `std::__debug::list`, 1351
 - `std::__debug::map`, 1356
 - `std::__debug::multimap`, 1361
 - `std::__debug::multiset`, 1366
 - `std::__debug::set`, 1372
 - `std::__debug::unordered_map`, 1377
 - `std::__debug::unordered_multimap`, 1381
 - `std::__debug::unordered_multiset`, 1386
 - `std::__debug::unordered_set`, 1391
 - `std::__debug::vector`, 1397
- `_M_key`
 - `__gnu_parallel::LoserTreeBase::_Loser`, 1183
- `_M_last`
 - `__gnu_parallel::Job`, 1165
- `_M_leftover_parts`
 - `__gnu_parallel::QSBThreadLocal`, 1210
- `_M_load`
 - `__gnu_parallel::Job`, 1165
- `_M_log_k`
 - `__gnu_parallel::LoserTree`, 1174
 - `__gnu_parallel::LoserTree< false, _Tp, _Compare >`, 1178
 - `__gnu_parallel::LoserTreeBase`, 1182
- `_M_losers`
 - `__gnu_parallel::LoserTree`, 1174
 - `__gnu_parallel::LoserTree< false, _Tp, _Compare >`, 1178
 - `__gnu_parallel::LoserTreeBase`, 1182
- `_M_mode`
 - `__gnu_cxx::enc_filebuf`, 908
 - `__gnu_cxx::stdio_filebuf`, 981
 - `std::basic_filebuf`, 1551
 - `std::basic_stringbuf`, 2127
- `_M_new_elements_at_back`
 - `std::deque`, 2395
- `_M_new_elements_at_front`
 - `std::deque`, 2395
- `_M_next`
 - `__gnu_debug::_Safe_iterator`, 1037
 - `__gnu_debug::_Safe_iterator_base`, 1042
- `_M_num_bins`
 - `__gnu_parallel::_DRandomShufflingGlobalData`, 1150
- `_M_num_bits`
 - `__gnu_parallel::_DRandomShufflingGlobalData`, 1150
- `_M_num_threads`
 - `__gnu_parallel::_DRSSorterPU`, 1152
 - `__gnu_parallel::_PMWMSortingData`, 1204
 - `__gnu_parallel::_QSBThreadLocal`, 1210
- `_M_offsets`
 - `__gnu_parallel::_PMWMSortingData`, 1204
- `_M_out_beg`
 - `__gnu_cxx::enc_filebuf`, 908
 - `__gnu_cxx::stdio_filebuf`, 982
 - `__gnu_cxx::stdio_sync_filebuf`, 1007
 - `std::basic_filebuf`, 1551
 - `std::basic_streambuf`, 2043
 - `std::basic_stringbuf`, 2127
- `_M_out_cur`
 - `__gnu_cxx::enc_filebuf`, 908
 - `__gnu_cxx::stdio_filebuf`, 982
 - `__gnu_cxx::stdio_sync_filebuf`, 1008
 - `std::basic_filebuf`, 1552
 - `std::basic_streambuf`, 2043
 - `std::basic_stringbuf`, 2127
- `_M_out_end`
 - `__gnu_cxx::enc_filebuf`, 909
 - `__gnu_cxx::stdio_filebuf`, 982

- __gnu_cxx::stdio_sync_filebuf, 1008
- std::basic_filebuf, 1552
- std::basic_streambuf, 2043
- std::basic_stringbuf, 2128
- _M_pback
 - __gnu_cxx::enc_filebuf, 909
 - __gnu_cxx::stdio_filebuf, 983
 - std::basic_filebuf, 1553
- _M_pback_cur_save
 - __gnu_cxx::enc_filebuf, 909
 - __gnu_cxx::stdio_filebuf, 983
 - std::basic_filebuf, 1553
- _M_pback_end_save
 - __gnu_cxx::enc_filebuf, 909
 - __gnu_cxx::stdio_filebuf, 983
 - std::basic_filebuf, 1553
- _M_pback_init
 - __gnu_cxx::enc_filebuf, 910
 - __gnu_cxx::stdio_filebuf, 983
 - std::basic_filebuf, 1553
- _M_pieces
 - __gnu_parallel::_-PMWMSSortingData, 1204
- _M_pop_back_aux
 - std::deque, 2396
- _M_pop_front_aux
 - std::deque, 2396
- _M_prior
 - __gnu_debug::_Safe_iterator, 1037
 - __gnu_debug::_Safe_iterator_base, 1042
- _M_push_back_aux
 - std::deque, 2396
- _M_push_front_aux
 - std::deque, 2396
- _M_range_check
 - std::deque, 2397
 - std::vector, 3155
- _M_range_initialize
 - std::deque, 2397
- _M_reading
 - __gnu_cxx::enc_filebuf, 910
 - __gnu_cxx::stdio_filebuf, 984
 - std::basic_filebuf, 1554
- _M_reallocate_map
 - std::deque, 2398
- _M_reserve_elements_at_back
 - std::deque, 2398
- _M_reserve_elements_at_front
 - std::deque, 2398
- _M_reserve_map_at_back
 - std::deque, 2399
- _M_reserve_map_at_front
 - std::deque, 2399
- _M_revalidate_singular
 - __gnu_debug::_Safe_sequence, 1045
 - __gnu_debug::_Safe_sequence_base, 1049
 - __gnu_debug::basic_string, 1059
 - std::__debug::bitset, 1339
 - std::__debug::deque, 1344
 - std::__debug::list, 1350
 - std::__debug::map, 1355
 - std::__debug::multimap, 1360
 - std::__debug::multiset, 1366
 - std::__debug::set, 1371
 - std::__debug::unordered_map, 1376
 - std::__debug::unordered_multimap, 1381
 - std::__debug::unordered_multiset, 1385
 - std::__debug::unordered_set, 1390
 - std::__debug::vector, 1396
- _M_samples
 - __gnu_parallel::_-PMWMSSortingData, 1205
- _M_sd
 - __gnu_parallel::_DRSSorterPU, 1152
- _M_seed
 - __gnu_parallel::_DRSSorterPU, 1153
- _M_sequence
 - __gnu_debug::_Safe_iterator, 1038
 - __gnu_debug::_Safe_iterator_base, 1042
- _M_sequential_algorithm
 - __gnu_parallel::_adjacent_find_selector, 1108

-
- __gnu_parallel::__find_first_of_-
selector, [1119](#)
 - __gnu_parallel::__find_if_selector,
[1120](#)
 - __gnu_parallel::__mismatch_-
selector, [1134](#)
 - _M_set_buffer
 - __gnu_cxx::enc_filebuf, [889](#)
 - __gnu_cxx::stdio_filebuf, [959](#)
 - std::basic_filebuf, [1530](#)
 - _M_set_node
 - std::_Deque_iterator, [1444](#)
 - _M_singular
 - __gnu_debug::_Safe_iterator, [1034](#)
 - __gnu_debug::_Safe_iterator_base,
[1042](#)
 - _M_source
 - __gnu_parallel::_-
DRandomShufflingGlobalData,
[1150](#)
 - __gnu_parallel::_LoserTreeBase::_-
Loser, [1183](#)
 - __gnu_parallel::_-
PMWMSSortingData, [1205](#)
 - _M_starts
 - __gnu_parallel::_-
DRandomShufflingGlobalData,
[1150](#)
 - __gnu_parallel::_-
PMWMSSortingData, [1205](#)
 - _M_sup
 - __gnu_parallel::_LoserTreeBase::_-
Loser, [1183](#)
 - _M_swap
 - __gnu_debug::_Safe_sequence,
[1045](#)
 - __gnu_debug::_Safe_sequence_-
base, [1049](#)
 - __gnu_debug::basic_string, [1059](#)
 - std::__debug::bitset, [1339](#)
 - std::__debug::deque, [1344](#)
 - std::__debug::list, [1350](#)
 - std::__debug::map, [1355](#)
 - std::__debug::multimap, [1361](#)
 - std::__debug::multiset, [1366](#)
 - std::__debug::set, [1371](#)
 - std::__debug::unordered_map, [1376](#)
 - std::__debug::unordered_multimap,
[1381](#)
 - std::__debug::unordered_multiset,
[1385](#)
 - std::__debug::unordered_set, [1390](#)
 - std::__debug::vector, [1396](#)
 - _M_temporaries
 - __gnu_parallel::_-
DRandomShufflingGlobalData,
[1151](#)
 - _M_temporary
 - __gnu_parallel::_-
PMWMSSortingData, [1205](#)
 - _M_transfer_iter
 - __gnu_debug::_Safe_sequence,
[1046](#)
 - __gnu_debug::basic_string, [1060](#)
 - std::__debug::deque, [1344](#)
 - std::__debug::list, [1350](#)
 - std::__debug::map, [1356](#)
 - std::__debug::multimap, [1361](#)
 - std::__debug::multiset, [1366](#)
 - std::__debug::set, [1372](#)
 - std::__debug::unordered_map, [1376](#)
 - std::__debug::unordered_multimap,
[1381](#)
 - std::__debug::unordered_multiset,
[1385](#)
 - std::__debug::unordered_set, [1390](#)
 - std::__debug::vector, [1396](#)
 - _M_use_pointer
 - __gnu_parallel::_LoserTreeTraits,
[1193](#)
 - _M_version
 - __gnu_debug::_Safe_iterator, [1038](#)
 - __gnu_debug::_Safe_iterator_base,
[1043](#)
 - __gnu_debug::_Safe_sequence,
[1046](#)
 - __gnu_debug::_Safe_sequence_-
base, [1050](#)
 - __gnu_debug::basic_string, [1102](#)
 - std::__debug::bitset, [1339](#)
 - std::__debug::deque, [1345](#)
 - std::__debug::list, [1351](#)
-

- std::__debug::map, 1356
- std::__debug::multimap, 1361
- std::__debug::multiset, 1367
- std::__debug::set, 1372
- std::__debug::unordered_map, 1377
- std::__debug::unordered_multimap, 1382
- std::__debug::unordered_multiset, 1386
- std::__debug::unordered_set, 1391
- std::__debug::vector, 1397
- _M_w
 - std::_Base_bitset, 1437
- _M_write
 - std::basic_fstream, 1571
 - std::basic_istream, 1720
 - std::basic_ofstream, 1889
 - std::basic_ostream, 1932
 - std::basic_ostringstream, 1977
 - std::basic_stringstream, 2145
- _MultiwayMergeAlgorithm
 - __gnu_parallel, 407
- _Parallelism
 - __gnu_parallel, 407
- _PartialSumAlgorithm
 - __gnu_parallel, 408
- _Piece
 - __gnu_parallel::__QSBThreadLocal, 1209
- _PseudoSequence
 - __gnu_parallel::_PseudoSequence, 1206
- _QSBThreadLocal
 - __gnu_parallel::__QSBThreadLocal, 1209
- _RandomNumber
 - __gnu_parallel::_RandomNumber, 1211
- _RestrictedBoundedConcurrentQueue
 - __gnu_parallel::_RestrictedBoundedConcurrentQueue, 1214
- _Safe_iterator
 - __gnu_debug::_Safe_iterator, 1029, 1030
- _Safe_iterator_base
 - __gnu_debug::_Safe_iterator_base, 1040
- _SequenceIndex
 - __gnu_parallel, 407
- _SortAlgorithm
 - __gnu_parallel, 408
- _SplittingAlgorithm
 - __gnu_parallel, 408
- _Temporary_buffer
 - std::_Temporary_buffer, 1479
- _ThreadIndex
 - __gnu_parallel, 407
- _Unchecked_flip
 - SGIextensions, 18
- _Unchecked_reset
 - SGIextensions, 18
- _Unchecked_set
 - SGIextensions, 18
- _Unchecked_test
 - SGIextensions, 18
- __atomic0::atomic_address, 773
- __atomic0::atomic_bool, 774
- __atomic0::atomic_flag, 775
- __atomic2::atomic_address, 775
- __atomic2::atomic_bool, 776
- __atomic2::atomic_flag, 777
- __base
 - __gnu_debug, 386
- __begin1_iterator
 - __gnu_parallel::__inner_product_selector, 1130
- __begin2_iterator
 - __gnu_parallel::__inner_product_selector, 1131
- __bins_end
 - __gnu_parallel::_DRSSorterPU, 1152
- __bit_allocate
 - __gnu_cxx::__detail, 378
- __bit_free
 - __gnu_cxx::__detail, 378
- __calc_borders
 - __gnu_parallel, 408
- __check_dereferenceable
 - __gnu_debug, 386, 387
- __check_singular

-
- __gnu_debug, 387
 - __check_singular_aux
 - __gnu_debug, 387
 - __check_string
 - __gnu_debug, 388
 - __compare_and_swap
 - __gnu_parallel, 408
 - __compare_and_swap_32
 - __gnu_parallel, 409
 - __compare_and_swap_64
 - __gnu_parallel, 409
 - __ctype_type
 - std::basic_fstream, 1563
 - std::basic_ifstream, 1628
 - std::basic_ios, 1679
 - std::basic_istream, 1713
 - std::basic_istream, 1776
 - std::basic_istream, 1830
 - std::basic_ostream, 1883
 - std::basic_ostream, 1926
 - std::basic_ostringstream, 1971
 - std::basic_stringstream, 2137
 - __cxxabiv1::__forced_unwind, 777
 - __decode2
 - __gnu_parallel, 410
 - __delete_min_insert
 - __gnu_parallel::_LoserTree, 1173
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1176
 - __determine_samples
 - __gnu_parallel, 410
 - __encode2
 - __gnu_parallel, 411
 - __env_t
 - __gnu_profile, 463
 - __fetch_and_add
 - __gnu_parallel, 411
 - __fetch_and_add_32
 - __gnu_parallel, 412
 - __fetch_and_add_64
 - __gnu_parallel, 412
 - __final_insertion_sort
 - std, 606
 - __find
 - std, 606, 607
 - __find_if
 - std, 607
 - __find_if_not
 - std, 607, 608
 - __find_template
 - __gnu_parallel, 412–414
 - __for_each_template_random_access
 - __gnu_parallel, 415
 - __for_each_template_random_access_ed
 - __gnu_parallel, 416
 - __for_each_template_random_access_-omp_loop
 - __gnu_parallel, 416
 - __for_each_template_random_access_-omp_loop_static
 - __gnu_parallel, 417
 - __for_each_template_random_access_-workstealing
 - __gnu_parallel, 418
 - __gcd
 - std, 608
 - __genrand_bits
 - __gnu_parallel::_RandomNumber, 1212
 - __get__global_lock
 - __gnu_profile, 463
 - __get_min_source
 - __gnu_parallel::_LoserTree, 1173
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1176
 - __gnu_parallel::_LoserTreeBase, 1181
 - __get_num_threads
 - __gnu_parallel::balanced_-quicksort_tag, 1228
 - __gnu_parallel::balanced_tag, 1230
 - __gnu_parallel::default_parallel_-tag, 1232
 - __gnu_parallel::exact_tag, 1235
 - __gnu_parallel::multiway_-mergesort_exact_tag, 1238
 - __gnu_parallel::multiway_-mergesort_sampling_tag, 1240
 - __gnu_parallel::multiway_-mergesort_tag, 1241
-

- __gnu_parallel::omp_loop_static_tag, 1243
- __gnu_parallel::omp_loop_tag, 1244
- __gnu_parallel::parallel_tag, 1247
- __gnu_parallel::quicksort_tag, 1249
- __gnu_parallel::sampling_tag, 1250
- __gnu_parallel::unbalanced_tag, 1252
- __libcxx_check_erase
 - macros.h, 3387
- __libcxx_check_erase_after
 - macros.h, 3387
- __libcxx_check_erase_range
 - macros.h, 3387
- __libcxx_check_erase_range_after
 - macros.h, 3387
- __libcxx_check_heap_pred
 - macros.h, 3387
- __libcxx_check_insert
 - macros.h, 3387
- __libcxx_check_insert_after
 - macros.h, 3388
- __libcxx_check_insert_range
 - macros.h, 3388
- __libcxx_check_insert_range_after
 - macros.h, 3388
- __libcxx_check_partitioned_lower
 - macros.h, 3388
- __libcxx_check_partitioned_lower_pred
 - macros.h, 3389
- __libcxx_check_partitioned_upper_pred
 - macros.h, 3389
- __libcxx_check_sorted_pred
 - macros.h, 3389
- __gnu_cxx, 343
 - _Bit_scan_forward, 366
 - __static_pointer_cast, 365
 - operator<, 369, 370
 - operator<=, 371, 372
 - operator>, 374
 - operator>=, 375, 376
 - operator+, 367–369
 - operator==, 372, 373
 - swap, 376
- __gnu_cxx::_Caster, 860
- __gnu_cxx::_Char_types, 860
- __gnu_cxx::_ExtPtr_allocator, 861
- __gnu_cxx::_Invalid_type, 863
- __gnu_cxx::_Pointer_adapter, 863
- __gnu_cxx::_Relative_pointer_impl, 866
- __gnu_cxx::_Relative_pointer_impl<const_Tp >, 867
- __gnu_cxx::_Std_pointer_impl, 867
- __gnu_cxx::_Unqualified_type, 868
- __gnu_cxx::__common_pool_policy, 778
- __gnu_cxx::__detail, 377
 - _bit_allocate, 378
 - _bit_free, 378
 - _num_bitmaps, 378
 - _num_blocks, 379
- __gnu_cxx::__detail::_Bitmap_counter, 779
- __gnu_cxx::__detail::_Ffit_finder, 781
- __gnu_cxx::__detail::argument_type, 782
- __gnu_cxx::__detail::result_type, 782
- __gnu_cxx::__detail::_mini_vector, 778
- __gnu_cxx::_mt_alloc, 782
- __gnu_cxx::_mt_alloc_base, 784
- __gnu_cxx::_per_type_pool_policy, 785
- __gnu_cxx::_pool<false >, 786
- __gnu_cxx::_pool<true >, 787
- __gnu_cxx::_pool_alloc, 789
- __gnu_cxx::_pool_alloc_base, 791
- __gnu_cxx::_pool_base, 792
- __gnu_cxx::_rc_string_base, 794
- __gnu_cxx::_scoped_lock, 797
- __gnu_cxx::_versa_string, 797
 - ~__versa_string, 806
- __versa_string, 802–805
 - append, 806–808
 - assign, 809–812
 - at, 813
 - back, 814
 - begin, 814
 - c_str, 814
 - capacity, 815
 - cbegin, 815
 - cend, 815
 - clear, 815
 - compare, 816–819
 - copy, 820

- crbegin, 820
- crend, 821
- data, 821
- empty, 821
- end, 822
- erase, 822, 823
- find, 824, 825
- find_first_not_of, 826–828
- find_first_of, 828–830
- find_last_not_of, 830–832
- find_last_of, 832–834
- front, 835
- get_allocator, 835
- insert, 835–840
- length, 840
- max_size, 841
- npos, 859
- operator+=, 841, 842
- operator=, 842–844
- push_back, 845
- rbegin, 845
- rend, 846
- replace, 846–853
- reserve, 854
- resize, 854, 855
- rfind, 855–857
- shrink_to_fit, 858
- size, 858
- substr, 858
- swap, 859
- __gnu_cxx::annotate_base, 869
- __gnu_cxx::array_allocator, 870
- __gnu_cxx::array_allocator_base, 871
- __gnu_cxx::binary_compose, 872
 - argument_type, 874
 - result_type, 874
- __gnu_cxx::bitmap_allocator, 874
 - _M_allocate_single_object, 876
 - _M_deallocate_single_object, 876
- __gnu_cxx::char_traits, 877
- __gnu_cxx::character, 878
- __gnu_cxx::condition_base, 879
- __gnu_cxx::constant_binary_fun, 880
- __gnu_cxx::constant_unary_fun, 881
- __gnu_cxx::constant_void_fun, 882
- __gnu_cxx::debug_allocator, 883
- __gnu_cxx::enc_filebuf, 884
 - _M_buf, 906
 - _M_buf_locale, 906
 - _M_buf_size, 906
 - _M_create_pback, 889
 - _M_destroy_pback, 889
 - _M_ext_buf, 906
 - _M_ext_buf_size, 906
 - _M_ext_next, 907
 - _M_in_beg, 907
 - _M_in_cur, 907
 - _M_in_end, 907
 - _M_mode, 908
 - _M_out_beg, 908
 - _M_out_cur, 908
 - _M_out_end, 909
 - _M_pback, 909
 - _M_pback_cur_save, 909
 - _M_pback_end_save, 909
 - _M_pback_init, 910
 - _M_reading, 910
 - _M_set_buffer, 889
 - __streambuf_type, 887
- char_type, 888
- close, 890
- eback, 890
- egptr, 890
- epptr, 891
- gbump, 891
- getloc, 891
- gptr, 892
- imbue, 892
- in_avail, 892
- int_type, 888
- is_open, 893
- off_type, 888
- open, 893
- overflow, 894
- pbackfail, 894
- pbase, 895
- pbump, 895
- pos_type, 888
- pptr, 896
- pubimbue, 896
- pubseekoff, 896
- pubseekpos, 897

- pubsetbuf, 897
- pubsync, 897
- sbumpc, 898
- seekoff, 898
- seekpos, 898
- setbuf, 899
- setg, 899
- setp, 900
- sgetc, 900
- sgetn, 900
- showmanyc, 901
- snextc, 901
- sputbackc, 902
- sputc, 902
- sputn, 902
- stossc, 903
- sungetc, 903
- sync, 903
- traits_type, 889
- uflow, 904
- underflow, 904
- xsgetn, 905
- xspn, 905
- __gnu_cxx::encoding_char_traits, 910
- __gnu_cxx::encoding_state, 912
- __gnu_cxx::forced_error, 913
 - what, 914
- __gnu_cxx::free_list, 915
 - _M_clear, 916
 - _M_get, 916
 - _M_insert, 916
- __gnu_cxx::hash_map, 916
- __gnu_cxx::hash_multimap, 919
- __gnu_cxx::hash_multiset, 921
- __gnu_cxx::hash_set, 923
- __gnu_cxx::limit_condition, 925
- __gnu_cxx::limit_condition::always_ -
 - adjustor, 926
- __gnu_cxx::limit_condition::limit_ -
 - adjustor, 926
- __gnu_cxx::limit_condition::never_ -
 - adjustor, 927
- __gnu_cxx::malloc_allocator, 927
- __gnu_cxx::new_allocator, 928
- __gnu_cxx::project1st, 930
 - first_argument_type, 931
 - result_type, 931
 - second_argument_type, 931
- __gnu_cxx::project2nd, 931
 - first_argument_type, 932
 - result_type, 932
 - second_argument_type, 932
- __gnu_cxx::random_condition, 933
- __gnu_cxx::random_condition::always_ -
 - adjustor, 934
- __gnu_cxx::random_condition::group_ -
 - adjustor, 934
- __gnu_cxx::random_condition::never_ -
 - adjustor, 935
- __gnu_cxx::rb_tree, 935
- __gnu_cxx::recursive_init_error, 938
 - what, 939
- __gnu_cxx::rope, 940
- __gnu_cxx::select1st, 946
 - argument_type, 947
 - result_type, 947
- __gnu_cxx::select2nd, 947
 - argument_type, 948
 - result_type, 948
- __gnu_cxx::slist, 948
- __gnu_cxx::stdio_filebuf, 951
 - ~stdio_filebuf, 958
 - _M_buf, 979
 - _M_buf_locale, 979
 - _M_buf_size, 979
 - _M_create_pback, 959
 - _M_destroy_pback, 959
 - _M_ext_buf, 980
 - _M_ext_buf_size, 980
 - _M_ext_next, 980
 - _M_in_beg, 980
 - _M_in_cur, 981
 - _M_in_end, 981
 - _M_mode, 981
 - _M_out_beg, 982
 - _M_out_cur, 982
 - _M_out_end, 982
 - _M_pback, 983
 - _M_pback_cur_save, 983
 - _M_pback_end_save, 983
 - _M_pback_init, 983
 - _M_reading, 984

`_M_set_buffer`, 959
`__streambuf_type`, 956
`char_type`, 956
`close`, 959
`eback`, 960
`egptr`, 960
`epptr`, 961
`fd`, 961
`file`, 961
`gbump`, 962
`getloc`, 962
`gptr`, 962
`imbue`, 963
`in_avail`, 963
`int_type`, 956
`is_open`, 964
`off_type`, 957
`open`, 964, 965
`overflow`, 965
`pbackfail`, 966
`pbase`, 966
`pbump`, 967
`pos_type`, 957
`pptr`, 967
`pubimbue`, 968
`pubseekoff`, 968
`pubseekpos`, 968
`pubsetbuf`, 969
`pubsync`, 969
`sbumpc`, 969
`seekoff`, 970
`seekpos`, 970
`setbuf`, 971
`setg`, 971
`setp`, 972
`sgetc`, 972
`sgetn`, 973
`showmanyc`, 973
`snextc`, 974
`sputbackc`, 974
`sputc`, 974
`sputn`, 975
`stdio_filebuf`, 957, 958
`stossc`, 975
`sungetc`, 976
`sync`, 976
`traits_type`, 957
`uflow`, 976
`underflow`, 977
`xsgetn`, 978
`xspn`, 978
`__gnu_cxx::stdio_sync_filebuf`, 984
`_M_buf_locale`, 1006
`_M_in_beg`, 1006
`_M_in_cur`, 1006
`_M_in_end`, 1007
`_M_out_beg`, 1007
`_M_out_cur`, 1008
`_M_out_end`, 1008
`__streambuf_type`, 988
`char_type`, 988
`eback`, 990
`egptr`, 990
`epptr`, 991
`file`, 991
`gbump`, 991
`getloc`, 992
`gptr`, 992
`imbue`, 992
`in_avail`, 993
`int_type`, 989
`off_type`, 989
`overflow`, 993
`pbackfail`, 994
`pbase`, 994
`pbump`, 995
`pos_type`, 989
`pptr`, 995
`pubimbue`, 996
`pubseekoff`, 996
`pubseekpos`, 996
`pubsetbuf`, 997
`pubsync`, 997
`sbumpc`, 997
`seekoff`, 998
`seekpos`, 998
`setbuf`, 999
`setg`, 999
`setp`, 999
`sgetc`, 1000
`sgetn`, 1000
`showmanyc`, 1000

- snextc, 1001
- sputbackc, 1001
- sputc, 1002
- sputn, 1002
- stossc, 1003
- sungetc, 1003
- sync, 1003
- traits_type, 989
- uflow, 1004
- underflow, 1004
- xsgetn, 1005
- xspn, 1005
- __gnu_cxx::subtractive_rng, 1009
 - argument_type, 1010
 - operator(), 1010
 - result_type, 1010
 - subtractive_rng, 1010
- __gnu_cxx::temporary_buffer, 1011
 - ~temporary_buffer, 1012
 - begin, 1013
 - end, 1013
 - requested_size, 1013
 - size, 1013
 - temporary_buffer, 1012
- __gnu_cxx::throw_allocator_base, 1014
- __gnu_cxx::throw_allocator_limit, 1015
- __gnu_cxx::throw_allocator_random, 1017
- __gnu_cxx::throw_value_base, 1018
- __gnu_cxx::throw_value_limit, 1019
- __gnu_cxx::throw_value_random, 1021
- __gnu_cxx::typelist, 379
 - apply_generator, 379
- __gnu_cxx::unary_compose, 1022
 - argument_type, 1024
 - result_type, 1024
- __gnu_debug, 380
 - _base, 386
 - __check_dereferenceable, 386, 387
 - __check_singular, 387
 - __check_singular_aux, 387
 - __check_string, 388
 - __valid_range, 388
 - __valid_range_aux, 388, 389
 - __valid_range_aux2, 389
- __gnu_debug::_After_nth_from, 1025
- __gnu_debug::_BeforeBeginHelper, 1025
- __gnu_debug::_Not_equal_to, 1026
- __gnu_debug::_Safe_iterator, 1026
 - _M_attach, 1030
 - _M_attach_single, 1031
 - _M_attached_to, 1031
 - _M_before_dereferenceable, 1031
 - _M_can_compare, 1031
 - _M_dereferenceable, 1032
 - _M_detach, 1032
 - _M_detach_single, 1032
 - _M_get_distance, 1032
 - _M_get_mutex, 1032
 - _M_incrementable, 1033
 - _M_invalidate, 1033
 - _M_invalidate_single, 1033
 - _M_is_before_begin, 1034
 - _M_is_begin, 1034
 - _M_is_end, 1034
 - _M_next, 1037
 - _M_prior, 1037
 - _M_sequence, 1038
 - _M_singular, 1034
 - _M_version, 1038
- _Safe_iterator, 1029, 1030
- base, 1034
- operator_iterator, 1035
- operator*, 1035
- operator++, 1035, 1036
- operator->, 1036
- operator--, 1036
- operator=, 1037
- __gnu_debug::_Safe_iterator_base, 1038
 - _M_attach, 1041
 - _M_attach_single, 1041
 - _M_attached_to, 1041
 - _M_can_compare, 1041
 - _M_detach, 1041
 - _M_detach_single, 1041
 - _M_get_mutex, 1041
 - _M_next, 1042
 - _M_prior, 1042
 - _M_sequence, 1042
 - _M_singular, 1042
 - _M_version, 1043
- _Safe_iterator_base, 1040

-
- __gnu_debug::_Safe_sequence, 1043
 - _M_const_iterators, 1046
 - _M_detach_all, 1044
 - _M_detach_singular, 1044
 - _M_get_mutex, 1045
 - _M_invalidate_all, 1045
 - _M_invalidate_if, 1045
 - _M_iterators, 1046
 - _M_revalidate_singular, 1045
 - _M_swap, 1045
 - _M_transfer_iter, 1046
 - _M_version, 1046
 - __gnu_debug::_Safe_sequence_base, 1047
 - ~_Safe_sequence_base, 1048
 - _M_const_iterators, 1049
 - _M_detach_all, 1048
 - _M_detach_singular, 1048
 - _M_get_mutex, 1049
 - _M_invalidate_all, 1049
 - _M_iterators, 1050
 - _M_revalidate_singular, 1049
 - _M_swap, 1049
 - _M_version, 1050
 - __gnu_debug::__is_same, 1024
 - __gnu_debug::basic_string, 1050
 - _M_const_iterators, 1102
 - _M_detach_all, 1058
 - _M_detach_singular, 1058
 - _M_get_mutex, 1059
 - _M_invalidate_all, 1059
 - _M_invalidate_if, 1059
 - _M_iterators, 1102
 - _M_revalidate_singular, 1059
 - _M_swap, 1059
 - _M_transfer_iter, 1060
 - _M_version, 1102
 - append, 1060–1062
 - assign, 1062–1065
 - at, 1066
 - back, 1066, 1067
 - begin, 1067
 - c_str, 1067
 - capacity, 1067
 - cbegin, 1067
 - cend, 1068
 - clear, 1068
 - compare, 1068–1071
 - copy, 1071
 - crbegin, 1072
 - crend, 1072
 - data, 1072
 - empty, 1072
 - end, 1072, 1073
 - erase, 1073, 1074
 - find, 1074, 1075
 - find_first_not_of, 1076, 1077
 - find_first_of, 1077–1079
 - find_last_not_of, 1079, 1080
 - find_last_of, 1081, 1082
 - front, 1082, 1083
 - get_allocator, 1083
 - insert, 1083–1087
 - length, 1088
 - max_size, 1088
 - npos, 1102
 - operator+=, 1088, 1089
 - push_back, 1090
 - rbegin, 1090
 - rend, 1091
 - replace, 1091–1097
 - reserve, 1097
 - resize, 1098
 - rfind, 1099, 1100
 - shrink_to_fit, 1100
 - size, 1100
 - substr, 1101
 - swap, 1101
 - __gnu_internal, 390
 - __gnu_parallel, 390
 - _AlgorithmStrategy, 407
 - _BinIndex, 406
 - _CASable, 406
 - _CASable_bits, 454
 - _CASable_mask, 454
 - _FindAlgorithm, 407
 - _MultiwayMergeAlgorithm, 407
 - _Parallelism, 407
 - _PartialSumAlgorithm, 408
 - _SequenceIndex, 407
 - _SortAlgorithm, 408
 - _SplittingAlgorithm, 408
-

- `_ThreadIndex`, 407
- `__calc_borders`, 408
- `__compare_and_swap`, 408
- `__compare_and_swap_32`, 409
- `__compare_and_swap_64`, 409
- `__decode2`, 410
- `__determine_samples`, 410
- `__encode2`, 411
- `__fetch_and_add`, 411
- `__fetch_and_add_32`, 412
- `__fetch_and_add_64`, 412
- `__find_template`, 412–414
- `__for_each_template_random_-
access`, 415
- `__for_each_template_random_-
access_ed`, 416
- `__for_each_template_random_-
access_omp_loop`, 416
- `__for_each_template_random_-
access_omp_loop_static`, 417
- `__for_each_template_random_-
access_workstealing`, 418
- `__is_sorted`, 418
- `__median_of_three_iterators`, 419
- `__merge_advance`, 419
- `__merge_advance_movc`, 420
- `__merge_advance_usual`, 421
- `__parallel_merge_advance`, 421, 422
- `__parallel_nth_element`, 423
- `__parallel_partial_sort`, 423
- `__parallel_partial_sum`, 423
- `__parallel_partial_sum_basecase`,
424
- `__parallel_partial_sum_linear`, 424
- `__parallel_partition`, 425
- `__parallel_random_shuffle`, 426
- `__parallel_random_shuffle_drs`, 426
- `__parallel_random_shuffle_drs_pu`,
427
- `__parallel_sort`, 427–431
- `__parallel_sort_qs`, 432
- `__parallel_sort_qs_conquer`, 432
- `__parallel_sort_qs_divide`, 433
- `__parallel_sort_qsb`, 433
- `__parallel_unique_copy`, 434
- `__qsb_conquer`, 435
- `__qsb_divide`, 435
- `__qsb_local_sort_with_helping`, 436
- `__random_number_pow2`, 436
- `__rd_log2`, 437
- `__round_up_to_pow2`, 437
- `__search_template`, 438
- `__sequential_multiway_merge`, 438
- `__sequential_random_shuffle`, 439
- `__shrink`, 439
- `__shrink_and_double`, 440
- `__yield`, 440
- `equally_split`, 440
- `equally_split_point`, 441
- `list_partition`, 441
- `max`, 442
- `min`, 442
- `multiseq_partition`, 442
- `multiseq_selection`, 443
- `multiway_merge`, 444
- `multiway_merge_3_variant`, 446
- `multiway_merge_4_variant`, 446
- `multiway_merge_exact_splitting`,
447
- `multiway_merge_loser_tree`, 448
- `multiway_merge_loser_tree_-
sentinel`, 448
- `multiway_merge_loser_tree_-
unguarded`, 449
- `multiway_merge_sampling_-
splitting`, 450
- `multiway_merge_sentinels`, 450
- `parallel_multiway_merge`, 452
- `parallel_sort_mwms`, 453
- `parallel_sort_mwms_pu`, 453
- `__gnu_parallel::__DRSSorterPU`, 1151
- `__M_bins_begin`, 1152
- `__M_num_threads`, 1152
- `__M_sd`, 1152
- `__M_seed`, 1153
- `__bins_end`, 1152
- `__gnu_parallel::__
DRandomShufflingGlobalData`,
1148
- `__DRandomShufflingGlobalData`,
1149
- `__M_bin_proc`, 1149

- [_M_dist, 1149](#)
- [_M_num_bins, 1150](#)
- [_M_num_bits, 1150](#)
- [_M_source, 1150](#)
- [_M_starts, 1150](#)
- [_M_temporaries, 1151](#)
- [__gnu_parallel::_DummyReduct, 1153](#)
- [__gnu_parallel::_EqualFromLess, 1154](#)
 - [first_argument_type, 1155](#)
 - [result_type, 1155](#)
 - [second_argument_type, 1155](#)
- [__gnu_parallel::_EqualTo, 1155](#)
 - [first_argument_type, 1157](#)
 - [result_type, 1157](#)
 - [second_argument_type, 1157](#)
- [__gnu_parallel::_GuardedIterator, 1157](#)
 - [_GuardedIterator, 1158](#)
 - [operator_RAIter, 1158](#)
 - [operator<, 1159](#)
 - [operator<=, 1160](#)
 - [operator*, 1159](#)
 - [operator++, 1159](#)
- [__gnu_parallel::_IteratorPair, 1160](#)
 - [first, 1162](#)
 - [first_type, 1162](#)
 - [second, 1162](#)
 - [second_type, 1162](#)
- [__gnu_parallel::_IteratorTriple, 1163](#)
- [__gnu_parallel::_Job, 1164](#)
 - [_M_first, 1165](#)
 - [_M_last, 1165](#)
 - [_M_load, 1165](#)
- [__gnu_parallel::_Less, 1165](#)
 - [first_argument_type, 1167](#)
 - [result_type, 1167](#)
 - [second_argument_type, 1167](#)
- [__gnu_parallel::_Lexicographic, 1167](#)
 - [first_argument_type, 1169](#)
 - [result_type, 1169](#)
 - [second_argument_type, 1169](#)
- [__gnu_parallel::_LexicographicReverse, 1169](#)
 - [first_argument_type, 1171](#)
 - [result_type, 1171](#)
 - [second_argument_type, 1171](#)
- [__gnu_parallel::_LoserTree, 1171](#)
 - [_M_comp, 1174](#)
 - [_M_first_insert, 1174](#)
 - [_M_log_k, 1174](#)
 - [_M_losers, 1174](#)
 - [__delete_min_insert, 1173](#)
 - [__get_min_source, 1173](#)
 - [__insert_start, 1173](#)
- [__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1175](#)
 - [_M_comp, 1178](#)
 - [_M_first_insert, 1178](#)
 - [_M_log_k, 1178](#)
 - [_M_losers, 1178](#)
 - [__delete_min_insert, 1176](#)
 - [__get_min_source, 1176](#)
 - [__init_winner, 1177](#)
 - [__insert_start, 1177](#)
- [__gnu_parallel::_LoserTreeBase, 1179](#)
 - [~_LoserTreeBase, 1180](#)
 - [_LoserTreeBase, 1180](#)
 - [_M_comp, 1181](#)
 - [_M_first_insert, 1181](#)
 - [_M_log_k, 1182](#)
 - [_M_losers, 1182](#)
 - [__get_min_source, 1181](#)
 - [__insert_start, 1181](#)
- [__gnu_parallel::_LoserTreeBase::_Loser, 1182](#)
 - [_M_key, 1183](#)
 - [_M_source, 1183](#)
 - [_M_sup, 1183](#)
- [__gnu_parallel::_LoserTreePointer, 1184](#)
- [__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >, 1185](#)
- [__gnu_parallel::_LoserTreePointerBase, 1186](#)
- [__gnu_parallel::_LoserTreePointerBase::_Loser, 1187](#)
- [__gnu_parallel::_LoserTreePointerUnguarded, 1188](#)
- [__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >, 1189](#)

-
- __gnu_parallel::_-
 - LoserTreePointerUnguardedBase, 1191
 - __gnu_parallel::_LoserTreeTraits, 1192
 - _M_use_pointer, 1193
 - __gnu_parallel::_LoserTreeUnguarded, 1193
 - __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >, 1195
 - __gnu_parallel::_-
 - LoserTreeUnguardedBase, 1196
 - __gnu_parallel::_Multiplies, 1197
 - first_argument_type, 1199
 - result_type, 1199
 - second_argument_type, 1199
 - __gnu_parallel::_Nothing, 1199
 - operator(), 1200
 - __gnu_parallel::_PMWMSortingData, 1203
 - _M_num_threads, 1204
 - _M_offsets, 1204
 - _M_pieces, 1204
 - _M_samples, 1205
 - _M_source, 1205
 - _M_starts, 1205
 - _M_temporary, 1205
 - __gnu_parallel::_Piece, 1200
 - _M_begin, 1201
 - _M_end, 1201
 - __gnu_parallel::_Plus, 1201
 - first_argument_type, 1203
 - result_type, 1203
 - second_argument_type, 1203
 - __gnu_parallel::_PseudoSequence, 1206
 - _PseudoSequence, 1206
 - begin, 1207
 - end, 1207
 - __gnu_parallel::_-
 - PseudoSequenceIterator, 1207
 - __gnu_parallel::_QSBThreadLocal, 1208
 - _M_elements_leftover, 1210
 - _M_global, 1210
 - _M_initial, 1210
 - _M_leftover_parts, 1210
 - _M_num_threads, 1210
 - _Piece, 1209
 - _QSBThreadLocal, 1209
 - __gnu_parallel::_RandomNumber, 1211
 - _RandomNumber, 1211
 - __genrand_bits, 1212
 - operator(), 1212
 - __gnu_parallel::_-
 - RestrictedBoundedConcurrentQueue, 1212
 - ~_RestrictedBoundedConcurrentQueue, 1214
 - _RestrictedBoundedConcurrentQueue, 1214
 - pop_back, 1214
 - pop_front, 1214
 - push_front, 1214
 - __gnu_parallel::_SamplingSorter, 1215
 - __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >, 1216
 - __gnu_parallel::_Settings, 1216
 - accumulate_minimal_n, 1218
 - adjacent_difference_minimal_n, 1218
 - cache_line_size, 1219
 - count_minimal_n, 1219
 - fill_minimal_n, 1219
 - find_increasing_factor, 1219
 - find_initial_block_size, 1219
 - find_maximum_block_size, 1219
 - find_scale_factor, 1220
 - find_sequential_search_size, 1220
 - for_each_minimal_n, 1220
 - generate_minimal_n, 1220
 - get, 1218
 - L1_cache_size, 1220
 - L2_cache_size, 1220
 - max_element_minimal_n, 1220
 - merge_minimal_n, 1221
 - merge_oversampling, 1221
 - min_element_minimal_n, 1221
 - multiway_merge_minimal_k, 1221
 - multiway_merge_minimal_n, 1221
 - multiway_merge_oversampling, 1221
-

- nth_element_minimal_n, 1222
- partial_sort_minimal_n, 1222
- partial_sum_dilation, 1222
- partial_sum_minimal_n, 1222
- partition_chunk_share, 1222
- partition_chunk_size, 1222
- partition_minimal_n, 1223
- qsb_steals, 1223
- random_shuffle_minimal_n, 1223
- replace_minimal_n, 1223
- search_minimal_n, 1223
- set, 1218
- set_difference_minimal_n, 1223
- set_intersection_minimal_n, 1223
- set_symmetric_difference_minimal_n, 1224
- set_union_minimal_n, 1224
- sort_minimal_n, 1224
- sort_mwms_oversampling, 1224
- sort_qs_num_samples_preset, 1224
- sort_qsb_base_case_maximal_n, 1224
- TLB_size, 1225
- transform_minimal_n, 1225
- unique_copy_minimal_n, 1225
- __gnu_parallel::__SplitConsistently, 1225
- __gnu_parallel::__SplitConsistently<false, _RAIter, _Compare, _SortingPlacesIterator >, 1226
- __gnu_parallel::__SplitConsistently<true, _RAIter, _Compare, _SortingPlacesIterator >, 1227
- __gnu_parallel::__accumulate_binop_reduct, 1103
- __gnu_parallel::__accumulate_selector, 1103
- __M_finish_iterator, 1105
- operator(), 1104
- __gnu_parallel::__adjacent_difference_selector, 1105
- __M_finish_iterator, 1106
- __gnu_parallel::__adjacent_find_selector, 1107
- __M_sequential_algorithm, 1108
- operator(), 1108
- __gnu_parallel::__binder1st, 1108
- argument_type, 1110
- result_type, 1110
- __gnu_parallel::__binder2nd, 1110
- argument_type, 1112
- result_type, 1112
- __gnu_parallel::__count_if_selector, 1112
- __M_finish_iterator, 1113
- operator(), 1113
- __gnu_parallel::__count_selector, 1114
- __M_finish_iterator, 1115
- operator(), 1115
- __gnu_parallel::__fill_selector, 1116
- __M_finish_iterator, 1117
- operator(), 1117
- __gnu_parallel::__find_first_of_selector, 1117
- __M_sequential_algorithm, 1119
- operator(), 1119
- __gnu_parallel::__find_if_selector, 1119
- __M_sequential_algorithm, 1120
- operator(), 1121
- __gnu_parallel::__for_each_selector, 1121
- __M_finish_iterator, 1123
- operator(), 1122
- __gnu_parallel::__generate_selector, 1123
- __M_finish_iterator, 1124
- operator(), 1124
- __gnu_parallel::__generic_find_selector, 1125
- __gnu_parallel::__generic_for_each_selector, 1125
- __M_finish_iterator, 1127
- __gnu_parallel::__identity_selector, 1127
- __M_finish_iterator, 1128
- operator(), 1128
- __gnu_parallel::__inner_product_selector, 1129
- __M_finish_iterator, 1131
- __begin1_iterator, 1130
- __begin2_iterator, 1131
- __inner_product_selector, 1130
- operator(), 1130

- __gnu_parallel::__max_element_reduct, 1131
- __gnu_parallel::__min_element_reduct, 1132
- __gnu_parallel::__mismatch_selector, 1133
 - _M_sequential_algorithm, 1134
 - operator(), 1134
- __gnu_parallel::__multiway_merge_3_variant_sentinel_switch, 1134
- __gnu_parallel::__multiway_merge_3_variant_sentinel_switch<true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 1135
- __gnu_parallel::__multiway_merge_4_variant_sentinel_switch, 1136
- __gnu_parallel::__multiway_merge_4_variant_sentinel_switch<true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 1136
- __gnu_parallel::__multiway_merge_k_variant_sentinel_switch, 1137
- __gnu_parallel::__multiway_merge_k_variant_sentinel_switch<false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 1138
- __gnu_parallel::__replace_if_selector, 1139
 - _M_finish_iterator, 1140
 - __new_val, 1140
 - __replace_if_selector, 1140
 - operator(), 1140
- __gnu_parallel::__replace_selector, 1141
 - _M_finish_iterator, 1143
 - __new_val, 1142
 - __replace_selector, 1142
 - operator(), 1142
- __gnu_parallel::__transform1_selector, 1143
 - _M_finish_iterator, 1144
 - operator(), 1144
- __gnu_parallel::__transform2_selector, 1145
 - _M_finish_iterator, 1146
 - operator(), 1146
- __gnu_parallel::__unary_negate, 1146
 - argument_type, 1148
 - result_type, 1148
- __gnu_parallel::balanced_quicksort_tag, 1227
 - __get_num_threads, 1228
 - set_num_threads, 1228
- __gnu_parallel::balanced_tag, 1229
 - __get_num_threads, 1230
 - set_num_threads, 1230
- __gnu_parallel::constant_size_blocks_tag, 1230
- __gnu_parallel::default_parallel_tag, 1231
 - __get_num_threads, 1232
 - set_num_threads, 1232
- __gnu_parallel::equal_split_tag, 1233
- __gnu_parallel::exact_tag, 1234
 - __get_num_threads, 1235
 - set_num_threads, 1235
- __gnu_parallel::find_tag, 1235
- __gnu_parallel::growing_blocks_tag, 1236
- __gnu_parallel::multiway_mergesort_exact_tag, 1237
 - __get_num_threads, 1238
 - set_num_threads, 1238
- __gnu_parallel::multiway_mergesort_sampling_tag, 1239
 - __get_num_threads, 1240
 - set_num_threads, 1240
- __gnu_parallel::multiway_mergesort_tag, 1240
 - __get_num_threads, 1241
 - set_num_threads, 1241
- __gnu_parallel::omp_loop_static_tag, 1242
 - __get_num_threads, 1243
 - set_num_threads, 1243
- __gnu_parallel::omp_loop_tag, 1243
 - __get_num_threads, 1244

-
- set_num_threads, [1244](#)
 - __gnu_parallel::parallel_tag, [1245](#)
 - __get_num_threads, [1247](#)
 - parallel_tag, [1247](#)
 - set_num_threads, [1247](#)
 - __gnu_parallel::quicksort_tag, [1248](#)
 - __get_num_threads, [1249](#)
 - set_num_threads, [1249](#)
 - __gnu_parallel::sampling_tag, [1249](#)
 - __get_num_threads, [1250](#)
 - set_num_threads, [1250](#)
 - __gnu_parallel::sequential_tag, [1251](#)
 - __gnu_parallel::unbalanced_tag, [1251](#)
 - __get_num_threads, [1252](#)
 - set_num_threads, [1252](#)
 - __gnu_pbds, [454](#)
 - __gnu_pbds::associative_container_tag, [1253](#)
 - __gnu_pbds::basic_hash_table, [1254](#)
 - __gnu_pbds::basic_hash_tag, [1255](#)
 - __gnu_pbds::basic_tree, [1256](#)
 - __gnu_pbds::basic_tree_tag, [1257](#)
 - __gnu_pbds::binary_heap_tag, [1258](#)
 - __gnu_pbds::binomial_heap_tag, [1259](#)
 - __gnu_pbds::cc_hash_table, [1260](#)
 - __gnu_pbds::cc_hash_tag, [1262](#)
 - __gnu_pbds::container_base, [1263](#)
 - __gnu_pbds::container_tag, [1264](#)
 - __gnu_pbds::container_traits, [1265](#)
 - __gnu_pbds::detail::value_type_base<Key, Mapped, Allocator, false>, [1265](#)
 - __gnu_pbds::detail::value_type_base<Key, Mapped, Allocator, true>, [1266](#)
 - __gnu_pbds::detail::value_type_base<Key, null_mapped_type, Allocator, false>, [1267](#)
 - __gnu_pbds::detail::value_type_base<Key, null_mapped_type, Allocator, true>, [1268](#)
 - __gnu_pbds::gp_hash_table, [1269](#)
 - __gnu_pbds::gp_hash_tag, [1271](#)
 - __gnu_pbds::list_update, [1272](#)
 - __gnu_pbds::list_update_tag, [1274](#)
 - __gnu_pbds::null_mapped_type, [1275](#)
 - __gnu_pbds::ov_tree_tag, [1276](#)
 - __gnu_pbds::pairing_heap_tag, [1277](#)
 - __gnu_pbds::pat_trie_tag, [1277](#)
 - __gnu_pbds::priority_queue_tag, [1279](#)
 - __gnu_pbds::rb_tree_tag, [1279](#)
 - __gnu_pbds::rc_binomial_heap_tag, [1281](#)
 - __gnu_pbds::sequence_tag, [1281](#)
 - __gnu_pbds::splay_tree_tag, [1282](#)
 - __gnu_pbds::string_tag, [1283](#)
 - __gnu_pbds::thin_heap_tag, [1284](#)
 - __gnu_pbds::tree, [1285](#)
 - __gnu_pbds::tree_tag, [1287](#)
 - __gnu_pbds::trie, [1288](#)
 - __gnu_pbds::trie_tag, [1290](#)
 - __gnu_profile, [457](#)
 - __env_t, [463](#)
 - __get__global_lock, [463](#)
 - __profcxx_init, [463](#)
 - __report, [463](#)
 - __gnu_profile::__container_size_info, [1291](#)
 - __gnu_profile::__container_size_stack_info, [1292](#)
 - __gnu_profile::__hashfunc_info, [1293](#)
 - __gnu_profile::__hashfunc_stack_info, [1295](#)
 - __gnu_profile::__list2vector_info, [1296](#)
 - __gnu_profile::__map2umap_info, [1297](#)
 - __gnu_profile::__map2umap_stack_info, [1299](#)
 - __gnu_profile::__object_info_base, [1300](#)
 - __gnu_profile::__reentrance_guard, [1301](#)
 - __gnu_profile::__stack_hash, [1302](#)
 - __gnu_profile::__stack_info_base, [1302](#)
 - __gnu_profile::__trace_base, [1303](#)
 - __gnu_profile::__trace_container_size, [1304](#)
 - __gnu_profile::__trace_hash_func, [1304](#)
 - __gnu_profile::__trace_hashtable_size, [1306](#)
 - __gnu_profile::__trace_map2umap, [1306](#)
 - __gnu_profile::__trace_vector_size, [1308](#)
 - __gnu_profile::__trace_vector_to_list, [1308](#)
 - __gnu_profile::__vector2list_info, [1310](#)
-

- __gnu_profile::__vector2list_stack_info, 1311
- __gnu_profile::__warning_data, 1313
- __gnu_sequential, 464
- __heap_select
 - std, 608
- __init_winner
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1177
- __inner_product_selector
 - __gnu_parallel::__inner_product_selector, 1130
- __inplace_stable_partition
 - std, 608
- __inplace_stable_sort
 - std, 609
- __insert_start
 - __gnu_parallel::_LoserTree, 1173
 - __gnu_parallel::_LoserTree< false, _Tp, _Compare >, 1177
 - __gnu_parallel::_LoserTreeBase, 1181
- __insertion_sort
 - std, 609
- __introsort_loop
 - std, 610
- __invoke
 - std, 694
- __is_sorted
 - __gnu_parallel, 418
- __iterator_category
 - iterators, 274
- __lg
 - std, 610
- __match_flag
 - std::regex_constants, 752
- __median
 - SGIextensions, 16, 17
- __median_of_three_iterators
 - __gnu_parallel, 419
- __merge_adaptive
 - std, 610, 611
- __merge_advance
 - __gnu_parallel, 419
- __merge_advance_movc
 - __gnu_parallel, 420
- __merge_advance_usual
 - __gnu_parallel, 421
- __merge_backward
 - std, 611
- __merge_without_buffer
 - std, 612
- __move_median_first
 - std, 612, 613
- __new_val
 - __gnu_parallel::__replace_if_selector, 1140
 - __gnu_parallel::__replace_selector, 1142
- __num_bitmaps
 - __gnu_cxx::__detail, 378
- __num_blocks
 - __gnu_cxx::__detail, 379
- __num_get_type
 - std::basic_fstream, 1563
 - std::basic_ifstream, 1628
 - std::basic_ios, 1679
 - std::basic_iostream, 1713
 - std::basic_istream, 1776
 - std::basic_istream, 1830
 - std::basic_ofstream, 1883
 - std::basic_ostream, 1926
 - std::basic_ostringstream, 1971
 - std::basic_stringstream, 2137
- __num_put_type
 - std::basic_fstream, 1563
 - std::basic_ifstream, 1628
 - std::basic_ios, 1679
 - std::basic_iostream, 1713, 1714
 - std::basic_istream, 1777
 - std::basic_istream, 1830
 - std::basic_ofstream, 1883
 - std::basic_ostream, 1926
 - std::basic_ostringstream, 1971
 - std::basic_stringstream, 2137
- __parallel_merge_advance
 - __gnu_parallel, 421, 422
- __parallel_nth_element
 - __gnu_parallel, 423
- __parallel_partial_sort
 - __gnu_parallel, 423
- __parallel_partial_sum

-
- __gnu_parallel, 423
 - __parallel_partial_sum_basecase
 - __gnu_parallel, 424
 - __parallel_partial_sum_linear
 - __gnu_parallel, 424
 - __parallel_partition
 - __gnu_parallel, 425
 - __parallel_random_shuffle
 - __gnu_parallel, 426
 - __parallel_random_shuffle_drs
 - __gnu_parallel, 426
 - __parallel_random_shuffle_drs_pu
 - __gnu_parallel, 427
 - __parallel_sort
 - __gnu_parallel, 427–431
 - __parallel_sort_qs
 - __gnu_parallel, 432
 - __parallel_sort_qs_conquer
 - __gnu_parallel, 432
 - __parallel_sort_qs_divide
 - __gnu_parallel, 433
 - __parallel_sort_qsb
 - __gnu_parallel, 433
 - __parallel_unique_copy
 - __gnu_parallel, 434
 - __partition
 - std, 613
 - __profcxx_init
 - __gnu_profile, 463
 - __qsb_conquer
 - __gnu_parallel, 435
 - __qsb_divide
 - __gnu_parallel, 435
 - __qsb_local_sort_with_helping
 - __gnu_parallel, 436
 - __random_number_pow2
 - __gnu_parallel, 436
 - __rd_log2
 - __gnu_parallel, 437
 - __replace_if_selector
 - __gnu_parallel::__replace_if_selector, 1140
 - __replace_selector
 - __gnu_parallel::__replace_selector, 1142
 - __report
 - __gnu_profile, 463
 - __reverse
 - std, 613
 - __rotate
 - std, 614
 - __rotate_adaptive
 - std, 614
 - __round_up_to_pow2
 - __gnu_parallel, 437
 - __search_n
 - std, 615
 - __search_template
 - __gnu_parallel, 438
 - __sequential_multiway_merge
 - __gnu_parallel, 438
 - __sequential_random_shuffle
 - __gnu_parallel, 439
 - __shrink
 - __gnu_parallel, 439
 - __shrink_and_double
 - __gnu_parallel, 440
 - __stable_partition_adaptive
 - std, 616
 - __static_pointer_cast
 - __gnu_cxx, 365
 - __streambuf_type
 - __gnu_cxx::enc_filebuf, 887
 - __gnu_cxx::stdio_filebuf, 956
 - __gnu_cxx::stdio_sync_filebuf, 988
 - std::basic_filebuf, 1527
 - std::basic_streambuf, 2023
 - std::basic_stringbuf, 2106
 - __syntax_option
 - std::regex_constants, 752
 - __unguarded_insertion_sort
 - std, 616
 - __unguarded_linear_insert
 - std, 616, 617
 - __unguarded_partition
 - std, 617
 - __unguarded_partition_pivot
 - std, 617, 618
 - __unique_copy
 - std, 618, 619
 - __valid_range
 - __gnu_debug, 388
-

- __valid_range_aux
 - __gnu_debug, 388, 389
- __valid_range_aux2
 - __gnu_debug, 389
- __verbose_terminate_handler
 - exceptions, 36
- __versa_string
 - __gnu_cxx::__versa_string, 802–805
- __yield
 - __gnu_parallel, 440
- a
 - std::extreme_value_distribution, 2442
 - std::weibull_distribution, 3178
- abi, 464
- abs
 - complex_numbers, 44
- accumulate
 - std, 620, 621
- accumulate_minimal_n
 - __gnu_parallel::_Settings, 1218
- acos
 - complex_numbers, 44
- acosh
 - complex_numbers, 44
- Adaptors for pointers to functions, 260
- Adaptors for pointers to members, 262
- addressof
 - std, 621
- adjacent_difference
 - std, 621, 622
- adjacent_difference_minimal_n
 - __gnu_parallel::_Settings, 1218
- adjacent_find
 - non_mutating_algorithms, 154
- adjustfield
 - std::basic_fstream, 1615
 - std::basic_ifstream, 1668
 - std::basic_ios, 1699
 - std::basic_iostream, 1764
 - std::basic_istream, 1815
 - std::basic_istreamstream, 1870
 - std::basic_ofstream, 1914
 - std::basic_ostream, 1956
 - std::basic_ostreamstream, 2002
 - std::basic_stringstream, 2188
 - std::ios_base, 2563
- advance
 - std, 622
- algo.h, 3181
- algbase.h, 3195
- algorithm, 3197, 3199
- algorithmfwd.h, 3199, 3206
- Algorithms, 128
- all
 - std::bitset, 2217
 - std::locale, 2636
- all_of
 - non_mutating_algorithms, 155
- allocate_shared
 - pointer_abstractions, 68
 - std::shared_ptr, 3011
- allocator.h, 3219
- allocator_type
 - std::set, 2978
- Allocators, 202
- alpha
 - std::gamma_distribution, 2499
- any
 - std::bitset, 2217
- any_of
 - non_mutating_algorithms, 155
- app
 - std::basic_fstream, 1615
 - std::basic_ifstream, 1669
 - std::basic_ios, 1699
 - std::basic_iostream, 1764
 - std::basic_istream, 1815
 - std::basic_istreamstream, 1870
 - std::basic_ofstream, 1914
 - std::basic_ostream, 1956
 - std::basic_ostreamstream, 2002
 - std::basic_stringstream, 2189
 - std::ios_base, 2563
- append
 - __gnu_cxx::__versa_string, 806–808
 - __gnu_debug::basic_string, 1060–1062
 - std::basic_string, 2053–2056

-
- apply
 - numeric_arrays, 91
 - apply_generator
 - __gnu_cxx::typelist, 379
 - arg
 - complex_numbers, 44
 - argument_type
 - __gnu_cxx::__detail::_Ffit_finder, 782
 - __gnu_cxx::binary_compose, 874
 - __gnu_cxx::select1st, 947
 - __gnu_cxx::select2nd, 948
 - __gnu_cxx::subtractive_rng, 1010
 - __gnu_cxx::unary_compose, 1024
 - __gnu_parallel::__binder1st, 1110
 - __gnu_parallel::__binder2nd, 1112
 - __gnu_parallel::__unary_negate, 1148
 - std::Maybe_unary_or_binary_function<_Res, _T1 >, 1463
 - std::binder1st, 2203
 - std::binder2nd, 2205
 - std::const_mem_fun_ref_t, 2295
 - std::const_mem_fun_t, 2297
 - std::hash< __gnu_cxx::throw_value_limit >, 2520
 - std::hash< __gnu_cxx::throw_value_random >, 2522
 - std::hash< __shared_ptr<_Tp, _Lp> >, 2525
 - std::hash< shared_ptr<_Tp> >, 2528
 - std::hash< unique_ptr<_Tp, _Dp> >, 2532
 - std::logical_not, 2648
 - std::mem_fun_ref_t, 2695
 - std::mem_fun_t, 2697
 - std::negate, 2788
 - std::pointer_to_unary_function, 2918
 - std::unary_function, 3118
 - std::unary_negate, 3120
 - Arithmetic Classes, 256
 - array, 3220
 - array_allocator.h, 3221
 - asin
 - complex_numbers, 45
 - asinh
 - complex_numbers, 45
 - assign
 - __gnu_cxx::__versa_string, 809–812
 - __gnu_debug::basic_string, 1062–1065
 - std::basic_regex, 2013–2016
 - std::basic_string, 2056–2059
 - std::deque, 2399, 2400
 - std::forward_list, 2457
 - std::list, 2611, 2612
 - std::vector, 3156
 - assoc_container.hpp, 3222
 - assoc_laguerre
 - tr1_math_spec_func, 113
 - assoc_legendre
 - tr1_math_spec_func, 113
 - Associative, 29
 - at
 - __gnu_cxx::__versa_string, 813
 - __gnu_debug::basic_string, 1066
 - std::basic_string, 2060
 - std::deque, 2400, 2401
 - std::map, 2661
 - std::vector, 3157
 - atan
 - complex_numbers, 45
 - atanh
 - complex_numbers, 45
 - ate
 - std::basic_fstream, 1615
 - std::basic_ifstream, 1669
 - std::basic_ios, 1699
 - std::basic_iostream, 1764
 - std::basic_istream, 1815
 - std::basic_istreamstream, 1870
 - std::basic_ofstream, 1914
 - std::basic_ostream, 1957
 - std::basic_ostreamstream, 2002
 - std::basic_stringstream, 2189
 - std::ios_base, 2563
 - atomic, 3223
 - atomic_0.h, 3228
 - atomic_2.h, 3228
-

- atomic_base.h, 3229
- atomic_char
 - atomics, 210
- atomic_char16_t
 - atomics, 210
- atomic_char32_t
 - atomics, 210
- atomic_int
 - atomics, 210
- atomic_llong
 - atomics, 210
- atomic_long
 - atomics, 211
- atomic_schar
 - atomics, 211
- atomic_short
 - atomics, 211
- atomic_uchar
 - atomics, 211
- atomic_uint
 - atomics, 211
- atomic_ullong
 - atomics, 211
- atomic_ulong
 - atomics, 211
- atomic_ushort
 - atomics, 212
- atomic_wchar_t
 - atomics, 212
- atomic_word.h, 3231
- atomicfwd_c.h, 3231
- atomicfwd_cxx.h, 3232
- atomicity.h, 3233
- Atoms, 204
- atomics
 - _GLIBCXX_ATOMIC_-
PROPERTY, 210
 - atomic_char, 210
 - atomic_char16_t, 210
 - atomic_char32_t, 210
 - atomic_int, 210
 - atomic_llong, 210
 - atomic_long, 211
 - atomic_schar, 211
 - atomic_short, 211
 - atomic_uchar, 211
 - atomic_uint, 211
 - atomic_ullong, 211
 - atomic_ulong, 211
 - atomic_ushort, 212
 - atomic_wchar_t, 212
 - kill_dependency, 212
 - memory_order, 212
- auto_ptr
 - std::auto_ptr, 1509, 1510
- auto_ptr.h, 3234
- awk
 - std::regex_constants, 754
- b
 - std::extreme_value_distribution,
2442
 - std::weibull_distribution, 3178
- back
 - __gnu_cxx::__versa_string, 814
 - __gnu_debug::basic_string, 1066,
1067
 - std::basic_string, 2061
 - std::deque, 2401, 2402
 - std::list, 2612, 2613
 - std::queue, 2932
 - std::vector, 3158
- back_insert_iterator
 - std::back_insert_iterator, 1516
- back_inserter
 - iterators, 274
- bad
 - std::basic_fstream, 1571
 - std::basic_ifstream, 1634
 - std::basic_ios, 1685
 - std::basic_iostream, 1721
 - std::basic_istream, 1782
 - std::basic_istreamstream, 1836
 - std::basic_ofstream, 1889
 - std::basic_ostream, 1932
 - std::basic_ostreamstream, 1977
 - std::basic_stringstream, 2145
- badbit
 - std::basic_fstream, 1616
 - std::basic_ifstream, 1669
 - std::basic_ios, 1700
 - std::basic_iostream, 1764

- std::basic_istream, 1816
- std::basic_istreamstream, 1871
- std::basic_ofstream, 1915
- std::basic_ostream, 1957
- std::basic_ostringstream, 2002
- std::basic_stringstream, 2189
- std::ios_base, 2564
- balanced_quicksort.h, 3234
- base
 - __gnu_debug::_Safe_iterator, 1034
 - std::discard_block_engine, 2418
 - std::independent_bits_engine, 2537
 - std::reverse_iterator, 2969
 - std::shuffle_order_engine, 3015
- base.h, 3235, 3237
- basefield
 - std::basic_fstream, 1616
 - std::basic_ifstream, 1669
 - std::basic_ios, 1700
 - std::basic_iostream, 1765
 - std::basic_istream, 1816
 - std::basic_istreamstream, 1871
 - std::basic_ofstream, 1915
 - std::basic_ostream, 1957
 - std::basic_ostringstream, 2002
 - std::basic_stringstream, 2189
 - std::ios_base, 2564
- basic
 - std::regex_constants, 754
- basic_file.h, 3238
- basic_filebuf
 - std::basic_filebuf, 1529
- basic_fstream
 - std::basic_fstream, 1569, 1570
- basic_ifstream
 - std::basic_ifstream, 1633
- basic_ios
 - std::basic_ios, 1684
- basic_ios.h, 3238
- basic_ios.tcc, 3239
- basic_iostream
 - std::basic_iostream, 1720
- basic_istream
 - std::basic_istream, 1781
- basic_istreamstream
 - std::basic_istreamstream, 1835
- basic_iterator.h, 3239
- basic_ofstream
 - std::basic_ofstream, 1887, 1888
- basic_ostream
 - std::basic_ostream, 1931
- basic_ostringstream
 - std::basic_ostringstream, 1975
- basic_regex
 - std::basic_regex, 2010–2012
- basic_streambuf
 - std::basic_streambuf, 2025
- basic_string
 - std::basic_string, 2050–2052
- basic_string.h, 3239
- basic_string.tcc, 3243
- basic_stringbuf
 - std::basic_stringbuf, 2108
- basic_stringstream
 - std::basic_stringstream, 2143, 2144
- basic_types.hpp, 3244
- before_begin
 - std::forward_list, 2458
- beg
 - std::basic_fstream, 1616
 - std::basic_ifstream, 1670
 - std::basic_ios, 1700
 - std::basic_iostream, 1765
 - std::basic_istream, 1816
 - std::basic_istreamstream, 1871
 - std::basic_ofstream, 1915
 - std::basic_ostream, 1957
 - std::basic_ostringstream, 2003
 - std::basic_stringstream, 2189
 - std::ios_base, 2564
- begin
 - __gnu_cxx::__versa_string, 814
 - __gnu_cxx::temporary_buffer, 1013
 - __gnu_debug::basic_string, 1067
 - __gnu_parallel::_PseudoSequence, 1207
 - numeric_arrays, 92
 - std, 623, 624
 - std::_Temporary_buffer, 1479
 - std::basic_string, 2061
 - std::deque, 2402
 - std::forward_list, 2458

- std::list, 2613
- std::map, 2662
- std::match_results, 2684
- std::multimap, 2754
- std::multiset, 2773
- std::set, 2984
- std::vector, 3158
- Bernoulli, 296
- bernoulli_distribution
 - std::bernoulli_distribution, 2196
- beta
 - std::gamma_distribution, 2499
 - tr1_math_spec_func, 114
- binary
 - std::basic_fstream, 1616
 - std::basic_ifstream, 1670
 - std::basic_ios, 1700
 - std::basic_iostream, 1765
 - std::basic_istream, 1816
 - std::basic_istream, 1871
 - std::basic_ofstream, 1915
 - std::basic_ostream, 1958
 - std::basic_ostringstream, 2003
 - std::basic_stringstream, 2190
 - std::ios_base, 2564
- Binary Search, 196
- binary_search
 - binary_search_algorithms, 198
- binary_search_algorithms
 - binary_search, 198
 - equal_range, 199
 - lower_bound, 200
 - upper_bound, 201, 202
- bind
 - binders, 127
 - std, 624
- bind1st
 - binders, 127
- bind2nd
 - binders, 127
- Binder Classes, 125
- binders
 - _GLIBCXX_DEPRECATED_ - ATTR, 127
 - bind, 127
 - bind1st, 127
 - bind2nd, 127
 - binders.h, 3244
 - bitmap_allocator.h, 3245
 - _BALLOC_ALIGN_BYTES, 3247
 - bitset, 3247–3249
 - std::bitset, 2215, 2216
 - boolalpha
 - std, 624
 - std::basic_fstream, 1617
 - std::basic_ifstream, 1670
 - std::basic_ios, 1701
 - std::basic_iostream, 1765
 - std::basic_istream, 1816
 - std::basic_istream, 1872
 - std::basic_ofstream, 1916
 - std::basic_ostream, 1958
 - std::basic_ostringstream, 2003
 - std::basic_stringstream, 2190
 - std::ios_base, 2565
 - Boolean Operations Classes, 258
 - boost_concept_check.h, 3250
 - boost_sp_counted_base.h, 3251

c

 - std::queue, 2934
 - c++0x_warning.h, 3252
 - c++allocator.h, 3252
 - c++config.h, 3252
 - c++io.h, 3257
 - c++locale.h, 3258
 - c++locale_internal.h, 3259
 - c_str
 - __gnu_cxx::__versa_string, 814
 - __gnu_debug::basic_string, 1067
 - std::basic_string, 2061
 - cache_line_size
 - __gnu_parallel::_Settings, 1219
 - call_once
 - mutexes, 71
 - std::once_flag, 2884
 - capacity
 - __gnu_cxx::__versa_string, 815
 - __gnu_debug::basic_string, 1067
 - std::basic_string, 2062
 - std::vector, 3159
 - cassert, 3259

-
- category
 - std::locale, [2631](#)
 - cbefore_begin
 - std::forward_list, [2459](#)
 - cbegin
 - __gnu_cxx::__versa_string, [815](#)
 - __gnu_debug::basic_string, [1067](#)
 - std::basic_string, [2062](#)
 - std::deque, [2402](#)
 - std::forward_list, [2459](#)
 - std::list, [2613](#)
 - std::map, [2662](#)
 - std::match_results, [2684](#)
 - std::multimap, [2754](#)
 - std::multiset, [2773](#)
 - std::set, [2984](#)
 - std::vector, [3159](#)
 - ccomplex, [3259](#)
 - cctype, [3260](#)
 - cend
 - __gnu_cxx::__versa_string, [815](#)
 - __gnu_debug::basic_string, [1068](#)
 - std::basic_string, [2062](#)
 - std::deque, [2402](#)
 - std::forward_list, [2459](#)
 - std::list, [2613](#)
 - std::map, [2662](#)
 - std::match_results, [2684](#)
 - std::multimap, [2754](#)
 - std::multiset, [2774](#)
 - std::set, [2984](#)
 - std::vector, [3159](#)
 - cerr
 - std, [694](#)
 - cerrno, [3261](#)
 - cfenv, [3261](#), [3262](#)
 - cfloat, [3262](#)
 - char_traits.h, [3263](#)
 - char_type
 - __gnu_cxx::enc_filebuf, [888](#)
 - __gnu_cxx::stdio_filebuf, [956](#)
 - __gnu_cxx::stdio_sync_filebuf, [988](#)
 - std::__ctype_abstract_base, [1323](#)
 - std::basic_filebuf, [1527](#)
 - std::basic_fstream, [1564](#)
 - std::basic_ifstream, [1629](#)
 - std::basic_ios, [1679](#)
 - std::basic_iostream, [1714](#)
 - std::basic_istream, [1777](#)
 - std::basic_istreamstream, [1831](#)
 - std::basic_ofstream, [1883](#)
 - std::basic_ostream, [1926](#)
 - std::basic_ostreamstream, [1971](#)
 - std::basic_streambuf, [2023](#)
 - std::basic_stringbuf, [2106](#)
 - std::basic_stringstream, [2138](#)
 - std::collate, [2274](#)
 - std::collate_byname, [2281](#)
 - std::ctype, [2300](#)
 - std::ctype< char >, [2316](#)
 - std::ctype< wchar_t >, [2330](#)
 - std::ctype_byname, [2351](#)
 - std::ctype_byname< char >, [2366](#)
 - std::istreambuf_iterator, [2586](#)
 - std::messages, [2700](#)
 - std::messages_byname, [2705](#)
 - std::money_get, [2713](#)
 - std::money_put, [2719](#)
 - std::moneypunct, [2725](#)
 - std::moneypunct_byname, [2738](#)
 - std::num_get, [2805](#)
 - std::num_put, [2825](#)
 - std::numpunct, [2871](#)
 - std::numpunct_byname, [2879](#)
 - std::ostream_iterator, [2885](#)
 - std::ostreambuf_iterator, [2890](#)
 - std::time_get, [3043](#)
 - std::time_get_byname, [3054](#)
 - std::time_put, [3064](#)
 - std::time_put_byname, [3069](#)
 - checkers.h, [3263](#)
 - chrono, [3264](#)
 - cin
 - std, [694](#)
 - cinttypes, [3267](#), [3268](#)
 - ciso646, [3268](#)
 - classic
 - std::locale, [2633](#)
 - classic_table
 - std::ctype< char >, [2317](#)
 - std::ctype_byname< char >, [2367](#)
 - clear
-

- __gnu_cxx::__versa_string, 815
- __gnu_debug::basic_string, 1068
- std::basic_fstream, 1572
- std::basic_ifstream, 1634
- std::basic_ios, 1685
- std::basic_iostream, 1721
- std::basic_istream, 1782
- std::basic_istreamstream, 1836
- std::basic_ofstream, 1889
- std::basic_ostream, 1932
- std::basic_ostreamstream, 1977
- std::basic_string, 2062
- std::basic_stringstream, 2145
- std::deque, 2403
- std::forward_list, 2459
- std::list, 2614
- std::map, 2663
- std::multimap, 2754
- std::multiset, 2774
- std::set, 2984
- std::vector, 3159
- climits, 3268, 3269
- clocale, 3269
- clog
 - std, 694
- close
 - __gnu_cxx::enc_filebuf, 890
 - __gnu_cxx::stdio_filebuf, 959
 - std::basic_filebuf, 1530
 - std::basic_fstream, 1572
 - std::basic_ifstream, 1635
 - std::basic_ostream, 1890
- cmath, 3270, 3273, 3276
- code
 - std::regex_error, 2948
- codecvt.h, 3276
- codecvt_specializations.h, 3277
- collate
 - std::collate, 2274
 - std::locale, 2636
 - std::regex_constants, 755
- combine
 - std::locale, 2633
- comp_ellint_1
 - tr1_math_spec_func, 114
- comp_ellint_2
 - tr1_math_spec_func, 114
- comp_ellint_3
 - tr1_math_spec_func, 114
- compare
 - __gnu_cxx::__versa_string, 816–819
 - __gnu_debug::basic_string, 1068–1071
 - std::basic_string, 2063–2066
 - std::collate, 2275
 - std::collate_byname, 2281
 - std::sub_match, 3033, 3034
- Comparison Classes, 257
- compatibility.h, 3278
- compiletime_settings.h, 3279
- _GLIBCXX_ASSERTIONS, 3280
- _GLIBCXX_CALL, 3280
- _GLIBCXX_RANDOM_-SHUFFLE_CONSIDER_L1, 3280
- _GLIBCXX_RANDOM_-SHUFFLE_CONSIDER_TLB, 3280
- _GLIBCXX_SCALE_DOWN_-FPU, 3281
- _GLIBCXX_VERBOSE_LEVEL, 3281
- complex, 3281, 3285, 3286
 - std::complex, 2286
- Complex Numbers, 39
- complex.h, 3287
- complex_numbers
 - abs, 44
 - acos, 44
 - acosh, 44
 - arg, 44
 - asin, 45
 - asinh, 45
 - atan, 45
 - atanh, 45
 - conj, 45
 - cos, 45
 - cosh, 45
 - exp, 46
 - fabs, 46
 - log, 46

- log10, [46](#)
- norm, [46](#)
- operator<<, [50](#)
- operator>>, [52](#)
- operator*, [47](#)
- operator*=[, 48](#)
- operator+, [48](#)
- operator+=, [49](#)
- operator-, [49](#)
- operator=, [49](#)
- operator/, [50](#)
- operator/=, [50](#)
- operator=, [51](#)
- operator==, [51](#)
- polar, [52](#)
- pow, [52](#)
- sin, [53](#)
- sinh, [53](#)
- sqrt, [53](#)
- tan, [53](#)
- tanh, [53](#)
- compose1
 - SGIextensions, [19](#)
- compose2
 - SGIextensions, [19](#)
- concept_check.h, [3288](#)
- concurrency.h, [3288](#)
- Concurrency, [32](#)
- cond_dealtor.hpp, [3289](#)
- Condition Variables, [54](#)
- condition_variable, [3289](#)
- condition_variables
 - cv_status, [54](#)
- conf_hyperg
 - tr1_math_spec_func, [114](#)
- conj
 - complex_numbers, [45](#)
- const_iterator
 - std::set, [2978](#)
- const_pointer
 - std::set, [2979](#)
- const_pointer_cast
 - std, [625](#)
- const_reference
 - std::set, [2979](#)
- const_reverse_iterator
 - std::set, [2979](#)
- constant0
 - SGIextensions, [19](#)
- constant1
 - SGIextensions, [19](#)
- constant2
 - SGIextensions, [19](#)
- constructors_destructor_fn_imps.hpp,
[3290](#)
- container_base_dispatch.hpp, [3291](#)
- container_type
 - std::back_insert_iterator, [1515](#)
 - std::front_insert_iterator, [2476](#)
 - std::insert_iterator, [2547](#)
- Containers, [26](#)
- copy
 - __gnu_cxx::__versa_string, [820](#)
 - __gnu_debug::basic_string, [1071](#)
 - mutating_algorithms, [132](#)
 - std::basic_string, [2066](#)
- copy_backward
 - mutating_algorithms, [132](#)
- copy_exception
 - exceptions, [36](#)
- copy_if
 - mutating_algorithms, [133](#)
- copy_n
 - mutating_algorithms, [133](#)
 - SGIextensions, [20](#)
- copyfmt
 - std::basic_fstream, [1572](#)
 - std::basic_ifstream, [1635](#)
 - std::basic_ios, [1685](#)
 - std::basic_iostream, [1722](#)
 - std::basic_istream, [1783](#)
 - std::basic_istreamstream, [1837](#)
 - std::basic_ofstream, [1890](#)
 - std::basic_ostream, [1933](#)
 - std::basic_ostringstream, [1978](#)
 - std::basic_stringstream, [2146](#)
- cos
 - complex_numbers, [45](#)
- cosh
 - complex_numbers, [45](#)
- count
 - non_mutating_algorithms, [156](#)

- std::bitset, 2217
- std::map, 2663
- std::multimap, 2755
- std::multiset, 2774
- std::set, 2985
- count_if
 - non_mutating_algorithms, 156
- count_minimal_n
 - __gnu_parallel::_Settings, 1219
- cout
 - std, 694
- cpp_type_traits.h, 3291
- cpu_defines.h, 3292
- crbegin
 - __gnu_cxx::__versa_string, 820
 - __gnu_debug::basic_string, 1072
 - std::basic_string, 2067
 - std::deque, 2403
 - std::list, 2614
 - std::map, 2663
 - std::multimap, 2755
 - std::multiset, 2774
 - std::set, 2985
 - std::vector, 3160
- ceref
 - std, 625
- cregex_token_iterator
 - regex, 223
- crend
 - __gnu_cxx::__versa_string, 821
 - __gnu_debug::basic_string, 1072
 - std::basic_string, 2067
 - std::deque, 2403
 - std::list, 2614
 - std::map, 2663
 - std::multimap, 2755
 - std::multiset, 2775
 - std::set, 2985
 - std::vector, 3160
- csetjmp, 3292
- cshift
 - numeric_arrays, 92
- csignal, 3292
- cstdarg, 3293
- cstdbool, 3293, 3294
- cstddef, 3294
- cstdint, 3294, 3295
- cstdio, 3295, 3296
- cstdlib, 3296, 3297
- cstring, 3297
- csub_match
 - regex, 223
- ctgmach, 3298
- ctime, 3298, 3299
- ctype
 - std::ctype< char >, 2316
 - std::ctype< wchar_t >, 2331
 - std::locale, 2637
- ctype_base.h, 3299
- ctype_inline.h, 3300
- ctype_noninline.h, 3300
- cur
 - std::basic_fstream, 1617
 - std::basic_ifstream, 1670
 - std::basic_ios, 1701
 - std::basic_iostream, 1766
 - std::basic_istream, 1817
 - std::basic_istream, 1872
 - std::basic_ofstream, 1916
 - std::basic_ostream, 1958
 - std::basic_ostringstream, 2003
 - std::basic_stringstream, 2190
 - std::ios_base, 2565
- curr_symbol
 - std::moneypunct, 2727
 - std::moneypunct_byname, 2739
- current_exception
 - exceptions, 36
- cv_status
 - condition_variables, 54
- cwchar, 3300, 3301
- cwctype, 3301, 3302
- cxxabi-forced.h, 3303
- cxxabi.h, 3303
- cxxabi_tweaks.h, 3305
- cyl_bessel_i
 - tr1_math_spec_func, 114
- cyl_bessel_j
 - tr1_math_spec_func, 115
- cyl_bessel_k
 - tr1_math_spec_func, 115
- cyl_neumann

- tr1_math_spec_func, 115
- data
 - __gnu_cxx::__versa_string, 821
 - __gnu_debug::basic_string, 1072
 - std::basic_string, 2067
 - std::vector, 3160
- date_order
 - std::time_get, 3044
 - std::time_get_byname, 3054
- debug.h, 3305
- debug_allocator.h, 3306
- debug_map_base.hpp, 3307
- dec
 - std, 625
 - std::basic_fstream, 1617
 - std::basic_ifstream, 1670
 - std::basic_ios, 1701
 - std::basic_iostream, 1766
 - std::basic_istream, 1817
 - std::basic_istreamstream, 1872
 - std::basic_ofstream, 1916
 - std::basic_ostream, 1958
 - std::basic_ostreamstream, 2003
 - std::basic_stringstream, 2190
 - std::ios_base, 2565
- decimal, 3307
- Decimal Floating-Point Arithmetic, 124
- decimal128
 - std::decimal::decimal128, 2379
- decimal32
 - std::decimal::decimal32, 2381
- decimal32_to_long_long
 - std::decimal, 749
- decimal64
 - std::decimal::decimal64, 2383
- decimal_point
 - std::moneypunct, 2727
 - std::moneypunct_byname, 2739
 - std::numpunct, 2872
 - std::numpunct_byname, 2879
- denorm_absent
 - std, 605
- denorm_indeterminate
 - std, 605
- denorm_present
 - std, 606
- denorm_min
 - std::numeric_limits, 2839
- densities
 - std::piecewise_constant_distribution, 2904
 - std::piecewise_linear_distribution, 2909
- deque, 3319, 3320
 - std::deque, 2391–2393
- deque.tcc, 3321
- Diagnostics, 31
- difference_type
 - std::back_insert_iterator, 1515
 - std::front_insert_iterator, 2476
 - std::insert_iterator, 2547
 - std::istream_iterator, 2583
 - std::istreambuf_iterator, 2586
 - std::iterator, 2590
 - std::ostream_iterator, 2885
 - std::ostreambuf_iterator, 2890
 - std::raw_storage_iterator, 2943
 - std::reverse_iterator, 2967
 - std::set, 2979
- digits
 - std::__numeric_limits_base, 1410
 - std::numeric_limits, 2840
- digits10
 - std::__numeric_limits_base, 1410
 - std::numeric_limits, 2841
- discard
 - std::discard_block_engine, 2418
 - std::independent_bits_engine, 2537
 - std::linear_congruential_engine, 2600
 - std::shuffle_order_engine, 3015
- discard_block_engine
 - std::discard_block_engine, 2416, 2417
- distance
 - SGIextensions, 20
 - std, 625
- do_compare
 - std::collate, 2275
 - std::collate_byname, 2281
- do_curr_symbol

- std::moneypunct, 2727
- std::moneypunct_byname, 2739
- do_date_order
 - std::time_get, 3044
 - std::time_get_byname, 3055
- do_decimal_point
 - std::moneypunct, 2728
 - std::moneypunct_byname, 2740
 - std::numpunct, 2873
 - std::numpunct_byname, 2879
- do_falsename
 - std::numpunct, 2873
 - std::numpunct_byname, 2880
- do_frac_digits
 - std::moneypunct, 2728
 - std::moneypunct_byname, 2740
- do_get
 - std::money_get, 2714
 - std::num_get, 2806–2812
- do_get_date
 - std::time_get, 3044
 - std::time_get_byname, 3055
- do_get_monthname
 - std::time_get, 3045
 - std::time_get_byname, 3056
- do_get_time
 - std::time_get, 3045
 - std::time_get_byname, 3056
- do_get_weekday
 - std::time_get, 3046
 - std::time_get_byname, 3057
- do_get_year
 - std::time_get, 3047
 - std::time_get_byname, 3058
- do_grouping
 - std::moneypunct, 2729
 - std::moneypunct_byname, 2740
 - std::numpunct, 2873
 - std::numpunct_byname, 2880
- do_hash
 - std::collate, 2276
 - std::collate_byname, 2282
- do_is
 - std::__ctype_abstract_base, 1324
 - std::ctype, 2301
 - std::ctype< wchar_t >, 2331, 2332
- std::ctype_byname, 2352
- do_narrow
 - std::__ctype_abstract_base, 1324, 1325
 - std::ctype, 2302
 - std::ctype< char >, 2317
 - std::ctype< wchar_t >, 2333, 2334
 - std::ctype_byname, 2352, 2353
 - std::ctype_byname< char >, 2367
- do_neg_format
 - std::moneypunct, 2729
 - std::moneypunct_byname, 2741
- do_negative_sign
 - std::moneypunct, 2729
 - std::moneypunct_byname, 2741
- do_out
 - std::__codecvt_abstract_base, 1317
 - std::codecvt, 2245
 - std::codecvt< _InternT, _ExternT, encoding_state >, 2251
 - std::codecvt< char, char, mbstate_t >, 2256
 - std::codecvt< wchar_t, char, mbstate_t >, 2261
 - std::codecvt_byname, 2268
- do_pos_format
 - std::moneypunct, 2730
 - std::moneypunct_byname, 2742
- do_positive_sign
 - std::moneypunct, 2730
 - std::moneypunct_byname, 2742
- do_put
 - std::money_put, 2720
 - std::num_put, 2826–2829
 - std::time_put, 3065
 - std::time_put_byname, 3070
- do_scan_is
 - std::__ctype_abstract_base, 1326
 - std::ctype, 2303
 - std::ctype< wchar_t >, 2335
 - std::ctype_byname, 2354
- do_scan_not
 - std::__ctype_abstract_base, 1326
 - std::ctype, 2303
 - std::ctype< wchar_t >, 2336
 - std::ctype_byname, 2354

- do_thousands_sep
 - std::moneypunct, 2731
 - std::moneypunct_byname, 2743
 - std::numpunct, 2874
 - std::numpunct_byname, 2880
- do_tolower
 - std::__ctype_abstract_base, 1327
 - std::ctype, 2304
 - std::ctype< char >, 2318
 - std::ctype< wchar_t >, 2337, 2338
 - std::ctype_byname, 2355
 - std::ctype_byname< char >, 2368
- do_toupper
 - std::__ctype_abstract_base, 1328
 - std::ctype, 2305
 - std::ctype< char >, 2319
 - std::ctype< wchar_t >, 2339, 2340
 - std::ctype_byname, 2356
 - std::ctype_byname< char >, 2369
- do_transform
 - std::collate, 2276
 - std::collate_byname, 2282
- do_truename
 - std::numpunct, 2874
 - std::numpunct_byname, 2881
- do_widen
 - std::__ctype_abstract_base, 1329
 - std::ctype, 2305, 2306
 - std::ctype< char >, 2319, 2320
 - std::ctype< wchar_t >, 2340, 2341
 - std::ctype_byname, 2356, 2357
 - std::ctype_byname< char >, 2369, 2370
- duration_cast
 - std::chrono, 738
- dynamic_pointer_cast
 - std, 626
- eback
 - __gnu_cxx::enc_filebuf, 890
 - __gnu_cxx::stdio_filebuf, 960
 - __gnu_cxx::stdio_sync_filebuf, 990
 - std::basic_filebuf, 1530
 - std::basic_streambuf, 2026
 - std::basic_stringbuf, 2108
- ECMAScript
 - std::regex_constants, 755
- egptr
 - __gnu_cxx::enc_filebuf, 890
 - __gnu_cxx::stdio_filebuf, 960
 - __gnu_cxx::stdio_sync_filebuf, 990
 - std::basic_filebuf, 1531
 - std::basic_streambuf, 2026
 - std::basic_stringbuf, 2109
- egrep
 - std::regex_constants, 755
- element_type
 - std::auto_ptr, 1509
- ellint_1
 - tr1_math_spec_func, 115
- ellint_2
 - tr1_math_spec_func, 115
- ellint_3
 - tr1_math_spec_func, 116
- emplace
 - std::deque, 2403
 - std::list, 2614
 - std::vector, 3160
- emplace_after
 - std::forward_list, 2460
- emplace_front
 - std::forward_list, 2460
- empty
 - __gnu_cxx::__versa_string, 821
 - __gnu_debug::basic_string, 1072
 - std::basic_string, 2068
 - std::deque, 2404
 - std::forward_list, 2460
 - std::list, 2615
 - std::map, 2664
 - std::match_results, 2685
 - std::multimap, 2755
 - std::multiset, 2775
 - std::priority_queue, 2926
 - std::queue, 2932
 - std::set, 2985
 - std::stack, 3025
 - std::vector, 3161
- enc_filebuf.h, 3322
- end
 - __gnu_cxx::__versa_string, 822
 - __gnu_cxx::temporary_buffer, 1013

- __gnu_debug::basic_string, 1072, 1073
- __gnu_parallel::_PseudoSequence, 1207
- numeric_arrays, 93
- std, 626, 627
- std::_Temporary_buffer, 1479
- std::basic_fstream, 1617
- std::basic_ifstream, 1670
- std::basic_ios, 1701
- std::basic_iostream, 1766
- std::basic_istream, 1817
- std::basic_istreamstream, 1872
- std::basic_ofstream, 1916
- std::basic_ostream, 1958
- std::basic_ostreamstream, 2004
- std::basic_string, 2068
- std::basic_stringstream, 2190
- std::deque, 2404
- std::forward_list, 2461
- std::ios_base, 2565
- std::list, 2615
- std::map, 2664
- std::match_results, 2685
- std::multimap, 2756
- std::multiset, 2775
- std::set, 2986
- std::vector, 3161
- endl
 - std, 627
- ends
 - std, 628
- eof
 - std::basic_fstream, 1573
 - std::basic_ifstream, 1636
 - std::basic_ios, 1686
 - std::basic_iostream, 1722
 - std::basic_istream, 1783
 - std::basic_istreamstream, 1837
 - std::basic_ofstream, 1891
 - std::basic_ostream, 1933
 - std::basic_ostreamstream, 1978
 - std::basic_stringstream, 2146
- eofbit
 - std::basic_fstream, 1617
 - std::basic_ifstream, 1671
 - std::basic_ios, 1701
 - std::basic_iostream, 1766
 - std::basic_istream, 1817
 - std::basic_istreamstream, 1872
 - std::basic_ofstream, 1916
 - std::basic_ostream, 1958
 - std::basic_ostreamstream, 2004
 - std::basic_stringstream, 2190
 - std::ios_base, 2565
- epptr
 - __gnu_cxx::enc_filebuf, 891
 - __gnu_cxx::stdio_filebuf, 961
 - __gnu_cxx::stdio_sync_filebuf, 991
 - std::basic_filebuf, 1531
 - std::basic_streambuf, 2026
 - std::basic_stringbuf, 2109
- epsilon
 - std::numeric_limits, 2839
- equal
 - non_mutating_algorithms, 156, 157
 - std::istreambuf_iterator, 2588
- equal_range
 - binary_search_algorithms, 199
 - std::map, 2664, 2665
 - std::multimap, 2756, 2757
 - std::multiset, 2775, 2776
 - std::set, 2986
- equally_split
 - __gnu_parallel, 440
- equally_split.h, 3323
- equally_split_point
 - __gnu_parallel, 441
- erase
 - __gnu_cxx::__versa_string, 822, 823
 - __gnu_debug::basic_string, 1073, 1074
 - std::basic_string, 2068, 2069
 - std::deque, 2404, 2405
 - std::list, 2615, 2616
 - std::map, 2666
 - std::multimap, 2757, 2758
 - std::multiset, 2776, 2777
 - std::set, 2987, 2988
 - std::vector, 3161, 3162
- erase_after

-
- std::forward_list, [2461](#), [2462](#)
 - error_backref
 - std::regex_constants, [753](#)
 - error_badbrace
 - std::regex_constants, [753](#)
 - error_badrepeat
 - std::regex_constants, [753](#)
 - error_brace
 - std::regex_constants, [753](#)
 - error_brack
 - std::regex_constants, [753](#)
 - error_collate
 - std::regex_constants, [753](#)
 - error_complexity
 - std::regex_constants, [753](#)
 - error_constants.h, [3323](#)
 - error_ctype
 - std::regex_constants, [753](#)
 - error_escape
 - std::regex_constants, [754](#)
 - error_paren
 - std::regex_constants, [754](#)
 - error_range
 - std::regex_constants, [754](#)
 - error_space
 - std::regex_constants, [754](#)
 - error_stack
 - std::regex_constants, [754](#)
 - error_type
 - std::regex_constants, [752](#)
 - event
 - std::basic_fstream, [1569](#)
 - std::basic_ifstream, [1632](#)
 - std::basic_ios, [1684](#)
 - std::basic_iostream, [1719](#)
 - std::basic_istream, [1781](#)
 - std::basic_istreamstream, [1834](#)
 - std::basic_ofstream, [1887](#)
 - std::basic_ostream, [1931](#)
 - std::basic_ostreamstream, [1975](#)
 - std::basic_stringstream, [2143](#)
 - std::ios_base, [2557](#)
 - event_callback
 - std::basic_fstream, [1564](#)
 - std::basic_ifstream, [1629](#)
 - std::basic_ios, [1680](#)
 - std::basic_iostream, [1715](#)
 - std::basic_istream, [1777](#)
 - std::basic_istreamstream, [1831](#)
 - std::basic_ofstream, [1883](#)
 - std::basic_ostream, [1927](#)
 - std::basic_ostreamstream, [1971](#)
 - std::basic_stringstream, [2138](#)
 - std::ios_base, [2554](#)
 - exception, [3324](#)
 - exception.hpp, [3325](#)
 - exception_ptr.h, [3326](#)
 - Exceptions, [33](#)
 - exceptions
 - __verbose_terminate_handler, [36](#)
 - copy_exception, [36](#)
 - current_exception, [36](#)
 - make_exception_ptr, [36](#)
 - rethrow_exception, [36](#)
 - rethrow_if_nested, [37](#)
 - set_terminate, [37](#)
 - set_unexpected, [37](#)
 - std::basic_fstream, [1573](#)
 - std::basic_ifstream, [1636](#)
 - std::basic_ios, [1686](#), [1687](#)
 - std::basic_iostream, [1722](#), [1723](#)
 - std::basic_istream, [1783](#), [1784](#)
 - std::basic_istreamstream, [1838](#)
 - std::basic_ofstream, [1891](#)
 - std::basic_ostream, [1934](#)
 - std::basic_ostreamstream, [1979](#)
 - std::basic_stringstream, [2147](#)
 - terminate, [37](#)
 - terminate_handler, [35](#)
 - throw_with_nested, [37](#)
 - uncaught_exception, [37](#)
 - unexpected, [38](#)
 - unexpected_handler, [35](#)
 - exp
 - complex_numbers, [46](#)
 - expint
 - tr1_math_spec_func, [116](#)
 - exponential_distribution
 - std::exponential_distribution, [2438](#)
 - extended
 - std::regex_constants, [755](#)
 - Extensions, [11](#)
-

- extptr_allocator.h, [3327](#)
- fabs
 - complex_numbers, [46](#)
- facet
 - std::locale::facet, [2640](#)
- fail
 - std::basic_fstream, [1574](#)
 - std::basic_ifstream, [1637](#)
 - std::basic_ios, [1687](#)
 - std::basic_iostream, [1723](#)
 - std::basic_istream, [1784](#)
 - std::basic_istreamstream, [1839](#)
 - std::basic_ofstream, [1892](#)
 - std::basic_ostream, [1935](#)
 - std::basic_ostreamstream, [1980](#)
 - std::basic_stringstream, [2148](#)
- failbit
 - std::basic_fstream, [1618](#)
 - std::basic_ifstream, [1671](#)
 - std::basic_ios, [1702](#)
 - std::basic_iostream, [1766](#)
 - std::basic_istream, [1817](#)
 - std::basic_istreamstream, [1873](#)
 - std::basic_ofstream, [1917](#)
 - std::basic_ostream, [1959](#)
 - std::basic_ostreamstream, [2004](#)
 - std::basic_stringstream, [2191](#)
 - std::ios_base, [2566](#)
- failed
 - std::ostreambuf_iterator, [2892](#)
- false_type
 - metaprogramming, [124](#)
- falsename
 - std::numpunct, [2875](#)
 - std::numpunct_byname, [2881](#)
- fd
 - __gnu_cxx::stdio_filebuf, [961](#)
- features.h, [3327](#)
 - _GLIBCXX_BAL_QUICKSORT, [3328](#)
 - _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS, [3328](#)
 - _GLIBCXX_FIND_EQUAL_SPLIT, [3328](#)
 - _GLIBCXX_FIND_GROWING_BLOCKS, [3328](#)
 - _GLIBCXX_MERGESORT, [3329](#)
 - _GLIBCXX_QUICKSORT, [3329](#)
 - _GLIBCXX_TREE_DYNAMIC_BALANCING, [3329](#)
 - _GLIBCXX_TREE_FULL_COPY, [3329](#)
 - _GLIBCXX_TREE_INITIAL_SPLITTING, [3330](#)
- fenv.h, [3330](#)
- file
 - __gnu_cxx::stdio_filebuf, [961](#)
 - __gnu_cxx::stdio_sync_filebuf, [991](#)
- filebuf
 - io, [61](#)
- fill
 - mutating_algorithms, [134](#)
 - std::basic_fstream, [1575](#)
 - std::basic_ifstream, [1637](#), [1638](#)
 - std::basic_ios, [1688](#)
 - std::basic_iostream, [1724](#)
 - std::basic_istream, [1785](#)
 - std::basic_istreamstream, [1839](#), [1840](#)
 - std::basic_ofstream, [1892](#), [1893](#)
 - std::basic_ostream, [1935](#), [1936](#)
 - std::basic_ostreamstream, [1980](#)
 - std::basic_stringstream, [2148](#), [2149](#)
- fill_minimal_n
 - __gnu_parallel::_Settings, [1219](#)
- fill_n
 - mutating_algorithms, [134](#)
- find
 - __gnu_cxx::__versa_string, [824](#), [825](#)
 - __gnu_debug::basic_string, [1074](#), [1075](#)
 - non_mutating_algorithms, [157](#)
 - std::basic_string, [2070](#), [2071](#)
 - std::map, [2667](#)
 - std::multimap, [2759](#)
 - std::multiset, [2778](#)
 - std::set, [2988](#), [2989](#)
- find.h, [3330](#)
- find_end
 - non_mutating_algorithms, [158](#), [159](#)

-
- find_first_not_of
 - __gnu_cxx::__versa_string, 826–828
 - __gnu_debug::basic_string, 1076, 1077
 - std::basic_string, 2072, 2073
 - find_first_of
 - __gnu_cxx::__versa_string, 828–830
 - __gnu_debug::basic_string, 1077–1079
 - non_mutating_algorithms, 159, 160
 - std::basic_string, 2074, 2075
 - find_if
 - non_mutating_algorithms, 161
 - find_if_not
 - non_mutating_algorithms, 161
 - find_increasing_factor
 - __gnu_parallel::_Settings, 1219
 - find_initial_block_size
 - __gnu_parallel::_Settings, 1219
 - find_last_not_of
 - __gnu_cxx::__versa_string, 830–832
 - __gnu_debug::basic_string, 1079, 1080
 - std::basic_string, 2076, 2077
 - find_last_of
 - __gnu_cxx::__versa_string, 832–834
 - __gnu_debug::basic_string, 1081, 1082
 - std::basic_string, 2078, 2079
 - find_maximum_block_size
 - __gnu_parallel::_Settings, 1219
 - find_scale_factor
 - __gnu_parallel::_Settings, 1220
 - find_selectors.h, 3331
 - find_sequential_search_size
 - __gnu_parallel::_Settings, 1220
 - first
 - __gnu_parallel::_IteratorPair, 1162
 - std::pair, 2902
 - std::sub_match, 3035
 - first_argument_type
 - __gnu_cxx::project1st, 931
 - __gnu_cxx::project2nd, 932
 - __gnu_parallel::_EqualFromLess, 1155
 - __gnu_parallel::_EqualTo, 1157
 - __gnu_parallel::_Less, 1167
 - __gnu_parallel::_Lexicographic, 1169
 - __gnu_parallel::_-
 - LexicographicReverse, 1171
 - __gnu_parallel::_Multiplies, 1199
 - __gnu_parallel::_Plus, 1203
 - std::_Maybe_unary_or_binary_-
 - function< _Res, _T1, _T2 >, 1465
 - std::binary_function, 2200
 - std::binary_negate, 2201
 - std::const_mem_fun1_ref_t, 2291
 - std::const_mem_fun1_t, 2293
 - std::divides, 2427
 - std::equal_to, 2432
 - std::greater, 2508
 - std::greater_equal, 2510
 - std::less, 2595
 - std::less_equal, 2597
 - std::logical_and, 2646
 - std::logical_or, 2650
 - std::mem_fun1_ref_t, 2691
 - std::mem_fun1_t, 2693
 - std::minus, 2707
 - std::modulus, 2709
 - std::multiplies, 2768
 - std::not_equal_to, 2802
 - std::owner_less< shared_ptr< _Tp > >, 2896
 - std::owner_less< weak_ptr< _Tp > >, 2898
 - std::plus, 2914
 - std::pointer_to_binary_function, 2916
 - first_type
 - __gnu_parallel::_IteratorPair, 1162
 - std::pair, 2901
 - std::sub_match, 3033
 - fixed
 - std, 628
 - std::basic_fstream, 1618
-

- std::basic_ifstream, 1671
- std::basic_ios, 1702
- std::basic_iostream, 1767
- std::basic_istream, 1818
- std::basic_istream, 1873
- std::basic_ofstream, 1917
- std::basic_ostream, 1959
- std::basic_ostringstream, 2004
- std::basic_stringstream, 2191
- std::ios_base, 2566
- flags
 - std::basic_fstream, 1575, 1576
 - std::basic_ifstream, 1638
 - std::basic_ios, 1688, 1689
 - std::basic_iostream, 1725
 - std::basic_istream, 1786
 - std::basic_istream, 1840
 - std::basic_ofstream, 1893
 - std::basic_ostream, 1936
 - std::basic_ostringstream, 1981
 - std::basic_regex, 2016
 - std::basic_stringstream, 2149
 - std::ios_base, 2557, 2558
- flip
 - std::bitset, 2217, 2218
- float_denorm_style
 - std, 605
- float_round_style
 - std, 606
- floatfield
 - std::basic_fstream, 1618
 - std::basic_ifstream, 1671
 - std::basic_ios, 1702
 - std::basic_iostream, 1767
 - std::basic_istream, 1818
 - std::basic_istream, 1873
 - std::basic_ofstream, 1917
 - std::basic_ostream, 1959
 - std::basic_ostringstream, 2004
 - std::basic_stringstream, 2191
 - std::ios_base, 2566
- flush
 - std, 628
 - std::basic_fstream, 1576
 - std::basic_iostream, 1725
 - std::basic_ofstream, 1894
 - std::basic_ostream, 1937
 - std::basic_ostringstream, 1982
 - std::basic_stringstream, 2150
- fmtflags
 - std::basic_fstream, 1565
 - std::basic_ifstream, 1629
 - std::basic_ios, 1680
 - std::basic_iostream, 1715
 - std::basic_istream, 1778
 - std::basic_istream, 1831
 - std::basic_ofstream, 1884
 - std::basic_ostream, 1927
 - std::basic_ostringstream, 1972
 - std::basic_stringstream, 2139
 - std::ios_base, 2554
- for_each
 - non_mutating_algorithms, 161
- for_each.h, 3332
- for_each_minimal_n
 - __gnu_parallel::Settings, 1220
- for_each_selectors.h, 3333
- format
 - std::match_results, 2685, 2686
- format_default
 - std::regex_constants, 755
- format_first_only
 - std::regex_constants, 756
- format_no_copy
 - std::regex_constants, 756
- format_sed
 - std::regex_constants, 756
- formatter.h, 3335
- forward
 - std, 628, 629
- forward_list
 - std::forward_list, 2454–2456
- forward_list.h, 3336
- forward_list.tcc, 3337
- fpos
 - std::fpos, 2473
- frac_digits
 - std::moneypunct, 2731
 - std::moneypunct_byname, 2743
- front
 - __gnu_cxx::__versa_string, 835

- __gnu_debug::basic_string, 1082, 1083
 - std::basic_string, 2079, 2080
 - std::deque, 2405, 2406
 - std::forward_list, 2462
 - std::list, 2616, 2617
 - std::queue, 2932
 - std::vector, 3162, 3163
- front_insert_iterator
 - std::front_insert_iterator, 2477
- front_inserter
 - iterators, 274
- fstream, 3338
 - io, 61
- fstream.tcc, 3339
- functexcept.h, 3339
- function
 - std::function< _Res(_- ArgTypes...)>, 2481, 2482
- Function Objects, 254
- functional, 3340, 3344
- functional_hash.h, 3345
- functions.h, 3346
- functors
 - mem_fn, 256
- future, 3349
 - std::future, 2488
 - std::future< _Res & >, 2491
 - std::future< void >, 2494
- future_category
 - futures, 57
- future_errc
 - futures, 57
- Futures, 55
- futures
 - future_category, 57
 - future_errc, 57
- gamma_distribution
 - std::gamma_distribution, 2498
- gbump
 - __gnu_cxx::enc_filebuf, 891
 - __gnu_cxx::stdio_filebuf, 962
 - __gnu_cxx::stdio_sync_filebuf, 991
 - std::basic_filebuf, 1532
 - std::basic_streambuf, 2027
 - std::basic_stringbuf, 2109
- gcount
 - std::basic_fstream, 1576
 - std::basic_ifstream, 1639
 - std::basic_iostream, 1726
 - std::basic_istream, 1786
 - std::basic_istreamstream, 1841
 - std::basic_stringstream, 2150
- generate
 - mutating_algorithms, 135
- generate_canonical
 - random, 217
- generate_minimal_n
 - __gnu_parallel::_Settings, 1220
- generate_n
 - mutating_algorithms, 135
- get
 - __gnu_parallel::_Settings, 1218
 - std::auto_ptr, 1511
 - std::basic_fstream, 1577–1580
 - std::basic_ifstream, 1639–1642
 - std::basic_iostream, 1726–1729
 - std::basic_istream, 1787–1789
 - std::basic_istreamstream, 1841–1844
 - std::basic_stringstream, 2150–2153
 - std::future, 2489
 - std::future< _Res & >, 2492
 - std::future< void >, 2495
 - std::money_get, 2715, 2716
 - std::num_get, 2813–2821
 - std::shared_future, 2998
 - std::shared_future< _Res & >, 3001
- get_allocator
 - __gnu_cxx::__versa_string, 835
 - __gnu_debug::basic_string, 1083
 - std::basic_string, 2080
 - std::deque, 2406
 - std::forward_list, 2462
 - std::list, 2617
 - std::map, 2668
 - std::match_results, 2686
 - std::multimap, 2759
 - std::multiset, 2779
 - std::set, 2989
- get_date
 - std::time_get, 3047

- std::time_get_byname, 3058
- get_deleter
 - pointer_abstractions, 69
- get_id
 - std::this_thread, 762
- get_money
 - std, 629
- get_monthname
 - std::time_get, 3048
 - std::time_get_byname, 3059
- get_temporary_buffer
 - std, 629
- get_time
 - std::time_get, 3049
 - std::time_get_byname, 3060
- get_weekday
 - std::time_get, 3049
 - std::time_get_byname, 3060
- get_year
 - std::time_get, 3050
 - std::time_get_byname, 3061
- getline
 - std, 630, 631
 - std::basic_fstream, 1580
 - std::basic_ifstream, 1642, 1643
 - std::basic_iostream, 1729, 1730
 - std::basic_istream, 1790, 1791
 - std::basic_istreamstream, 1844, 1845
 - std::basic_stringstream, 2154, 2155
- getloc
 - __gnu_cxx::enc_filebuf, 891
 - __gnu_cxx::stdio_filebuf, 962
 - __gnu_cxx::stdio_sync_filebuf, 992
 - std::basic_filebuf, 1532
 - std::basic_fstream, 1581
 - std::basic_ifstream, 1644
 - std::basic_ios, 1689
 - std::basic_iostream, 1731
 - std::basic_istream, 1791
 - std::basic_istreamstream, 1846
 - std::basic_ofstream, 1894
 - std::basic_ostream, 1937
 - std::basic_ostreamstream, 1982
 - std::basic_regex, 2016
 - std::basic_streambuf, 2027
 - std::basic_stringbuf, 2110
 - std::basic_stringstream, 2155
- std::basic_stringstream, 2155
- std::ios_base, 2558
- std::regex_traits, 2960
- global
 - std::locale, 2633
- good
 - std::basic_fstream, 1582
 - std::basic_ifstream, 1644
 - std::basic_ios, 1689
 - std::basic_iostream, 1731
 - std::basic_istream, 1791
 - std::basic_istreamstream, 1846
 - std::basic_ofstream, 1894
 - std::basic_ostream, 1937
 - std::basic_ostreamstream, 1982
 - std::basic_stringstream, 2155
- goodbit
 - std::basic_fstream, 1618
 - std::basic_ifstream, 1671
 - std::basic_ios, 1702
 - std::basic_iostream, 1767
 - std::basic_istream, 1818
 - std::basic_istreamstream, 1873
 - std::basic_ofstream, 1917
 - std::basic_ostream, 1959
 - std::basic_ostreamstream, 2005
 - std::basic_stringstream, 2191
 - std::ios_base, 2566
- gptr
 - __gnu_cxx::enc_filebuf, 892
 - __gnu_cxx::stdio_filebuf, 962
 - __gnu_cxx::stdio_sync_filebuf, 992
 - std::basic_filebuf, 1532
 - std::basic_streambuf, 2027
 - std::basic_stringbuf, 2110
- grep
 - std::regex_constants, 757
- grouping
 - std::moneypunct, 2731
 - std::moneypunct_byname, 2743
 - std::numpunct, 2875
 - std::numpunct_byname, 2881
- gslice
 - numeric_arrays, 88
- gslice.h, 3351
- gslice_array

-
- numeric_arrays, 88
 - gslice_array.h, 3352
 - has_denorm
 - std::__numeric_limits_base, 1410
 - std::numeric_limits, 2841
 - has_denorm_loss
 - std::__numeric_limits_base, 1410
 - std::numeric_limits, 2841
 - has_facet
 - std, 632
 - std::locale, 2635
 - std::locale::id, 2641
 - has_infinity
 - std::__numeric_limits_base, 1410
 - std::numeric_limits, 2841
 - has_quiet_NaN
 - std::__numeric_limits_base, 1411
 - std::numeric_limits, 2841
 - has_signaling_NaN
 - std::__numeric_limits_base, 1411
 - std::numeric_limits, 2841
 - hash
 - std::collate, 2277
 - std::collate_byname, 2283
 - hash_fun.h, 3352
 - hash_map, 3353
 - hash_policy.hpp, 3354
 - hash_set, 3354
 - Hashes, 213
 - hashtable.h, 3355, 3356
 - hashtable_policy.h, 3357
 - Heap, 263
 - heap_algorithms
 - is_heap, 265
 - is_heap_until, 265, 266
 - make_heap, 266, 267
 - pop_heap, 267
 - push_heap, 268
 - sort_heap, 269
 - hermite
 - tr1_math_spec_func, 116
 - hex
 - std, 632
 - std::basic_fstream, 1619
 - std::basic_ifstream, 1672
 - std::basic_ios, 1703
 - std::basic_iostream, 1767
 - std::basic_istream, 1818
 - std::basic_istreamstream, 1874
 - std::basic_ofstream, 1918
 - std::basic_ostream, 1960
 - std::basic_ostreamstream, 2005
 - std::basic_stringstream, 2192
 - std::ios_base, 2567
 - hours
 - std::chrono, 737
 - hyperg
 - tr1_math_spec_func, 116
 - I/O, 57
 - icase
 - std::regex_constants, 757
 - id
 - std::collate, 2278
 - std::collate_byname, 2284
 - std::ctype, 2312
 - std::ctype< char >, 2326
 - std::ctype< wchar_t >, 2347
 - std::ctype_byname, 2363
 - std::ctype_byname< char >, 2376
 - std::locale::id, 2641
 - std::messages, 2701
 - std::messages_byname, 2705
 - std::money_get, 2716
 - std::money_put, 2722
 - std::moneypunct, 2735
 - std::moneypunct_byname, 2747
 - std::num_get, 2822
 - std::num_put, 2837
 - std::numpunct, 2876
 - std::numpunct_byname, 2883
 - std::time_get, 3051
 - std::time_get_byname, 3062
 - std::time_put, 3067
 - std::time_put_byname, 3072
 - identity_element
 - SGIextensions, 20, 21
 - ifstream
 - io, 61
 - ignore
 - std::basic_fstream, 1582, 1583
-

- std::basic_ifstream, 1644–1646
- std::basic_iostream, 1731–1733
- std::basic_istream, 1792, 1793
- std::basic_istreamstream, 1846, 1847
- std::basic_stringstream, 2156, 2157
- imbue
 - __gnu_cxx::enc_filebuf, 892
 - __gnu_cxx::stdio_filebuf, 963
 - __gnu_cxx::stdio_sync_filebuf, 992
 - std::basic_filebuf, 1533
 - std::basic_fstream, 1584
 - std::basic_ifstream, 1646
 - std::basic_ios, 1690
 - std::basic_iostream, 1733
 - std::basic_istream, 1793
 - std::basic_istreamstream, 1848
 - std::basic_ofstream, 1895
 - std::basic_ostream, 1938
 - std::basic_ostreamstream, 1983
 - std::basic_regex, 2017
 - std::basic_streambuf, 2028
 - std::basic_stringbuf, 2110
 - std::basic_stringstream, 2157
 - std::ios_base, 2558
 - std::regex_traits, 2960
- in
 - std::__codecvt_abstract_base, 1318
 - std::basic_fstream, 1619
 - std::basic_ifstream, 1672
 - std::basic_ios, 1703
 - std::basic_iostream, 1768
 - std::basic_istream, 1819
 - std::basic_istreamstream, 1874
 - std::basic_ofstream, 1918
 - std::basic_ostream, 1960
 - std::basic_ostreamstream, 2005
 - std::basic_stringstream, 2192
 - std::codecvt, 2245
 - std::codecvt< _InternT, _ExternT, encoding_state >, 2251
 - std::codecvt< char, char, mbstate_t >, 2256
 - std::codecvt< wchar_t, char, mbstate_t >, 2261
 - std::codecvt_byname, 2268
 - std::ios_base, 2567
- in_avail
 - __gnu_cxx::enc_filebuf, 892
 - __gnu_cxx::stdio_filebuf, 963
 - __gnu_cxx::stdio_sync_filebuf, 993
 - std::basic_filebuf, 1533
 - std::basic_streambuf, 2028
 - std::basic_stringbuf, 2111
- include/ Directory Reference, 328
- include/backward/ Directory Reference, 312
- include/bits/ Directory Reference, 315
- include/debug/ Directory Reference, 318
- include/decimal/ Directory Reference, 319
- include/ext/ Directory Reference, 323
- include/ext/pb_ds/ Directory Reference, 336
- include/ext/pb_ds/detail/ Directory Reference, 320
- include/parallel/ Directory Reference, 333
- include/profile/ Directory Reference, 338
- include/profile/impl/ Directory Reference, 326
- include/tr1/ Directory Reference, 340
- include/tr1/impl/ Directory Reference, 341
- include/x86_64-unknown-linux-gnu/ Directory Reference, 342
- include/x86_64-unknown-linux-gnu/bits/ Directory Reference, 313
- includes
 - set_algorithms, 190
- increment
 - std::linear_congruential_engine, 2603
- independent_bits_engine
 - std::independent_bits_engine, 2536, 2537
- indirect_array
 - numeric_arrays, 89
- indirect_array.h, 3358
- infinity
 - std::numeric_limits, 2839
- init
 - std::basic_fstream, 1584
 - std::basic_ifstream, 1647

-
- std::basic_ios, 1690
 - std::basic_iostream, 1734
 - std::basic_istream, 1794
 - std::basic_istreamstream, 1848
 - std::basic_ofstream, 1895
 - std::basic_ostream, 1938
 - std::basic_ostringstream, 1983
 - std::basic_stringstream, 2158
 - initializer_list, 3358
 - inner_product
 - std, 633
 - inplace_merge
 - sorting_algorithms, 170
 - insert
 - __gnu_cxx::__versa_string, 835–840
 - __gnu_debug::basic_string, 1083–1087
 - std::basic_string, 2080–2084
 - std::deque, 2406–2408
 - std::list, 2617–2619
 - std::map, 2668–2670
 - std::multimap, 2760, 2761
 - std::multiset, 2779, 2780
 - std::set, 2989–2991
 - std::vector, 3163–3165
 - insert_after
 - std::forward_list, 2463, 2464
 - insert_iterator
 - std::insert_iterator, 2548
 - inserter
 - iterators, 275
 - int_type
 - __gnu_cxx::enc_filebuf, 888
 - __gnu_cxx::stdio_filebuf, 956
 - __gnu_cxx::stdio_sync_filebuf, 989
 - std::basic_filebuf, 1528
 - std::basic_fstream, 1565, 1566
 - std::basic_ifstream, 1630
 - std::basic_ios, 1681
 - std::basic_iostream, 1716
 - std::basic_istream, 1778
 - std::basic_istreamstream, 1832
 - std::basic_ofstream, 1885
 - std::basic_ostream, 1928
 - std::basic_ostringstream, 1973
 - std::basic_streambuf, 2023
 - std::basic_stringbuf, 2106
 - std::basic_stringstream, 2139, 2140
 - std::istreambuf_iterator, 2586
 - internal
 - std, 634
 - std::basic_fstream, 1619
 - std::basic_ifstream, 1672
 - std::basic_ios, 1703
 - std::basic_iostream, 1768
 - std::basic_istream, 1819
 - std::basic_istreamstream, 1874
 - std::basic_ofstream, 1918
 - std::basic_ostream, 1960
 - std::basic_ostringstream, 2005
 - std::basic_stringstream, 2192
 - std::ios_base, 2567
 - intervals
 - std::piecewise_constant_distribution, 2904
 - std::piecewise_linear_distribution, 2909
 - intl
 - std::moneypunct, 2735
 - std::moneypunct_byname, 2747
 - io
 - filebuf, 61
 - fstream, 61
 - ifstream, 61
 - ios, 61
 - iostream, 61
 - istream, 61
 - istreamstream, 61
 - ofstream, 61
 - ostream, 62
 - ostringstream, 62
 - streambuf, 62
 - stringstream, 62
 - wfilebuf, 62
 - wfstream, 62
 - wifstream, 63
 - wios, 63
 - wiostream, 63
 - wistream, 63
 - wstringstream, 63
-

- wofstream, 63
- wostream, 63
- wostringstream, 64
- wstreambuf, 64
- wstringbuf, 64
- wstringstream, 64
- iomanip, 3359
- ios, 3360
 - io, 61
- ios_base.h, 3361
- iosfwd, 3363
- iostate
 - std::basic_fstream, 1566
 - std::basic_ifstream, 1630
 - std::basic_ios, 1681
 - std::basic_iostream, 1716
 - std::basic_istream, 1779
 - std::basic_istream, 1832
 - std::basic_ofstream, 1885
 - std::basic_ostream, 1928
 - std::basic_ostringstream, 1973
 - std::basic_stringstream, 2140
 - std::ios_base, 2555
- iostream, 3364
 - io, 61
- iota
 - SGIextensions, 21
 - std, 634
- is
 - std::__ctype_abstract_base, 1330
 - std::ctype, 2306, 2307
 - std::ctype< char >, 2320, 2321
 - std::ctype< wchar_t >, 2342
 - std::ctype_byname, 2357, 2358
 - std::ctype_byname< char >, 2370, 2371
- is_bounded
 - std::__numeric_limits_base, 1411
 - std::numeric_limits, 2841
- is_exact
 - std::__numeric_limits_base, 1411
 - std::numeric_limits, 2842
- is_heap
 - heap_algorithms, 265
 - SGIextensions, 21
- is_heap_until
 - heap_algorithms, 265, 266
- is_iec559
 - std::__numeric_limits_base, 1411
 - std::numeric_limits, 2842
- is_integer
 - std::__numeric_limits_base, 1411
 - std::numeric_limits, 2842
- is_modulo
 - std::__numeric_limits_base, 1412
 - std::numeric_limits, 2842
- is_open
 - __gnu_cxx::enc_filebuf, 893
 - __gnu_cxx::stdio_filebuf, 964
 - std::basic_filebuf, 1534
 - std::basic_fstream, 1584
 - std::basic_ifstream, 1647
 - std::basic_ofstream, 1895
- is_partitioned
 - mutating_algorithms, 136
- is_signed
 - std::__numeric_limits_base, 1412
 - std::numeric_limits, 2842
- is_sorted
 - SGIextensions, 22
 - sorting_algorithms, 171
- is_sorted_until
 - sorting_algorithms, 172
- is_specialized
 - std::__numeric_limits_base, 1412
 - std::numeric_limits, 2843
- isalnum
 - std, 634
- isalpha
 - std, 634
- isctrl
 - std, 635
- isctype
 - regex, 225
- isdigit
 - std, 635
- isgraph
 - std, 635
- islower
 - std, 635
- isprint
 - std, 635

- ispunct
 - std, 635
- isspace
 - std, 635
- istream, 3365
 - io, 61
- istream.tcc, 3366
- istream_iterator
 - std::istream_iterator, 2584
- istream_type
 - std::istreambuf_iterator, 2586
- istreambuf_iterator
 - std::istreambuf_iterator, 2587, 2588
- istringstream
 - io, 61
- isupper
 - std, 635
- isxdigit
 - std, 636
- iter_swap
 - mutating_algorithms, 136
- iter_type
 - std::money_get, 2713
 - std::money_put, 2719
 - std::num_get, 2805
 - std::num_put, 2825
 - std::time_get, 3043
 - std::time_get_byname, 3054
 - std::time_put, 3064
 - std::time_put_byname, 3069
- iterator, 3367
 - std::set, 2979
- Iterator Tags, 276
- iterator.h, 3367
- iterator_category
 - std::back_insert_iterator, 1516
 - std::front_insert_iterator, 2476
 - std::insert_iterator, 2547
 - std::istream_iterator, 2583
 - std::istreambuf_iterator, 2586
 - std::iterator, 2590
 - std::ostream_iterator, 2886
 - std::ostreambuf_iterator, 2890
 - std::raw_storage_iterator, 2943
 - std::reverse_iterator, 2967
- iterator_tracker.h, 3368
- Iterators, 270
- iterators
 - __iterator_category, 274
 - back_inserter, 274
 - front_inserter, 274
 - inserter, 275
 - operator==, 275, 276
- iword
 - std::basic_fstream, 1585
 - std::basic_ifstream, 1647
 - std::basic_ios, 1690
 - std::basic_iostream, 1734
 - std::basic_istream, 1794
 - std::basic_istreamstream, 1849
 - std::basic_ofstream, 1896
 - std::basic_ostream, 1938
 - std::basic_ostringstream, 1983
 - std::basic_stringstream, 2158
 - std::ios_base, 2559
- k
 - std::negative_binomial_distribution, 2790
- key_comp
 - std::map, 2670
 - std::multimap, 2761
 - std::multiset, 2781
 - std::set, 2991
- key_compare
 - std::set, 2980
- key_type
 - std::set, 2980
- kill_dependency
 - atomics, 212
- L1_cache_size
 - __gnu_parallel::_Settings, 1220
- L2_cache_size
 - __gnu_parallel::_Settings, 1220
- laguerre
 - tr1_math_spec_func, 116
- lambda
 - std::exponential_distribution, 2438
- left
 - std, 636
 - std::basic_fstream, 1619

- std::basic_ifstream, 1672
- std::basic_ios, 1703
- std::basic_iostream, 1768
- std::basic_istream, 1819
- std::basic_istream, 1874
- std::basic_ofstream, 1918
- std::basic_ostream, 1960
- std::basic_ostringstream, 2006
- std::basic_stringstream, 2192
- std::ios_base, 2567
- legendre
 - tr1_math_spec_func, 116
- length
 - __gnu_cxx::__versa_string, 840
 - __gnu_debug::basic_string, 1088
 - std::basic_string, 2085
 - std::match_results, 2686
 - std::regex_traits, 2961
 - std::sub_match, 3034
- lexicographical_compare
 - sorting_algorithms, 173
- lexicographical_compare_3way
 - SGIextensions, 22
- limits, 3370
- linear_congruential_engine
 - std::linear_congruential_engine, 2599
- list, 3372–3374
 - std::list, 2608–2610
- list.tcc, 3375
- list_partition
 - __gnu_parallel, 441
- list_partition.h, 3375
- list_update_policy.hpp, 3376
- locale, 3376
 - std::locale, 2631, 2632
- locale_classes.h, 3376
- locale_classes.tcc, 3377
- locale_facets.h, 3378
- locale_facets.tcc, 3381
- locale_facets_nonio.h, 3381
- locale_facets_nonio.tcc, 3383
- localefd.h, 3383
- Locales, 213
- lock
 - mutexes, 71
- log
 - complex_numbers, 46
- log10
 - complex_numbers, 46
- logic_error
 - std::logic_error, 2644
- lookup_classname
 - std::regex_traits, 2961
- lookup_collatename
 - std::regex_traits, 2962
- losertree.h, 3384
- lower_bound
 - binary_search_algorithms, 200
 - std::map, 2670, 2671
 - std::multimap, 2762
 - std::multiset, 2781
 - std::set, 2991, 2992
- lowest
 - std::numeric_limits, 2839
- macros.h, 3386
 - _GLIBCXX_DEBUG_VERIFY, 3389
 - __glibcxx_check_erase, 3387
 - __glibcxx_check_erase_after, 3387
 - __glibcxx_check_erase_range, 3387
 - __glibcxx_check_erase_range_after, 3387
 - __glibcxx_check_heap_pred, 3387
 - __glibcxx_check_insert, 3387
 - __glibcxx_check_insert_after, 3388
 - __glibcxx_check_insert_range, 3388
 - __glibcxx_check_insert_range_-after, 3388
 - __glibcxx_check_partitioned_lower, 3388
 - __glibcxx_check_partitioned_-lower_pred, 3389
 - __glibcxx_check_partitioned_-upper_pred, 3389
 - __glibcxx_check_sorted_pred, 3389
- make_exception_ptr
 - exceptions, 36
- make_heap
 - heap_algorithms, 266, 267
- make_pair

-
- std, 636
 - make_shared
 - pointer_abstractions, 69
 - malloc_allocator.h, 3390
 - map, 3390, 3391
 - std::map, 2659–2661
 - map.h, 3391, 3392
 - mark_count
 - std::basic_regex, 2017
 - mask_array
 - numeric_arrays, 89
 - mask_array.h, 3393
 - match_any
 - std::regex_constants, 757
 - match_continuous
 - std::regex_constants, 757
 - match_default
 - std::regex_constants, 757
 - match_flag_type
 - std::regex_constants, 751
 - match_not_bol
 - std::regex_constants, 757
 - match_not_bow
 - std::regex_constants, 758
 - match_not_eol
 - std::regex_constants, 758
 - match_not_eow
 - std::regex_constants, 758
 - match_not_null
 - std::regex_constants, 758
 - match_prev_avail
 - std::regex_constants, 758
 - match_results
 - std::match_results, 2683
 - Mathematical Special Functions, 110
 - max
 - __gnu_parallel, 442
 - numeric_arrays, 93
 - sorting_algorithms, 174
 - std::bernoulli_distribution, 2196
 - std::binomial_distribution, 2207
 - std::cauchy_distribution, 2226
 - std::chi_squared_distribution, 2235
 - std::discard_block_engine, 2418
 - std::discrete_distribution, 2423
 - std::exponential_distribution, 2438
 - std::extreme_value_distribution, 2442
 - std::fisher_f_distribution, 2446
 - std::gamma_distribution, 2499
 - std::geometric_distribution, 2504
 - std::independent_bits_engine, 2538
 - std::linear_congruential_engine, 2600
 - std::lognormal_distribution, 2652
 - std::negative_binomial_distribution, 2790
 - std::normal_distribution, 2796
 - std::numeric_limits, 2839
 - std::piecewise_constant_distribution, 2904
 - std::piecewise_linear_distribution, 2910
 - std::poisson_distribution, 2920
 - std::shuffle_order_engine, 3015
 - std::student_t_distribution, 3028
 - std::uniform_int_distribution, 3123
 - std::uniform_real_distribution, 3127
 - std::weibull_distribution, 3178
 - max_digits10
 - std::__numeric_limits_base, 1412
 - std::numeric_limits, 2843
 - max_element
 - sorting_algorithms, 175
 - max_element_minimal_n
 - __gnu_parallel::_Settings, 1220
 - max_exponent
 - std::__numeric_limits_base, 1412
 - std::numeric_limits, 2843
 - max_exponent10
 - std::__numeric_limits_base, 1412
 - std::numeric_limits, 2843
 - max_size
 - __gnu_cxx::__versa_string, 841
 - __gnu_debug::basic_string, 1088
 - std::basic_string, 2085
 - std::deque, 2408
 - std::forward_list, 2465
 - std::list, 2619
 - std::map, 2671
 - std::match_results, 2687
 - std::multimap, 2763
-

- std::multiset, 2782
- std::set, 2992
- std::vector, 3165
- mean
 - std::normal_distribution, 2796
 - std::poisson_distribution, 2920
- mem_fn
 - functors, 256
- Memory, 65
- memory, 3394
- memory_order
 - atomics, 212
- merge
 - sorting_algorithms, 176
 - std::forward_list, 2465
 - std::list, 2619, 2620
- merge.h, 3395
- merge_minimal_n
 - __gnu_parallel::_Settings, 1221
- merge_oversampling
 - __gnu_parallel::_Settings, 1221
- messages
 - std::locale, 2637
 - std::messages, 2700
- messages_members.h, 3396
- metaprogramming
 - _GLIBCXX_HAS_NESTED_-TYPE, 124
 - false_type, 124
 - true_type, 124
- microseconds
 - std::chrono, 737
- milliseconds
 - std::chrono, 737
- min
 - __gnu_parallel, 442
 - numeric_arrays, 94
 - sorting_algorithms, 177
 - std::bernoulli_distribution, 2196
 - std::binomial_distribution, 2207
 - std::cauchy_distribution, 2226
 - std::chi_squared_distribution, 2235
 - std::discard_block_engine, 2418
 - std::discrete_distribution, 2423
 - std::exponential_distribution, 2438
 - std::extreme_value_distribution, 2442
 - std::fisher_f_distribution, 2446
 - std::gamma_distribution, 2499
 - std::geometric_distribution, 2504
 - std::independent_bits_engine, 2538
 - std::linear_congruential_engine, 2600
 - std::lognormal_distribution, 2652
 - std::negative_binomial_distribution, 2790
 - std::normal_distribution, 2796
 - std::numeric_limits, 2840
 - std::piecewise_constant_distribution, 2904
 - std::piecewise_linear_distribution, 2910
 - std::poisson_distribution, 2920
 - std::shuffle_order_engine, 3016
 - std::student_t_distribution, 3028
 - std::uniform_int_distribution, 3123
 - std::uniform_real_distribution, 3127
 - std::weibull_distribution, 3178
- min_element
 - sorting_algorithms, 178
- min_element_minimal_n
 - __gnu_parallel::_Settings, 1221
- min_exponent
 - std::__numeric_limits_base, 1412
 - std::numeric_limits, 2843
- min_exponent10
 - std::__numeric_limits_base, 1413
 - std::numeric_limits, 2843
- minmax
 - sorting_algorithms, 179
- minmax_element
 - sorting_algorithms, 179, 180
- minstd_rand
 - random_generators, 283
- minstd_rand0
 - random_generators, 283
- minutes
 - std::chrono, 737
- mismatch
 - non_mutating_algorithms, 162
- modulus

- std::linear_congruential_engine, 2603
- monetary
 - std::locale, 2637
- money_get
 - std::money_get, 2714
- money_put
 - std::money_put, 2719
- moneypunct
 - std::moneypunct, 2726
- move
 - mutating_algorithms, 137
- move.h, 3396
- move_backward
 - mutating_algorithms, 137
- mt19937
 - random_generators, 283
- mt19937_64
 - random_generators, 283
- mt_allocator.h, 3397
- multimap
 - std::multimap, 2751–2753
- multimap.h, 3399, 3400
- multiplier
 - std::linear_congruential_engine, 2603
- multisec_partition
 - __gnu_parallel, 442
- multisec_selection
 - __gnu_parallel, 443
- multisec_selection.h, 3401
- multiset
 - std::multiset, 2771–2773
- multiset.h, 3402, 3403
- multiway_merge
 - __gnu_parallel, 444
- multiway_merge.h, 3404
 - _GLIBCXX_PARALLEL_LENGTH, 3410
- multiway_merge_3_variant
 - __gnu_parallel, 446
- multiway_merge_4_variant
 - __gnu_parallel, 446
- multiway_merge_exact_splitting
 - __gnu_parallel, 447
- multiway_merge_loser_tree
 - __gnu_parallel, 448
- multiway_merge_loser_tree_sentinel
 - __gnu_parallel, 448
- multiway_merge_loser_tree_unguarded
 - __gnu_parallel, 449
- multiway_merge_minimal_k
 - __gnu_parallel::_Settings, 1221
- multiway_merge_minimal_n
 - __gnu_parallel::_Settings, 1221
- multiway_merge_oversampling
 - __gnu_parallel::_Settings, 1221
- multiway_merge_sampling_splitting
 - __gnu_parallel, 450
- multiway_merge_sentinels
 - __gnu_parallel, 450
- multiway_mergesort.h, 3410
- Mutating, 129
- mutating_algorithms
 - copy, 132
 - copy_backward, 132
 - copy_if, 133
 - copy_n, 133
 - fill, 134
 - fill_n, 134
 - generate, 135
 - generate_n, 135
 - is_partitioned, 136
 - iter_swap, 136
 - move, 137
 - move_backward, 137
 - partition, 138
 - partition_copy, 138
 - partition_point, 139
 - random_shuffle, 139, 140
 - remove, 140
 - remove_copy, 141
 - remove_copy_if, 141
 - remove_if, 142
 - replace, 143
 - replace_copy_if, 143
 - replace_if, 144
 - reverse, 144
 - reverse_copy, 145
 - rotate, 145
 - rotate_copy, 146
 - shuffle, 146

- stable_partition, 147
- swap, 148
- swap_ranges, 148
- transform, 148, 149
- unique, 150
- unique_copy, 151
- mutex, 3411
- Mutexes, 70
- mutexes
 - call_once, 71
 - lock, 71
 - try_lock, 72
- name
 - std::locale, 2634
 - std::type_info, 3116
- nanoseconds
 - std::chrono, 737
- narrow
 - std::__ctype_abstract_base, 1331
 - std::basic_fstream, 1585
 - std::basic_ifstream, 1648
 - std::basic_ios, 1691
 - std::basic_iostream, 1734
 - std::basic_istream, 1794
 - std::basic_istream, 1849
 - std::basic_ofstream, 1896
 - std::basic_ostream, 1939
 - std::basic_ostringstream, 1984
 - std::basic_stringstream, 2159
 - std::ctype, 2307, 2308
 - std::ctype< char >, 2321, 2322
 - std::ctype< wchar_t >, 2343
 - std::ctype_byname, 2358, 2359
 - std::ctype_byname< char >, 2371, 2372
- native_handle
 - std::thread, 3038
- neg_format
 - std::moneypunct, 2732
 - std::moneypunct_byname, 2744
- negative_sign
 - std::moneypunct, 2733
 - std::moneypunct_byname, 2745
- Negators, 259
- negators
 - not1, 260
 - not2, 260
- nested_exception.h, 3413
- new, 3413
 - operator delete, 3415
 - operator new, 3417
- new_allocator.h, 3419
- new_handler
 - std, 604
- next_permutation
 - sorting_algorithms, 180, 181
- noboolalpha
 - std, 637
- Non-Mutating, 152
- non_mutating_algorithms
 - adjacent_find, 154
 - all_of, 155
 - any_of, 155
 - count, 156
 - count_if, 156
 - equal, 156, 157
 - find, 157
 - find_end, 158, 159
 - find_first_of, 159, 160
 - find_if, 161
 - find_if_not, 161
 - for_each, 161
 - mismatch, 162
 - none_of, 163
 - search, 163, 164
 - search_n, 165
- none
 - std::bitset, 2218
 - std::locale, 2637
- none_of
 - non_mutating_algorithms, 163
- norm
 - complex_numbers, 46
- Normal, 292
- normal_distribution
 - std::normal_distribution, 2796
- noshowbase
 - std, 637
- noshowpoint
 - std, 637
- noshowpos

- std, 637
- noskipws
 - std, 637
- nosubs
 - std::regex_constants, 759
- not1
 - negators, 260
- not2
 - negators, 260
- nounitbuf
 - std, 637
- nouppercase
 - std, 638
- npos
 - __gnu_cxx::__versa_string, 859
 - __gnu_debug::basic_string, 1102
 - std::basic_string, 2102
- nth_element
 - sorting_algorithms, 181, 182
- nth_element_minimal_n
 - __gnu_parallel::_Settings, 1222
- num_get
 - std::num_get, 2806
- num_put
 - std::num_put, 2826
- numeric, 3420, 3421
 - std::locale, 2638
- Numeric Arrays, 78
- numeric_arrays
 - ~gslice, 91
 - apply, 91
 - begin, 92
 - cshift, 92
 - end, 93
 - gslice, 88
 - gslice_array, 88
 - indirect_array, 89
 - mask_array, 89
 - max, 93
 - min, 94
 - operator<=, 97, 98
 - operator>=, 102, 103
 - operator*=, 95
 - operator~, 108
 - operator^=, 107
 - operator+, 96
 - operator+=, 96
 - operator-, 96
 - operator=, 96, 97
 - operator/=: 97
 - operator=, 98–102
 - operator%=, 94
 - operator&=, 95
 - resize, 108
 - shift, 108
 - size, 109
 - slice, 89
 - slice_array, 89
 - start, 109
 - stride, 109, 110
 - sum, 110
 - valarray, 90, 91
- numeric_traits.h, 3424
- numeric_fwd.h, 3424
- Numerics, 73
- numpunct
 - std::numpunct, 2871, 2872
- oct
 - std, 638
 - std::basic_fstream, 1619
 - std::basic_ifstream, 1673
 - std::basic_ios, 1703
 - std::basic_iostream, 1768
 - std::basic_istream, 1819
 - std::basic_istreamstream, 1874
 - std::basic_ofstream, 1918
 - std::basic_ostream, 1961
 - std::basic_ostreamstream, 2006
 - std::basic_stringstream, 2193
 - std::ios_base, 2567
- off_type
 - __gnu_cxx::enc_filebuf, 888
 - __gnu_cxx::stdio_filebuf, 957
 - __gnu_cxx::stdio_sync_filebuf, 989
 - std::basic_filebuf, 1528
 - std::basic_fstream, 1566, 1567
 - std::basic_ifstream, 1631
 - std::basic_ios, 1682
 - std::basic_iostream, 1717
 - std::basic_istream, 1779
 - std::basic_istreamstream, 1833

- std::basic_ofstream, 1885
- std::basic_ostream, 1929
- std::basic_ostringstream, 1973
- std::basic_streambuf, 2024
- std::basic_stringbuf, 2107
- std::basic_stringstream, 2140, 2141
- ofstream
 - io, 61
- omp_loop.h, 3427
- omp_loop_static.h, 3428
- open
 - __gnu_cxx::enc_filebuf, 893
 - __gnu_cxx::stdio_filebuf, 964, 965
 - std::basic_filebuf, 1534, 1535
 - std::basic_fstream, 1586
 - std::basic_ifstream, 1648
 - std::basic_ofstream, 1897
- openmode
 - std::basic_fstream, 1567
 - std::basic_ifstream, 1631
 - std::basic_ios, 1682
 - std::basic_iostream, 1717
 - std::basic_istream, 1779
 - std::basic_istringstream, 1833
 - std::basic_ofstream, 1886
 - std::basic_ostream, 1929
 - std::basic_ostringstream, 1974
 - std::basic_stringstream, 2141
 - std::ios_base, 2556
- operator_iterator
 - __gnu_debug::_Safe_iterator, 1035
- operator_RAlter
 - __gnu_parallel::_GuardedIterator, 1158
- operator_bool
 - std::basic_istream::sentry, 1823
 - std::basic_ostream::sentry, 1964
 - std::function< _Res(_-ArgTypes...)>, 2482
- operator_delete
 - new, 3415
- operator_new
 - new, 3417
- operator_streamoff
 - std::fpos, 2473
- operator_string_type
 - std::sub_match, 3034
- operator_void_*
 - std::basic_fstream, 1586
 - std::basic_ifstream, 1649
 - std::basic_ios, 1691
 - std::basic_iostream, 1735
 - std::basic_istream, 1795
 - std::basic_istringstream, 1850
 - std::basic_ofstream, 1897
 - std::basic_ostream, 1939
 - std::basic_ostringstream, 1984
 - std::basic_stringstream, 2159
- operator_less
 - __gnu_cxx, 369, 370
 - __gnu_parallel::_GuardedIterator, 1159
 - regex, 228–231
 - std, 644–650
 - std::multiset, 2785
- operator_less_less
 - complex_numbers, 50
 - pointer_abstractions, 69
 - random_distributions_bernoulli, 298
 - random_distributions_normal, 294
 - random_distributions_poisson, 304, 305
 - random_distributions_uniform, 289, 290
 - random_generators, 286
 - regex, 231
 - std, 650–657
 - std::basic_fstream, 1587–1593
 - std::basic_iostream, 1735–1742
 - std::basic_ofstream, 1898–1904
 - std::basic_ostream, 1940–1946
 - std::basic_ostringstream, 1985–1991
 - std::basic_stringstream, 2159–2166
 - std::binomial_distribution, 2209
 - std::bitset, 2219
 - std::chi_squared_distribution, 2236
 - std::discard_block_engine, 2420
 - std::discrete_distribution, 2424
 - std::fisher_f_distribution, 2447
 - std::gamma_distribution, 2501
 - std::linear_congruential_engine, 2602

- std::lognormal_distribution, 2653
- std::negative_binomial_distribution, 2792
- std::normal_distribution, 2798
- std::piecewise_constant_distribution, 2906
- std::piecewise_linear_distribution, 2911
- std::poisson_distribution, 2922
- std::shuffle_order_engine, 3017
- std::student_t_distribution, 3029
- operator<<=
 - numeric_arrays, 97, 98
 - std::bitset, 2219
 - std::indirect_array, 2543
 - std::mask_array, 2678
 - std::slice_array, 3021
- operator<=
 - __gnu_cxx, 371, 372
 - __gnu_parallel::_GuardedIterator, 1160
 - regex, 231–234
 - std, 657–661
 - std::rel_ops, 760
- operator>
 - __gnu_cxx, 374
 - regex, 237–240
 - std, 668–671
 - std::rel_ops, 760
- operator>>
 - complex_numbers, 52
 - random_distributions_bernoulli, 299, 300
 - random_distributions_normal, 295
 - random_distributions_poisson, 307, 308
 - random_distributions_uniform, 291
 - std, 674–680
 - std::basic_fstream, 1594–1600
 - std::basic_ifstream, 1649–1656
 - std::basic_iostream, 1742–1748
 - std::basic_istream, 1795–1802
 - std::basic_istreamstream, 1850–1856
 - std::basic_stringstream, 2166–2172
 - std::binomial_distribution, 2209
 - std::bitset, 2219
 - std::chi_squared_distribution, 2236
 - std::discard_block_engine, 2420
 - std::discrete_distribution, 2424
 - std::fisher_f_distribution, 2447
 - std::gamma_distribution, 2501
 - std::independent_bits_engine, 2540
 - std::linear_congruential_engine, 2602
 - std::lognormal_distribution, 2654
 - std::negative_binomial_distribution, 2792
 - std::normal_distribution, 2799
 - std::piecewise_constant_distribution, 2906
 - std::piecewise_linear_distribution, 2911
 - std::poisson_distribution, 2922
 - std::shuffle_order_engine, 3018
 - std::student_t_distribution, 3029
- operator>>=
 - numeric_arrays, 102, 103
 - std::bitset, 2219
 - std::indirect_array, 2543
 - std::mask_array, 2678
 - std::slice_array, 3022
- operator>=
 - __gnu_cxx, 375, 376
 - regex, 240–242
 - std, 671–674
 - std::rel_ops, 761
- operator*
 - __gnu_debug::_Safe_iterator, 1035
 - __gnu_parallel::_GuardedIterator, 1159
 - complex_numbers, 47
 - std::auto_ptr, 1511
 - std::back_insert_iterator, 1517
 - std::front_insert_iterator, 2477
 - std::insert_iterator, 2548
 - std::istreambuf_iterator, 2588
 - std::ostreambuf_iterator, 2892
 - std::regex_iterator, 2951
 - std::regex_token_iterator, 2957
 - std::reverse_iterator, 2969
- operator*=
 - complex_numbers, 48

- numeric_arrays, 95
- std::indirect_array, 2542
- std::mask_array, 2678
- std::slice_array, 3021
- operator~
 - numeric_arrays, 108
 - std::bitset, 2221
- operator^
 - std, 681
- operator^=
 - numeric_arrays, 107
 - std::bitset, 2221
 - std::indirect_array, 2543
 - std::mask_array, 2678
 - std::slice_array, 3022
- operator()
 - __gnu_cxx::subtractive_rng, 1010
 - __gnu_parallel::_Nothing, 1200
 - __gnu_parallel::_RandomNumber, 1212
 - __gnu_parallel::_accumulate_selector, 1104
 - __gnu_parallel::_adjacent_find_selector, 1108
 - __gnu_parallel::_count_if_selector, 1113
 - __gnu_parallel::_count_selector, 1115
 - __gnu_parallel::_fill_selector, 1117
 - __gnu_parallel::_find_first_of_selector, 1119
 - __gnu_parallel::_find_if_selector, 1121
 - __gnu_parallel::_for_each_selector, 1122
 - __gnu_parallel::_generate_selector, 1124
 - __gnu_parallel::_identity_selector, 1128
 - __gnu_parallel::_inner_product_selector, 1130
 - __gnu_parallel::_mismatch_selector, 1134
 - __gnu_parallel::_replace_if_selector, 1140
 - __gnu_parallel::_replace_selector, 1142
 - __gnu_parallel::_transform1_selector, 1144
 - __gnu_parallel::_transform2_selector, 1146
 - std::bernoulli_distribution, 2196
 - std::binomial_distribution, 2207
 - std::cauchy_distribution, 2226
 - std::chi_squared_distribution, 2235
 - std::discard_block_engine, 2418
 - std::discrete_distribution, 2423
 - std::exponential_distribution, 2438
 - std::extreme_value_distribution, 2442
 - std::fisher_f_distribution, 2446
 - std::function< _Res(_ArgTypes...)>, 2482
 - std::gamma_distribution, 2499
 - std::geometric_distribution, 2504
 - std::independent_bits_engine, 2538
 - std::linear_congruential_engine, 2600
 - std::locale, 2634
 - std::lognormal_distribution, 2652
 - std::negative_binomial_distribution, 2790
 - std::normal_distribution, 2797
 - std::piecewise_constant_distribution, 2904
 - std::piecewise_linear_distribution, 2910
 - std::poisson_distribution, 2920
 - std::shuffle_order_engine, 3016
 - std::student_t_distribution, 3028
 - std::uniform_int_distribution, 3124
 - std::uniform_real_distribution, 3127
 - std::weibull_distribution, 3178
- operator+
 - __gnu_cxx, 367–369
 - complex_numbers, 48
 - numeric_arrays, 96
 - std, 642–644
 - std::fpos, 2473
 - std::reverse_iterator, 2970
- operator++

-
- `__gnu_debug::_Safe_iterator`, 1035, 1036
 - `__gnu_parallel::_GuardedIterator`, 1159
 - `std::back_insert_iterator`, 1517
 - `std::front_insert_iterator`, 2477
 - `std::insert_iterator`, 2548
 - `std::istreambuf_iterator`, 2588
 - `std::ostreambuf_iterator`, 2892
 - `std::regex_iterator`, 2951
 - `std::regex_token_iterator`, 2957
 - `std::reverse_iterator`, 2970
 - `operator+=`
 - `__gnu_cxx::__versa_string`, 841, 842
 - `__gnu_debug::basic_string`, 1088, 1089
 - `complex_numbers`, 49
 - `numeric_arrays`, 96
 - `std::basic_string`, 2085, 2086
 - `std::complex`, 2286
 - `std::fpos`, 2473
 - `std::indirect_array`, 2543
 - `std::mask_array`, 2678
 - `std::reverse_iterator`, 2970
 - `std::slice_array`, 3021
 - `operator-`
 - `complex_numbers`, 49
 - `numeric_arrays`, 96
 - `std::fpos`, 2473
 - `std::reverse_iterator`, 2971
 - `operator->`
 - `__gnu_debug::_Safe_iterator`, 1036
 - `std::auto_ptr`, 1511
 - `std::regex_iterator`, 2952
 - `std::regex_token_iterator`, 2958
 - `std::reverse_iterator`, 2972
 - `operator--`
 - `__gnu_debug::_Safe_iterator`, 1036
 - `std::reverse_iterator`, 2971
 - `operator-=`
 - `complex_numbers`, 49
 - `numeric_arrays`, 96, 97
 - `std::complex`, 2286
 - `std::fpos`, 2474
 - `std::indirect_array`, 2543
 - `std::mask_array`, 2678
 - `std::reverse_iterator`, 2972
 - `std::slice_array`, 3021
 - `operator/`
 - `complex_numbers`, 50
 - `operator/=`
 - `complex_numbers`, 50
 - `numeric_arrays`, 97
 - `std::indirect_array`, 2543
 - `std::mask_array`, 2678
 - `std::slice_array`, 3021
 - `operator=`
 - `__gnu_cxx::__versa_string`, 842–844
 - `__gnu_debug::_Safe_iterator`, 1037
 - `complex_numbers`, 51
 - `numeric_arrays`, 98–102
 - `std::auto_ptr`, 1512
 - `std::back_insert_iterator`, 1517
 - `std::basic_regex`, 2017, 2018
 - `std::basic_string`, 2087, 2088
 - `std::deque`, 2408, 2409
 - `std::forward_list`, 2466
 - `std::front_insert_iterator`, 2478
 - `std::function<_Res(_ArgTypes...)>`, 2483–2485
 - `std::insert_iterator`, 2549
 - `std::list`, 2620, 2621
 - `std::locale`, 2635
 - `std::map`, 2671, 2672
 - `std::match_results`, 2687
 - `std::multimap`, 2763, 2764
 - `std::multiset`, 2782, 2783
 - `std::ostream_iterator`, 2888
 - `std::ostreambuf_iterator`, 2892
 - `std::regex_iterator`, 2952
 - `std::regex_token_iterator`, 2958
 - `std::set`, 2992, 2993
 - `std::vector`, 3165, 3166
 - `operator==`
 - `__gnu_cxx`, 372, 373
 - `complex_numbers`, 51
 - `iterators`, 275, 276
 - `random_distributions_bernoulli`, 299
 - `random_distributions_normal`, 295
-

- random_distributions_poisson, [305](#), [306](#)
- random_distributions_uniform, [290](#)
- regex, [234–237](#)
- std, [661–667](#)
- std::binomial_distribution, [2209](#)
- std::bitset, [2219](#)
- std::chi_squared_distribution, [2236](#)
- std::discard_block_engine, [2420](#)
- std::fisher_f_distribution, [2447](#)
- std::gamma_distribution, [2501](#)
- std::independent_bits_engine, [2540](#)
- std::linear_congruential_engine, [2602](#)
- std::locale, [2635](#)
- std::lognormal_distribution, [2653](#)
- std::multiset, [2785](#)
- std::negative_binomial_distribution, [2792](#)
- std::normal_distribution, [2799](#)
- std::poisson_distribution, [2922](#)
- std::regex_iterator, [2952](#)
- std::regex_token_iterator, [2958](#)
- std::shuffle_order_engine, [3017](#)
- std::student_t_distribution, [3029](#)
- operator%=
 - numeric_arrays, [94](#)
 - std::indirect_array, [2542](#)
 - std::mask_array, [2677](#)
 - std::slice_array, [3021](#)
- operator&
 - std, [642](#)
- operator&=
 - numeric_arrays, [95](#)
 - std::bitset, [2218](#)
 - std::indirect_array, [2542](#)
 - std::mask_array, [2677](#)
 - std::slice_array, [3021](#)
- optimize
 - std::regex_constants, [759](#)
- os_defines.h, [3428](#)
- ostream, [3429](#)
 - io, [62](#)
- ostream.tcc, [3430](#)
- ostream_insert.h, [3431](#)
- ostream_iterator
 - std::ostream_iterator, [2887](#)
- ostream_type
 - std::ostream_iterator, [2886](#)
 - std::ostreambuf_iterator, [2890](#)
- ostreambuf_iterator
 - std::ostreambuf_iterator, [2891](#)
- ostringstream
 - io, [62](#)
- out
 - std::__codecvt_abstract_base, [1319](#)
 - std::basic_fstream, [1620](#)
 - std::basic_ifstream, [1673](#)
 - std::basic_ios, [1704](#)
 - std::basic_iostream, [1768](#)
 - std::basic_istream, [1819](#)
 - std::basic_istreamstream, [1875](#)
 - std::basic_ofstream, [1919](#)
 - std::basic_ostream, [1961](#)
 - std::basic_ostreamstream, [2006](#)
 - std::basic_stringstream, [2193](#)
 - std::codecvt, [2246](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [2252](#)
 - std::codecvt< char, char, mbstate_t >, [2257](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [2262](#)
 - std::codecvt_byname, [2269](#)
 - std::ios_base, [2568](#)
- overflow
 - __gnu_cxx::enc_filebuf, [894](#)
 - __gnu_cxx::stdio_filebuf, [965](#)
 - __gnu_cxx::stdio_sync_filebuf, [993](#)
 - std::basic_filebuf, [1535](#)
 - std::basic_streambuf, [2029](#)
 - std::basic_stringbuf, [2111](#)
- p
 - std::bernoulli_distribution, [2196](#)
 - std::binomial_distribution, [2208](#)
 - std::geometric_distribution, [2504](#)
 - std::negative_binomial_distribution, [2790](#)
- pair
 - std::pair, [2901](#)
- par_loop.h, [3431](#)

-
- parallel.h, 3432
 - parallel_balanced
 - __gnu_parallel, 407
 - parallel_omp_loop
 - __gnu_parallel, 408
 - parallel_omp_loop_static
 - __gnu_parallel, 408
 - parallel_taskqueue
 - __gnu_parallel, 408
 - parallel_unbalanced
 - __gnu_parallel, 407
 - parallel_multiway_merge
 - __gnu_parallel, 452
 - parallel_sort_mwms
 - __gnu_parallel, 453
 - parallel_sort_mwms_pu
 - __gnu_parallel, 453
 - parallel_tag
 - __gnu_parallel::parallel_tag, 1247
 - param
 - std::bernoulli_distribution, 2196, 2197
 - std::binomial_distribution, 2208
 - std::cauchy_distribution, 2227
 - std::chi_squared_distribution, 2235
 - std::discrete_distribution, 2423
 - std::exponential_distribution, 2439
 - std::extreme_value_distribution, 2443
 - std::fisher_f_distribution, 2446
 - std::gamma_distribution, 2500
 - std::geometric_distribution, 2505
 - std::lognormal_distribution, 2652, 2653
 - std::negative_binomial_distribution, 2791
 - std::normal_distribution, 2797
 - std::piecewise_constant_distribution, 2905
 - std::piecewise_linear_distribution, 2910
 - std::poisson_distribution, 2921
 - std::student_t_distribution, 3028
 - std::uniform_int_distribution, 3124
 - std::uniform_real_distribution, 3128
 - std::weibull_distribution, 3178, 3179
 - partial_sort
 - sorting_algorithms, 183
 - partial_sort_copy
 - sorting_algorithms, 184
 - partial_sort_minimal_n
 - __gnu_parallel::_Settings, 1222
 - partial_sum
 - std, 682
 - partial_sum.h, 3432
 - partial_sum_dilation
 - __gnu_parallel::_Settings, 1222
 - partial_sum_minimal_n
 - __gnu_parallel::_Settings, 1222
 - partition
 - mutating_algorithms, 138
 - partition.h, 3433
 - _GLIBCXX_VOLATILE, 3434
 - partition_chunk_share
 - __gnu_parallel::_Settings, 1222
 - partition_chunk_size
 - __gnu_parallel::_Settings, 1222
 - partition_copy
 - mutating_algorithms, 138
 - partition_minimal_n
 - __gnu_parallel::_Settings, 1223
 - partition_point
 - mutating_algorithms, 139
 - pbackfail
 - __gnu_cxx::enc_filebuf, 894
 - __gnu_cxx::stdio_filebuf, 966
 - __gnu_cxx::stdio_sync_filebuf, 994
 - std::basic_filebuf, 1536
 - std::basic_streambuf, 2029
 - std::basic_stringbuf, 2112
 - pbase
 - __gnu_cxx::enc_filebuf, 895
 - __gnu_cxx::stdio_filebuf, 966
 - __gnu_cxx::stdio_sync_filebuf, 994
 - std::basic_filebuf, 1536
 - std::basic_streambuf, 2030
 - std::basic_stringbuf, 2113
 - pbump
 - __gnu_cxx::enc_filebuf, 895
 - __gnu_cxx::stdio_filebuf, 967
 - __gnu_cxx::stdio_sync_filebuf, 995
 - std::basic_filebuf, 1537
-

- std::basic_streambuf, 2030
- std::basic_stringbuf, 2113
- peek
 - std::basic_fstream, 1600
 - std::basic_ifstream, 1656
 - std::basic_iostream, 1749
 - std::basic_istream, 1802
 - std::basic_istreamstream, 1857
 - std::basic_stringstream, 2173
- pod_char_traits.h, 3434
- pointer
 - std::back_insert_iterator, 1516
 - std::front_insert_iterator, 2476
 - std::insert_iterator, 2547
 - std::istream_iterator, 2583
 - std::istreambuf_iterator, 2586
 - std::iterator, 2590
 - std::ostream_iterator, 2886
 - std::ostreambuf_iterator, 2890
 - std::raw_storage_iterator, 2943
 - std::reverse_iterator, 2968
 - std::set, 2980
- Pointer Abstractions, 65
- pointer.h, 3435
- pointer_abstractions
 - allocate_shared, 68
 - get_deleter, 69
 - make_shared, 69
 - operator<<, 69
- pointer_adaptors
 - ptr_fun, 261
- Poisson, 300
- polar
 - complex_numbers, 52
- Policy-Based Data Structures, 279
- pool_allocator.h, 3438
- pop
 - std::priority_queue, 2926
 - std::queue, 2933
 - std::stack, 3025
- pop_back
 - __gnu_parallel::_-
 - RestrictedBoundedConcurrentQueue, 1214
 - std::deque, 2410
 - std::list, 2621
 - std::vector, 3167
- pop_front
 - __gnu_parallel::_-
 - RestrictedBoundedConcurrentQueue, 1214
 - std::deque, 2411
 - std::forward_list, 2467
 - std::list, 2622
- pop_heap
 - heap_algorithms, 267
- pos_format
 - std::moneypunct, 2733
 - std::moneypunct_byname, 2745
- pos_type
 - __gnu_cxx::enc_filebuf, 888
 - __gnu_cxx::stdio_filebuf, 957
 - __gnu_cxx::stdio_sync_filebuf, 989
 - std::basic_filebuf, 1528
 - std::basic_fstream, 1567, 1568
 - std::basic_ifstream, 1631
 - std::basic_ios, 1682
 - std::basic_iostream, 1718
 - std::basic_istream, 1780
 - std::basic_istreamstream, 1833
 - std::basic_ofstream, 1886
 - std::basic_ostream, 1930
 - std::basic_ostringstream, 1974
 - std::basic_streambuf, 2024
 - std::basic_stringbuf, 2107
 - std::basic_stringstream, 2141, 2142
- position
 - std::match_results, 2687
- positive_sign
 - std::moneypunct, 2734
 - std::moneypunct_byname, 2746
- postypes.h, 3438
- pow
 - complex_numbers, 52
- power
 - SGIextensions, 23
- pptr
 - __gnu_cxx::enc_filebuf, 896
 - __gnu_cxx::stdio_filebuf, 967
 - __gnu_cxx::stdio_sync_filebuf, 995
 - std::basic_filebuf, 1537
 - std::basic_streambuf, 2030

- std::basic_stringbuf, 2113
- precision
 - std::basic_fstream, 1600, 1601
 - std::basic_ifstream, 1656, 1657
 - std::basic_ios, 1692
 - std::basic_istream, 1749
 - std::basic_istream, 1802, 1803
 - std::basic_istream, 1857, 1858
 - std::basic_ofstream, 1905
 - std::basic_ostream, 1946, 1947
 - std::basic_ostringstream, 1991, 1992
 - std::basic_stringstream, 2173
 - std::ios_base, 2559, 2560
- prefix
 - std::match_results, 2688
- prev_permutation
 - sorting_algorithms, 185, 186
- priority_queue
 - std::priority_queue, 2925
- priority_queue.hpp, 3439
- priority_queue_base_dispatch.hpp, 3440
- probabilities
 - std::discrete_distribution, 2424
- profiler.h, 3440
- profiler_algos.h, 3443
- profiler_hashtable_size.h, 3444
- profiler_list_to_slist.h, 3445
- profiler_list_to_vector.h, 3445
- profiler_map_to_unordered_map.h, 3446
- profiler_node.h, 3448
- profiler_state.h, 3449
- profiler_trace.h, 3449
- profiler_vector_size.h, 3452
- profiler_vector_to_list.h, 3452
- ptr_fun
 - pointer_adaptors, 261
- pubimbue
 - __gnu_cxx::enc_filebuf, 896
 - __gnu_cxx::stdio_filebuf, 968
 - __gnu_cxx::stdio_sync_filebuf, 996
 - std::basic_filebuf, 1537
 - std::basic_streambuf, 2031
 - std::basic_stringbuf, 2114
- pubseekoff
 - __gnu_cxx::enc_filebuf, 896
 - __gnu_cxx::stdio_filebuf, 968
 - __gnu_cxx::stdio_sync_filebuf, 996
 - std::basic_filebuf, 1538
 - std::basic_streambuf, 2032
 - std::basic_stringbuf, 2114
- pubseekpos
 - __gnu_cxx::enc_filebuf, 897
 - __gnu_cxx::stdio_filebuf, 968
 - __gnu_cxx::stdio_sync_filebuf, 996
 - std::basic_filebuf, 1538
 - std::basic_streambuf, 2032
 - std::basic_stringbuf, 2114
- pubsetbuf
 - __gnu_cxx::enc_filebuf, 897
 - __gnu_cxx::stdio_filebuf, 969
 - __gnu_cxx::stdio_sync_filebuf, 997
 - std::basic_filebuf, 1539
 - std::basic_streambuf, 2032
 - std::basic_stringbuf, 2115
- pubsync
 - __gnu_cxx::enc_filebuf, 897
 - __gnu_cxx::stdio_filebuf, 969
 - __gnu_cxx::stdio_sync_filebuf, 997
 - std::basic_filebuf, 1539
 - std::basic_streambuf, 2032
 - std::basic_stringbuf, 2115
- push
 - std::priority_queue, 2926
 - std::queue, 2933
 - std::stack, 3025
- push_back
 - __gnu_cxx::__versa_string, 845
 - __gnu_debug::basic_string, 1090
 - std::basic_string, 2089
 - std::deque, 2411
 - std::list, 2622
 - std::vector, 3168
- push_front
 - __gnu_parallel::_-
RestrictedBoundedConcurrentQueue,
1214
 - std::deque, 2411
 - std::forward_list, 2467
 - std::list, 2622
- push_heap
 - heap_algorithms, 268
- put

- std::basic_fstream, 1601
- std::basic_iostream, 1750
- std::basic_ofstream, 1905
- std::basic_ostream, 1947
- std::basic_ostringstream, 1992
- std::basic_stringstream, 2174
- std::money_put, 2721, 2722
- std::num_put, 2830–2836
- std::time_put, 3066, 3067
- std::time_put_byname, 3070, 3071
- put_money
 - std, 683
- putback
 - std::basic_fstream, 1602
 - std::basic_ifstream, 1657
 - std::basic_iostream, 1750
 - std::basic_istream, 1803
 - std::basic_istream, 1858
 - std::basic_stringstream, 2174
- pword
 - std::basic_fstream, 1602
 - std::basic_ifstream, 1658
 - std::basic_ios, 1693
 - std::basic_iostream, 1751
 - std::basic_istream, 1804
 - std::basic_istream, 1859
 - std::basic_ofstream, 1906
 - std::basic_ostream, 1948
 - std::basic_ostringstream, 1992
 - std::basic_stringstream, 2175
 - std::ios_base, 2560
- qsb_steals
 - __gnu_parallel::_Settings, 1223
- queue, 3453
 - std::queue, 2932
- queue.h, 3454
 - _GLIBCXX_VOLATILE, 3454
- quicksort.h, 3455
- quiet_NaN
 - std::numeric_limits, 2840
- radix
 - std::__numeric_limits_base, 1413
 - std::numeric_limits, 2844
- random, 3455
 - generate_canonical, 217
 - Random Number Distributions, 287
 - Random Number Generation, 216
 - Random Number Generators, 280
 - Random Number Utilities, 308
 - random.h, 3456
 - random.tcc, 3463
 - random_distributions_bernoulli
 - operator<<, 298
 - operator>>, 299, 300
 - operator==, 299
 - random_distributions_normal
 - operator<<, 294
 - operator>>, 295
 - operator==, 295
 - random_distributions_poisson
 - operator<<, 304, 305
 - operator>>, 307, 308
 - operator==, 305, 306
 - random_distributions_uniform
 - operator<<, 289, 290
 - operator>>, 291
 - operator==, 290
 - random_generators
 - minstd_rand, 283
 - minstd_rand0, 283
 - mt19937, 283
 - mt19937_64, 283
 - operator<<, 286
 - random_number.h, 3468
 - random_sample
 - SGIextensions, 23, 24
 - random_sample_n
 - SGIextensions, 24
 - random_shuffle
 - mutating_algorithms, 139, 140
 - random_shuffle.h, 3468
 - random_shuffle_minimal_n
 - __gnu_parallel::_Settings, 1223
 - range_access.h, 3470
 - ratio, 3470
 - Rational Arithmetic, 74
 - rb_tree, 3471
 - rbegin
 - __gnu_cxx::__versa_string, 845
 - __gnu_debug::basic_string, 1090

- [std::basic_string](#), [2089](#), [2090](#)
 - [std::deque](#), [2412](#)
 - [std::list](#), [2623](#)
 - [std::map](#), [2673](#)
 - [std::multimap](#), [2764](#)
 - [std::multiset](#), [2783](#)
 - [std::set](#), [2993](#)
 - [std::vector](#), [3168](#)
- [rc_string_base.h](#), [3472](#)
- [rdbuf](#)
 - [std::basic_fstream](#), [1603](#)
 - [std::basic_ifstream](#), [1658](#)
 - [std::basic_ios](#), [1693](#), [1694](#)
 - [std::basic_iostream](#), [1751](#), [1752](#)
 - [std::basic_istream](#), [1804](#), [1805](#)
 - [std::basic_istreamstream](#), [1859](#)
 - [std::basic_ofstream](#), [1906](#)
 - [std::basic_ostream](#), [1948](#), [1949](#)
 - [std::basic_ostreamstream](#), [1993](#), [1994](#)
 - [std::basic_stringstream](#), [2175](#), [2176](#)
- [rdstate](#)
 - [std::basic_fstream](#), [1604](#)
 - [std::basic_ifstream](#), [1659](#)
 - [std::basic_ios](#), [1694](#)
 - [std::basic_iostream](#), [1752](#)
 - [std::basic_istream](#), [1805](#)
 - [std::basic_istreamstream](#), [1860](#)
 - [std::basic_ofstream](#), [1907](#)
 - [std::basic_ostream](#), [1949](#)
 - [std::basic_ostreamstream](#), [1994](#)
 - [std::basic_stringstream](#), [2176](#)
- [read](#)
 - [std::basic_fstream](#), [1604](#)
 - [std::basic_ifstream](#), [1659](#)
 - [std::basic_iostream](#), [1753](#)
 - [std::basic_istream](#), [1806](#)
 - [std::basic_istreamstream](#), [1860](#)
 - [std::basic_stringstream](#), [2177](#)
- [readsome](#)
 - [std::basic_fstream](#), [1605](#)
 - [std::basic_ifstream](#), [1660](#)
 - [std::basic_iostream](#), [1753](#)
 - [std::basic_istream](#), [1806](#)
 - [std::basic_istreamstream](#), [1861](#)
 - [std::basic_stringstream](#), [2177](#)
- [ref](#)
 - [std](#), [683](#)
- [reference](#)
 - [std::back_insert_iterator](#), [1516](#)
 - [std::front_insert_iterator](#), [2476](#)
 - [std::insert_iterator](#), [2547](#)
 - [std::istream_iterator](#), [2583](#)
 - [std::istreambuf_iterator](#), [2587](#)
 - [std::iterator](#), [2590](#)
 - [std::ostream_iterator](#), [2886](#)
 - [std::ostreambuf_iterator](#), [2890](#)
 - [std::raw_storage_iterator](#), [2943](#)
 - [std::reverse_iterator](#), [2968](#)
 - [std::set](#), [2980](#)
- [regex](#), [3472](#)
 - [cregex_token_iterator](#), [223](#)
 - [csub_match](#), [223](#)
 - [isctype](#), [225](#)
 - [operator<](#), [228–231](#)
 - [operator<<](#), [231](#)
 - [operator<=](#), [231–234](#)
 - [operator>](#), [237–240](#)
 - [operator>=](#), [240–242](#)
 - [operator==](#), [234–237](#)
 - [regex](#), [223](#)
 - [regex_match](#), [243–246](#)
 - [regex_replace](#), [247](#)
 - [regex_search](#), [248–252](#)
 - [sregex_token_iterator](#), [224](#)
 - [ssub_match](#), [224](#)
 - [swap](#), [252](#), [253](#)
 - [value](#), [253](#)
 - [wcregex_token_iterator](#), [224](#)
 - [wcsub_match](#), [224](#)
 - [wregex](#), [224](#)
 - [wsregex_token_iterator](#), [224](#)
 - [wssub_match](#), [224](#)
- [regex.h](#), [3472](#)
- [regex_compiler.h](#), [3479](#)
- [regex_constants.h](#), [3479](#)
- [regex_cursor.h](#), [3481](#)
- [regex_error](#)
 - [std::regex_error](#), [2948](#)
- [regex_error.h](#), [3481](#)
- [regex_grep_matcher.h](#), [3482](#)
- [regex_grep_matcher.tcc](#), [3483](#)
- [regex_iterator](#)

- std::regex_iterator, 2950
- regex_match
 - regex, 243–246
- regex_nfa.h, 3483
- regex_nfa.tcc, 3484
- regex_replace
 - regex, 247
- regex_search
 - regex, 248–252
- regex_token_iterator
 - std::regex_token_iterator, 2954–2956
- regex_traits
 - std::regex_traits, 2960
- register_callback
 - std::basic_fstream, 1605
 - std::basic_ifstream, 1661
 - std::basic_ios, 1694
 - std::basic_iostream, 1754
 - std::basic_istream, 1807
 - std::basic_istreamstream, 1862
 - std::basic_ofstream, 1907
 - std::basic_ostream, 1950
 - std::basic_ostreamstream, 1994
 - std::basic_stringstream, 2178
 - std::ios_base, 2560
- Regular Expressions, 217
- release
 - std::auto_ptr, 1512
- remove
 - mutating_algorithms, 140
 - std::forward_list, 2467
 - std::list, 2623
- remove_copy
 - mutating_algorithms, 141
- remove_copy_if
 - mutating_algorithms, 141
- remove_if
 - mutating_algorithms, 142
 - std::forward_list, 2468
 - std::list, 2623
- rend
 - __gnu_cxx::__versa_string, 846
 - __gnu_debug::basic_string, 1091
 - std::basic_string, 2090
 - std::deque, 2412
 - std::list, 2624
 - std::map, 2673, 2674
 - std::multimap, 2764, 2765
 - std::multiset, 2783
 - std::set, 2994
 - std::vector, 3168
- replace
 - __gnu_cxx::__versa_string, 846–853
 - __gnu_debug::basic_string, 1091–1097
 - mutating_algorithms, 143
 - std::basic_string, 2090–2096
- replace_copy
 - std, 684
- replace_copy_if
 - mutating_algorithms, 143
- replace_if
 - mutating_algorithms, 144
- replace_minimal_n
 - __gnu_parallel::_Settings, 1223
- requested_size
 - __gnu_cxx::temporary_buffer, 1013
 - std::_Temporary_buffer, 1479
- reserve
 - __gnu_cxx::__versa_string, 854
 - __gnu_debug::basic_string, 1097
 - std::basic_string, 2097
 - std::vector, 3169
- reset
 - std::auto_ptr, 1513
 - std::bernoulli_distribution, 2197
 - std::binomial_distribution, 2208
 - std::bitset, 2221, 2222
 - std::cauchy_distribution, 2227
 - std::chi_squared_distribution, 2236
 - std::discrete_distribution, 2424
 - std::exponential_distribution, 2439
 - std::extreme_value_distribution, 2443
 - std::fisher_f_distribution, 2447
 - std::gamma_distribution, 2500
 - std::geometric_distribution, 2505
 - std::lognormal_distribution, 2653
 - std::negative_binomial_distribution, 2791

- std::normal_distribution, 2798
- std::piecewise_constant_-distribution, 2905
- std::piecewise_linear_distribution, 2911
- std::poisson_distribution, 2921
- std::student_t_distribution, 3029
- std::uniform_int_distribution, 3124
- std::uniform_real_distribution, 3128
- std::weibull_distribution, 3179
- resetiosflags
 - std, 684
- resize
 - __gnu_cxx::__versa_string, 854, 855
 - __gnu_debug::basic_string, 1098
 - numeric_arrays, 108
 - std::basic_string, 2098
 - std::deque, 2413
 - std::forward_list, 2468
 - std::list, 2624, 2625
 - std::vector, 3169, 3170
- result_type
 - __gnu_cxx::__detail::_Ffit_finder, 782
 - __gnu_cxx::binary_compose, 874
 - __gnu_cxx::project1st, 931
 - __gnu_cxx::project2nd, 932
 - __gnu_cxx::select1st, 947
 - __gnu_cxx::select2nd, 948
 - __gnu_cxx::subtractive_rng, 1010
 - __gnu_cxx::unary_compose, 1024
 - __gnu_parallel::_EqualFromLess, 1155
 - __gnu_parallel::_EqualTo, 1157
 - __gnu_parallel::_Less, 1167
 - __gnu_parallel::_Lexicographic, 1169
 - __gnu_parallel::_-LexicographicReverse, 1171
 - __gnu_parallel::_Multiplies, 1199
 - __gnu_parallel::_Plus, 1203
 - __gnu_parallel::_binder1st, 1110
 - __gnu_parallel::_binder2nd, 1112
 - __gnu_parallel::_unary_negate, 1148
 - std::_Maybe_unary_or_binary_-function< _Res, _T1 >, 1464
 - std::_Maybe_unary_or_binary_-function< _Res, _T1, _T2 >, 1465
 - std::bernoulli_distribution, 2195
 - std::binary_function, 2200
 - std::binary_negate, 2202
 - std::binder1st, 2203
 - std::binder2nd, 2205
 - std::binomial_distribution, 2207
 - std::cauchy_distribution, 2226
 - std::chi_squared_distribution, 2234
 - std::const_mem_fun1_ref_t, 2291
 - std::const_mem_fun1_t, 2293
 - std::const_mem_fun_ref_t, 2295
 - std::const_mem_fun_t, 2297
 - std::discard_block_engine, 2416
 - std::discrete_distribution, 2422
 - std::divides, 2427
 - std::equal_to, 2432
 - std::exponential_distribution, 2437
 - std::extreme_value_distribution, 2441
 - std::fisher_f_distribution, 2445
 - std::gamma_distribution, 2498
 - std::geometric_distribution, 2504
 - std::greater, 2508
 - std::greater_equal, 2510
 - std::hash< __gnu_cxx::throw_-value_limit >, 2520
 - std::hash< __gnu_cxx::throw_-value_random >, 2522
 - std::hash< __shared_ptr< _Tp, _Lp > >, 2525
 - std::hash< shared_ptr< _Tp > >, 2528
 - std::hash< unique_ptr< _Tp, _Dp > >, 2532
 - std::independent_bits_engine, 2535
 - std::less, 2595
 - std::less_equal, 2597
 - std::linear_congruential_engine, 2599
 - std::logical_and, 2646
 - std::logical_not, 2648

- std::logical_or, 2650
- std::lognormal_distribution, 2652
- std::mem_fun1_ref_t, 2691
- std::mem_fun1_t, 2693
- std::mem_fun_ref_t, 2695
- std::mem_fun_t, 2697
- std::minus, 2707
- std::modulus, 2709
- std::multiplies, 2768
- std::negate, 2788
- std::negative_binomial_distribution, 2790
- std::normal_distribution, 2796
- std::not_equal_to, 2802
- std::owner_less< shared_ptr< _Tp > >, 2896
- std::owner_less< weak_ptr< _Tp > >, 2898
- std::piecewise_constant_distribution, 2904
- std::piecewise_linear_distribution, 2909
- std::plus, 2914
- std::pointer_to_binary_function, 2916
- std::pointer_to_unary_function, 2918
- std::poisson_distribution, 2919
- std::random_device, 2936
- std::seed_seq, 2975
- std::shuffle_order_engine, 3013
- std::student_t_distribution, 3027
- std::unary_function, 3118
- std::unary_negate, 3120
- std::uniform_int_distribution, 3123
- std::uniform_real_distribution, 3127
- std::weibull_distribution, 3177
- rethrow_exception
 - exceptions, 36
- rethrow_if_nested
 - exceptions, 37
- return_temporary_buffer
 - std, 684
- reverse
 - mutating_algorithms, 144
 - std::forward_list, 2469
 - std::list, 2625
- reverse_copy
 - mutating_algorithms, 145
- reverse_iterator
 - std::reverse_iterator, 2968, 2969
 - std::set, 2980
- rfind
 - __gnu_cxx::__versa_string, 855–857
 - __gnu_debug::basic_string, 1099, 1100
 - std::basic_string, 2098–2100
- riemann_zeta
 - tr1_math_spec_func, 117
- right
 - std, 685
 - std::basic_fstream, 1620
 - std::basic_ifstream, 1673
 - std::basic_ios, 1704
 - std::basic_iostream, 1769
 - std::basic_istream, 1820
 - std::basic_istreamstream, 1875
 - std::basic_ofstream, 1919
 - std::basic_ostream, 1961
 - std::basic_ostreamstream, 2006
 - std::basic_stringstream, 2193
 - std::ios_base, 2568
- rope, 3484
- ropeimpl.h, 3489
- rotate
 - mutating_algorithms, 145
- rotate_copy
 - mutating_algorithms, 146
- round_indeterminate
 - std, 606
- round_to_nearest
 - std, 606
- round_toward_infinity
 - std, 606
- round_toward_neg_infinity
 - std, 606
- round_toward_zero
 - std, 606
- round_error
 - std::numeric_limits, 2840
- round_style

- std::__numeric_limits_base, [1413](#)
- std::numeric_limits, [2844](#)
- runtime_error
 - std::runtime_error, [2974](#)
- safe_base.h, [3489](#)
- safe_iterator.h, [3490](#)
- safe_iterator.tcc, [3492](#)
- safe_sequence.h, [3492](#)
- sbumpc
 - __gnu_cxx::enc_filebuf, [898](#)
 - __gnu_cxx::stdio_filebuf, [969](#)
 - __gnu_cxx::stdio_sync_filebuf, [997](#)
 - std::basic_filebuf, [1539](#)
 - std::basic_streambuf, [2033](#)
 - std::basic_stringbuf, [2115](#)
- scan_is
 - std::__ctype_abstract_base, [1332](#)
 - std::ctype, [2309](#)
 - std::ctype< char >, [2322](#)
 - std::ctype< wchar_t >, [2344](#)
 - std::ctype_byname, [2360](#)
 - std::ctype_byname< char >, [2372](#)
- scan_not
 - std::__ctype_abstract_base, [1332](#)
 - std::ctype, [2309](#)
 - std::ctype< char >, [2323](#)
 - std::ctype< wchar_t >, [2344](#)
 - std::ctype_byname, [2360](#)
 - std::ctype_byname< char >, [2373](#)
- scientific
 - std, [685](#)
 - std::basic_fstream, [1620](#)
 - std::basic_ifstream, [1673](#)
 - std::basic_ios, [1704](#)
 - std::basic_iostream, [1769](#)
 - std::basic_istream, [1820](#)
 - std::basic_istream, [1875](#)
 - std::basic_ofstream, [1919](#)
 - std::basic_ostream, [1961](#)
 - std::basic_ostream, [2006](#)
 - std::basic_stringstream, [2193](#)
 - std::ios_base, [2568](#)
- search
 - non_mutating_algorithms, [163](#), [164](#)
- search.h, [3492](#)
- search_minimal_n
 - __gnu_parallel::_Settings, [1223](#)
- search_n
 - non_mutating_algorithms, [165](#)
- second
 - __gnu_parallel::_IteratorPair, [1162](#)
 - std::pair, [2902](#)
 - std::sub_match, [3035](#)
- second_argument_type
 - __gnu_cxx::project1st, [931](#)
 - __gnu_cxx::project2nd, [932](#)
 - __gnu_parallel::_EqualFromLess, [1155](#)
 - __gnu_parallel::_EqualTo, [1157](#)
 - __gnu_parallel::_Less, [1167](#)
 - __gnu_parallel::_Lexicographic, [1169](#)
 - __gnu_parallel::_-LexicographicReverse, [1171](#)
 - __gnu_parallel::_Multiplies, [1199](#)
 - __gnu_parallel::_Plus, [1203](#)
 - std::_Maybe_unary_or_binary_-function< _Res, _T1, _T2 >, [1465](#)
- std::binary_function, [2200](#)
- std::binary_negate, [2202](#)
- std::const_mem_fun1_ref_t, [2291](#)
- std::const_mem_fun1_t, [2293](#)
- std::divides, [2427](#)
- std::equal_to, [2432](#)
- std::greater, [2508](#)
- std::greater_equal, [2510](#)
- std::less, [2595](#)
- std::less_equal, [2597](#)
- std::logical_and, [2646](#)
- std::logical_or, [2650](#)
- std::mem_fun1_ref_t, [2691](#)
- std::mem_fun1_t, [2693](#)
- std::minus, [2707](#)
- std::modulus, [2709](#)
- std::multiplies, [2768](#)
- std::not_equal_to, [2802](#)
- std::owner_less< shared_ptr< _Tp > >, [2897](#)
- std::owner_less< weak_ptr< _Tp > >, [2898](#)

- std::plus, 2914
- std::pointer_to_binary_function, 2916
- second_type
 - __gnu_parallel::_IteratorPair, 1162
 - std::pair, 2901
 - std::sub_match, 3033
- seconds
 - std::chrono, 737
- seed
 - std::discard_block_engine, 2419
 - std::independent_bits_engine, 2539
 - std::linear_congruential_engine, 2601
 - std::shuffle_order_engine, 3016, 3017
- seed_seq
 - std::seed_seq, 2975
- seekdir
 - std::basic_fstream, 1568
 - std::basic_ifstream, 1632
 - std::basic_ios, 1683
 - std::basic_iostream, 1718
 - std::basic_istream, 1780
 - std::basic_istreamstream, 1834
 - std::basic_ofstream, 1886
 - std::basic_ostream, 1930
 - std::basic_ostreamstream, 1974
 - std::basic_stringstream, 2142
 - std::ios_base, 2556
- seekg
 - std::basic_fstream, 1606
 - std::basic_ifstream, 1661, 1662
 - std::basic_iostream, 1754, 1755
 - std::basic_istream, 1807, 1808
 - std::basic_istreamstream, 1862, 1863
 - std::basic_stringstream, 2178, 2179
- seekoff
 - __gnu_cxx::enc_filebuf, 898
 - __gnu_cxx::stdio_filebuf, 970
 - __gnu_cxx::stdio_sync_filebuf, 998
 - std::basic_filebuf, 1540
 - std::basic_streambuf, 2033
 - std::basic_stringbuf, 2116
- seekp
 - std::basic_fstream, 1607
 - std::basic_iostream, 1756
 - std::basic_ofstream, 1908
 - std::basic_ostream, 1950
 - std::basic_ostreamstream, 1994, 1995
 - std::basic_stringstream, 2180
- seekpos
 - __gnu_cxx::enc_filebuf, 898
 - __gnu_cxx::stdio_filebuf, 970
 - __gnu_cxx::stdio_sync_filebuf, 998
 - std::basic_filebuf, 1540
 - std::basic_streambuf, 2033
 - std::basic_stringbuf, 2116
- sentry
 - std::basic_istream::sentry, 1822
 - std::basic_ostream::sentry, 1963
- Sequences, 27
- sequential
 - __gnu_parallel, 407
- set, 3493
 - __gnu_parallel::_Settings, 1218
 - std::bitset, 2222
 - std::set, 2981–2983
- Set Operation, 189
- set.h, 3494, 3495
- set_algorithms
 - includes, 190
 - set_difference, 191, 192
 - set_intersection, 192, 193
 - set_symmetric_difference, 194
 - set_union, 195
- set_difference
 - set_algorithms, 191, 192
- set_difference_minimal_n
 - __gnu_parallel::_Settings, 1223
- set_intersection
 - set_algorithms, 192, 193
- set_intersection_minimal_n
 - __gnu_parallel::_Settings, 1223
- set_new_handler
 - std, 685
- set_num_threads
 - __gnu_parallel::balanced_ - quicksort_tag, 1228
 - __gnu_parallel::balanced_tag, 1230
 - __gnu_parallel::default_parallel_ - tag, 1232

- __gnu_parallel::exact_tag, 1235
- __gnu_parallel::multiway_-mergesort_exact_tag, 1238
- __gnu_parallel::multiway_-mergesort_sampling_tag, 1240
- __gnu_parallel::multiway_-mergesort_tag, 1241
- __gnu_parallel::omp_loop_static_tag, 1243
- __gnu_parallel::omp_loop_tag, 1244
- __gnu_parallel::parallel_tag, 1247
- __gnu_parallel::quicksort_tag, 1249
- __gnu_parallel::sampling_tag, 1250
- __gnu_parallel::unbalanced_tag, 1252
- set_operations.h, 3496
- set_symmetric_difference
 - set_algorithms, 194
- set_symmetric_difference_minimal_n
 - __gnu_parallel::_Settings, 1224
- set_terminate
 - exceptions, 37
- set_unexpected
 - exceptions, 37
- set_union
 - set_algorithms, 195
- set_union_minimal_n
 - __gnu_parallel::_Settings, 1224
- setbase
 - std, 685
- setbuf
 - __gnu_cxx::enc_filebuf, 899
 - __gnu_cxx::stdio_filebuf, 971
 - __gnu_cxx::stdio_sync_filebuf, 999
 - std::basic_filebuf, 1540, 1541
 - std::basic_streambuf, 2034
 - std::basic_stringbuf, 2117
- setf
 - std::basic_fstream, 1608
 - std::basic_ifstream, 1662, 1663
 - std::basic_ios, 1695
 - std::basic_iostream, 1757
 - std::basic_istream, 1809
 - std::basic_istream, 1863, 1864
- std::basic_ofstream, 1909
- std::basic_ostream, 1951
- std::basic_ostringstream, 1995, 1996
- std::basic_stringstream, 2181
- std::ios_base, 2561
- setfill
 - std, 686
- setg
 - __gnu_cxx::enc_filebuf, 899
 - __gnu_cxx::stdio_filebuf, 971
 - __gnu_cxx::stdio_sync_filebuf, 999
 - std::basic_filebuf, 1541
 - std::basic_streambuf, 2034
 - std::basic_stringbuf, 2117
- setiosflags
 - std, 686
- setp
 - __gnu_cxx::enc_filebuf, 900
 - __gnu_cxx::stdio_filebuf, 972
 - __gnu_cxx::stdio_sync_filebuf, 999
 - std::basic_filebuf, 1542
 - std::basic_streambuf, 2035
 - std::basic_stringbuf, 2118
- setprecision
 - std, 686
- setstate
 - std::basic_fstream, 1609
 - std::basic_ifstream, 1663
 - std::basic_ios, 1696
 - std::basic_iostream, 1757
 - std::basic_istream, 1810
 - std::basic_istream, 1864
 - std::basic_ofstream, 1909
 - std::basic_ostream, 1952
 - std::basic_ostringstream, 1996
 - std::basic_stringstream, 2181
- settings.h, 3497
 - _GLIBCXX_PARALLEL_CONDITION, 3498
- setw
 - std, 686
- sgetc
 - __gnu_cxx::enc_filebuf, 900
 - __gnu_cxx::stdio_filebuf, 972
 - __gnu_cxx::stdio_sync_filebuf, 1000

- std::basic_filebuf, 1542
- std::basic_streambuf, 2035
- std::basic_stringbuf, 2118
- sgetn
 - __gnu_cxx::enc_filebuf, 900
 - __gnu_cxx::stdio_filebuf, 973
 - __gnu_cxx::stdio_sync_filebuf, 1000
 - std::basic_filebuf, 1542
 - std::basic_streambuf, 2035
 - std::basic_stringbuf, 2119
- SGI, 12
- SGIextensions
 - _Find_first, 17
 - _Find_next, 17
 - _Unchecked_flip, 18
 - _Unchecked_reset, 18
 - _Unchecked_set, 18
 - _Unchecked_test, 18
 - __median, 16, 17
 - compose1, 19
 - compose2, 19
 - constant0, 19
 - constant1, 19
 - constant2, 19
 - copy_n, 20
 - distance, 20
 - identity_element, 20, 21
 - iota, 21
 - is_heap, 21
 - is_sorted, 22
 - lexicographical_compare_3way, 22
 - power, 23
 - random_sample, 23, 24
 - random_sample_n, 24
 - uninitialized_copy_n, 25
- shared_future
 - std::shared_future, 2997
 - std::shared_future< _Res & >, 3000
 - std::shared_future< void >, 3003
- shared_ptr
 - std::shared_ptr, 3006–3011
- shared_ptr.h, 3498
- shared_ptr_base.h, 3500
- shift
 - numeric_arrays, 108
- showbase
 - std, 687
 - std::basic_fstream, 1620
 - std::basic_ifstream, 1673
 - std::basic_ios, 1704
 - std::basic_iostream, 1769
 - std::basic_istream, 1820
 - std::basic_istreamstream, 1875
 - std::basic_ofstream, 1919
 - std::basic_ostream, 1961
 - std::basic_ostreamstream, 2006
 - std::basic_stringstream, 2193
 - std::ios_base, 2568
- showmanyc
 - __gnu_cxx::enc_filebuf, 901
 - __gnu_cxx::stdio_filebuf, 973
 - __gnu_cxx::stdio_sync_filebuf, 1000
 - std::basic_filebuf, 1543
 - std::basic_streambuf, 2036
 - std::basic_stringbuf, 2119
- showpoint
 - std, 687
 - std::basic_fstream, 1620
 - std::basic_ifstream, 1673
 - std::basic_ios, 1704
 - std::basic_iostream, 1769
 - std::basic_istream, 1820
 - std::basic_istreamstream, 1875
 - std::basic_ofstream, 1919
 - std::basic_ostream, 1961
 - std::basic_ostreamstream, 2007
 - std::basic_stringstream, 2193
 - std::ios_base, 2568
- showpos
 - std, 687
 - std::basic_fstream, 1620
 - std::basic_ifstream, 1674
 - std::basic_ios, 1704
 - std::basic_iostream, 1769
 - std::basic_istream, 1820
 - std::basic_istreamstream, 1875
 - std::basic_ofstream, 1919
 - std::basic_ostream, 1962
 - std::basic_ostreamstream, 2007
 - std::basic_stringstream, 2194

-
- std::ios_base, 2568
 - shrink_to_fit
 - __gnu_cxx::__versa_string, 858
 - __gnu_debug::basic_string, 1100
 - std::basic_string, 2100
 - std::deque, 2413
 - std::vector, 3170
 - shuffle
 - mutating_algorithms, 146
 - shuffle_order_engine
 - std::shuffle_order_engine, 3014, 3015
 - signaling_NaN
 - std::numeric_limits, 2840
 - sin
 - complex_numbers, 53
 - sinh
 - complex_numbers, 53
 - size
 - __gnu_cxx::__versa_string, 858
 - __gnu_cxx::temporary_buffer, 1013
 - __gnu_debug::basic_string, 1100
 - numeric_arrays, 109
 - std::_Temporary_buffer, 1480
 - std::basic_string, 2101
 - std::bitset, 2222
 - std::deque, 2413
 - std::list, 2625
 - std::map, 2674
 - std::match_results, 2688
 - std::multimap, 2765
 - std::multiset, 2783
 - std::priority_queue, 2927
 - std::queue, 2933
 - std::set, 2994
 - std::stack, 3025
 - std::vector, 3170
 - size_type
 - std::set, 2981
 - skipws
 - std, 687
 - std::basic_fstream, 1621
 - std::basic_ifstream, 1674
 - std::basic_ios, 1705
 - std::basic_iostream, 1769
 - std::basic_istream, 1820
 - std::basic_istream, 1875
 - std::basic_ofstream, 1920
 - std::basic_ostream, 1962
 - std::basic_ostringstream, 2007
 - std::basic_stringstream, 2194
 - std::ios_base, 2569
 - sleep_for
 - std::this_thread, 762
 - sleep_until
 - std::this_thread, 762
 - slice
 - numeric_arrays, 89
 - slice_array
 - numeric_arrays, 89
 - slice_array.h, 3502
 - slist, 3503
 - snextc
 - __gnu_cxx::enc_filebuf, 901
 - __gnu_cxx::stdio_filebuf, 974
 - __gnu_cxx::stdio_sync_filebuf, 1001
 - std::basic_filebuf, 1543
 - std::basic_streambuf, 2036
 - std::basic_stringbuf, 2120
 - sort
 - sorting_algorithms, 186, 187
 - std::forward_list, 2469
 - std::list, 2625, 2626
 - sort.h, 3504
 - sort_heap
 - heap_algorithms, 269
 - sort_minimal_n
 - __gnu_parallel::_Settings, 1224
 - sort_mwms_oversampling
 - __gnu_parallel::_Settings, 1224
 - sort_qs_num_samples_preset
 - __gnu_parallel::_Settings, 1224
 - sort_qsb_base_case_maximal_n
 - __gnu_parallel::_Settings, 1224
 - Sorting, 167
 - sorting_algorithms
 - inplace_merge, 170
 - is_sorted, 171
 - is_sorted_until, 172
 - lexicographical_compare, 173
 - max, 174
-

- max_element, 175
- merge, 176
- min, 177
- min_element, 178
- minmax, 179
- minmax_element, 179, 180
- next_permutation, 180, 181
- nth_element, 181, 182
- partial_sort, 183
- partial_sort_copy, 184
- prev_permutation, 185, 186
- sort, 186, 187
- stable_sort, 187, 188
- sph_bessel
 - tr1_math_spec_func, 117
- sph_legendre
 - tr1_math_spec_func, 117
- sph_neumann
 - tr1_math_spec_func, 117
- splice
 - std::list, 2626, 2627
- splice_after
 - std::forward_list, 2469, 2470
- sputbackc
 - __gnu_cxx::enc_filebuf, 902
 - __gnu_cxx::stdio_filebuf, 974
 - __gnu_cxx::stdio_sync_filebuf, 1001
 - std::basic_filebuf, 1544
 - std::basic_streambuf, 2037
 - std::basic_stringbuf, 2120
- sputc
 - __gnu_cxx::enc_filebuf, 902
 - __gnu_cxx::stdio_filebuf, 974
 - __gnu_cxx::stdio_sync_filebuf, 1002
 - std::basic_filebuf, 1544
 - std::basic_streambuf, 2037
 - std::basic_stringbuf, 2120
- sputn
 - __gnu_cxx::enc_filebuf, 902
 - __gnu_cxx::stdio_filebuf, 975
 - __gnu_cxx::stdio_sync_filebuf, 1002
 - std::basic_filebuf, 1545
 - std::basic_streambuf, 2038
 - std::basic_stringbuf, 2121
- sqrt
 - complex_numbers, 53
- sregex_token_iterator
 - regex, 224
- sso_string_base.h, 3505
- sstream, 3505
- sstream.tcc, 3506
- ssub_match
 - regex, 224
- stable_partition
 - mutating_algorithms, 147
- stable_sort
 - sorting_algorithms, 187, 188
- stack, 3507
 - std::stack, 3024
- standard_policies.hpp, 3507
- start
 - numeric_arrays, 109
- state
 - std::fpos, 2474
- static_pointer_cast
 - std, 687
- std, 464
 - _Construct, 619
 - _Destroy, 620
 - __final_insertion_sort, 606
 - __find, 606, 607
 - __find_if, 607
 - __find_if_not, 607, 608
 - __gcd, 608
 - __heap_select, 608
 - __inplace_stable_partition, 608
 - __inplace_stable_sort, 609
 - __insertion_sort, 609
 - __introsort_loop, 610
 - __invoke, 694
 - __lg, 610
 - __merge_adaptive, 610, 611
 - __merge_backward, 611
 - __merge_without_buffer, 612
 - __move_median_first, 612, 613
 - __partition, 613
 - __reverse, 613
 - __rotate, 614
 - __rotate_adaptive, 614

- [__search_n](#), 615
- [__stable_partition_adaptive](#), 616
- [__unguarded_insertion_sort](#), 616
- [__unguarded_linear_insert](#), 616, 617
- [__unguarded_partition](#), 617
- [__unguarded_partition_pivot](#), 617, 618
- [__unique_copy](#), 618, 619
- [accumulate](#), 620, 621
- [addressof](#), 621
- [adjacent_difference](#), 621, 622
- [advance](#), 622
- [begin](#), 623, 624
- [bind](#), 624
- [boolalpha](#), 624
- [cerr](#), 694
- [cin](#), 694
- [clog](#), 694
- [const_pointer_cast](#), 625
- [cout](#), 694
- [cref](#), 625
- [dec](#), 625
- [denorm_absent](#), 605
- [denorm_indeterminate](#), 605
- [denorm_present](#), 606
- [distance](#), 625
- [dynamic_pointer_cast](#), 626
- [end](#), 626, 627
- [endl](#), 627
- [ends](#), 628
- [fixed](#), 628
- [float_denorm_style](#), 605
- [float_round_style](#), 606
- [flush](#), 628
- [forward](#), 628, 629
- [get_money](#), 629
- [get_temporary_buffer](#), 629
- [getline](#), 630, 631
- [has_facet](#), 632
- [hex](#), 632
- [inner_product](#), 633
- [internal](#), 634
- [iota](#), 634
- [isalnum](#), 634
- [isalpha](#), 634
- [isctrl](#), 635
- [isdigit](#), 635
- [isgraph](#), 635
- [islower](#), 635
- [isprint](#), 635
- [ispunct](#), 635
- [isspace](#), 635
- [isupper](#), 635
- [isxdigit](#), 636
- [left](#), 636
- [make_pair](#), 636
- [new_handler](#), 604
- [noboolalpha](#), 637
- [noshowbase](#), 637
- [noshowpoint](#), 637
- [noshowpos](#), 637
- [noskipws](#), 637
- [nunitbuf](#), 637
- [nouppercase](#), 638
- [oct](#), 638
- [operator<](#), 644–650
- [operator<<](#), 650–657
- [operator<=](#), 657–661
- [operator>](#), 668–671
- [operator>>](#), 674–680
- [operator>=](#), 671–674
- [operator^](#), 681
- [operator+](#), 642–644
- [operator==](#), 661–667
- [operator&](#), 642
- [partial_sum](#), 682
- [put_money](#), 683
- [ref](#), 683
- [replace_copy](#), 684
- [resetiosflags](#), 684
- [return_temporary_buffer](#), 684
- [right](#), 685
- [round_indeterminate](#), 606
- [round_to_nearest](#), 606
- [round_toward_infinity](#), 606
- [round_toward_neg_infinity](#), 606
- [round_toward_zero](#), 606
- [scientific](#), 685
- [set_new_handler](#), 685
- [setbase](#), 685
- [setfill](#), 686
- [setiosflags](#), 686

- setprecision, 686
- setw, 686
- showbase, 687
- showpoint, 687
- showpos, 687
- skipws, 687
- static_pointer_cast, 687
- streamoff, 604
- streampos, 604
- streamsize, 604
- swap, 688–690
- tolower, 690
- toupper, 690
- u16streampos, 605
- u32streampos, 605
- uninitialized_copy, 691
- uninitialized_copy_n, 691
- uninitialized_fill, 691
- uninitialized_fill_n, 692
- unitbuf, 692
- uppercase, 692
- use_facet, 693
- wcerr, 694
- wcin, 694
- wclog, 695
- wcout, 695
- ws, 693
- wstreampos, 605
- std::__basic_future, 1314
 - _M_get_result, 1315
- std::__codecvt_abstract_base, 1315
 - do_out, 1317
 - in, 1318
 - out, 1319
 - unshift, 1320
- std::__ctype_abstract_base, 1321
 - char_type, 1323
 - do_is, 1324
 - do_narrow, 1324, 1325
 - do_scan_is, 1326
 - do_scan_not, 1326
 - do_tolower, 1327
 - do_toupper, 1328
 - do_widen, 1329
 - is, 1330
 - narrow, 1331
 - scan_is, 1332
 - scan_not, 1332
 - tolower, 1333
 - toupper, 1333, 1334
 - widen, 1334, 1335
- std::__debug, 695
- std::__debug::bitset, 1336
 - _M_const_iterators, 1339
 - _M_detach_all, 1338
 - _M_detach_singular, 1338
 - _M_get_mutex, 1338
 - _M_invalidate_all, 1338
 - _M_iterators, 1339
 - _M_revalidate_singular, 1339
 - _M_swap, 1339
 - _M_version, 1339
- std::__debug::deque, 1340
 - _M_const_iterators, 1345
 - _M_detach_all, 1343
 - _M_detach_singular, 1343
 - _M_get_mutex, 1343
 - _M_invalidate_all, 1344
 - _M_invalidate_if, 1344
 - _M_iterators, 1345
 - _M_revalidate_singular, 1344
 - _M_swap, 1344
 - _M_transfer_iter, 1344
 - _M_version, 1345
- std::__debug::list, 1345
 - _M_const_iterators, 1350
 - _M_detach_all, 1349
 - _M_detach_singular, 1349
 - _M_get_mutex, 1349
 - _M_invalidate_all, 1349
 - _M_invalidate_if, 1350
 - _M_iterators, 1351
 - _M_revalidate_singular, 1350
 - _M_swap, 1350
 - _M_transfer_iter, 1350
 - _M_version, 1351
- std::__debug::map, 1351
 - _M_const_iterators, 1356
 - _M_detach_all, 1354
 - _M_detach_singular, 1354
 - _M_get_mutex, 1355
 - _M_invalidate_all, 1355

- [_M_invalidate_if, 1355](#)
 - [_M_iterators, 1356](#)
 - [_M_revalidate_singular, 1355](#)
 - [_M_swap, 1355](#)
 - [_M_transfer_iter, 1356](#)
 - [_M_version, 1356](#)
- [std::__debug::multimap, 1357](#)
 - [_M_const_iterators, 1361](#)
 - [_M_detach_all, 1360](#)
 - [_M_detach_singular, 1360](#)
 - [_M_get_mutex, 1360](#)
 - [_M_invalidate_all, 1360](#)
 - [_M_invalidate_if, 1360](#)
 - [_M_iterators, 1361](#)
 - [_M_revalidate_singular, 1360](#)
 - [_M_swap, 1361](#)
 - [_M_transfer_iter, 1361](#)
 - [_M_version, 1361](#)
- [std::__debug::multiset, 1362](#)
 - [_M_const_iterators, 1366](#)
 - [_M_detach_all, 1365](#)
 - [_M_detach_singular, 1365](#)
 - [_M_get_mutex, 1365](#)
 - [_M_invalidate_all, 1365](#)
 - [_M_invalidate_if, 1365](#)
 - [_M_iterators, 1366](#)
 - [_M_revalidate_singular, 1366](#)
 - [_M_swap, 1366](#)
 - [_M_transfer_iter, 1366](#)
 - [_M_version, 1367](#)
- [std::__debug::set, 1367](#)
 - [_M_const_iterators, 1372](#)
 - [_M_detach_all, 1370](#)
 - [_M_detach_singular, 1370](#)
 - [_M_get_mutex, 1371](#)
 - [_M_invalidate_all, 1371](#)
 - [_M_invalidate_if, 1371](#)
 - [_M_iterators, 1372](#)
 - [_M_revalidate_singular, 1371](#)
 - [_M_swap, 1371](#)
 - [_M_transfer_iter, 1372](#)
 - [_M_version, 1372](#)
- [std::__debug::unordered_map, 1373](#)
 - [_M_const_iterators, 1377](#)
 - [_M_detach_all, 1375](#)
 - [_M_detach_singular, 1375](#)
 - [_M_get_mutex, 1375](#)
 - [_M_invalidate_all, 1376](#)
 - [_M_invalidate_if, 1376](#)
 - [_M_iterators, 1377](#)
 - [_M_revalidate_singular, 1376](#)
 - [_M_swap, 1376](#)
 - [_M_transfer_iter, 1376](#)
 - [_M_version, 1377](#)
- [std::__debug::unordered_multimap, 1378](#)
 - [_M_const_iterators, 1381](#)
 - [_M_detach_all, 1380](#)
 - [_M_detach_singular, 1380](#)
 - [_M_get_mutex, 1380](#)
 - [_M_invalidate_all, 1380](#)
 - [_M_invalidate_if, 1380](#)
 - [_M_iterators, 1381](#)
 - [_M_revalidate_singular, 1381](#)
 - [_M_swap, 1381](#)
 - [_M_transfer_iter, 1381](#)
 - [_M_version, 1382](#)
- [std::__debug::unordered_multiset, 1382](#)
 - [_M_const_iterators, 1386](#)
 - [_M_detach_all, 1384](#)
 - [_M_detach_singular, 1384](#)
 - [_M_get_mutex, 1385](#)
 - [_M_invalidate_all, 1385](#)
 - [_M_invalidate_if, 1385](#)
 - [_M_iterators, 1386](#)
 - [_M_revalidate_singular, 1385](#)
 - [_M_swap, 1385](#)
 - [_M_transfer_iter, 1385](#)
 - [_M_version, 1386](#)
- [std::__debug::unordered_set, 1387](#)
 - [_M_const_iterators, 1391](#)
 - [_M_detach_all, 1389](#)
 - [_M_detach_singular, 1389](#)
 - [_M_get_mutex, 1389](#)
 - [_M_invalidate_all, 1390](#)
 - [_M_invalidate_if, 1390](#)
 - [_M_iterators, 1391](#)
 - [_M_revalidate_singular, 1390](#)
 - [_M_swap, 1390](#)
 - [_M_transfer_iter, 1390](#)
 - [_M_version, 1391](#)
- [std::__debug::vector, 1392](#)
 - [_M_const_iterators, 1396](#)

- [_M_detach_all, 1395](#)
- [_M_detach_singular, 1395](#)
- [_M_get_mutex, 1395](#)
- [_M_invalidate_all, 1395](#)
- [_M_invalidate_if, 1396](#)
- [_M_iterators, 1397](#)
- [_M_revalidate_singular, 1396](#)
- [_M_swap, 1396](#)
- [_M_transfer_iter, 1396](#)
- [_M_version, 1397](#)
- [vector, 1395](#)
- [std::__declval_protector, 1397](#)
- [std::__detail, 701](#)
- [std::__exception_ptr::exception_ptr, 1398](#)
- [std::__future_base, 1399](#)
- [std::__future_base::_Ptr, 1400](#)
- [std::__future_base::_Result, 1401](#)
- [std::__future_base::_Result< _Res & >, 1402](#)
- [std::__future_base::_Result< void >, 1403](#)
- [std::__future_base::_Result_alloc, 1404](#)
- [std::__future_base::_Result_base, 1405](#)
- [std::__future_base::_State, 1406](#)
- [std::__is_location_invariant, 1407](#)
- [std::__iterator_traits, 1408](#)
- [std::__numeric_limits_base, 1408](#)
- [digits, 1410](#)
- [digits10, 1410](#)
- [has_denorm, 1410](#)
- [has_denorm_loss, 1410](#)
- [has_infinity, 1410](#)
- [has_quiet_NaN, 1411](#)
- [has_signaling_NaN, 1411](#)
- [is_bounded, 1411](#)
- [is_exact, 1411](#)
- [is_iec559, 1411](#)
- [is_integer, 1411](#)
- [is_modulo, 1412](#)
- [is_signed, 1412](#)
- [is_specialized, 1412](#)
- [max_digits10, 1412](#)
- [max_exponent, 1412](#)
- [max_exponent10, 1412](#)
- [min_exponent, 1412](#)
- [min_exponent10, 1413](#)
- [radix, 1413](#)
- [round_style, 1413](#)
- [tinyness_before, 1413](#)
- [traps, 1413](#)
- [std::__parallel, 702](#)
- [std::__parallel::_CRandNumber, 1414](#)
- [std::__profile, 727](#)
- [std::__profile::bitset, 1414](#)
- [std::__profile::deque, 1416](#)
- [std::__profile::list, 1418](#)
- [std::__profile::map, 1421](#)
- [std::__profile::multimap, 1423](#)
- [std::__profile::multiset, 1425](#)
- [std::__profile::set, 1427](#)
- [std::__profile::unordered_map, 1429](#)
- [std::__profile::unordered_multimap, 1431](#)
- [std::__profile::unordered_multiset, 1432](#)
- [std::__profile::unordered_set, 1434](#)
- [std::_Base_bitset, 1435](#)
- [_M_w, 1437](#)
- [std::_Base_bitset< 0 >, 1437](#)
- [std::_Base_bitset< 1 >, 1438](#)
- [std::_Build_index_tuple, 1440](#)
- [std::_Deque_base, 1440](#)
- [_M_initialize_map, 1442](#)
- [std::_Deque_iterator, 1442](#)
- [_M_set_node, 1444](#)
- [std::_Derives_from_binary_function, 1445](#)
- [std::_Derives_from_unary_function, 1445](#)
- [std::_Function_base, 1446](#)
- [std::_Function_to_function_pointer, 1447](#)
- [std::_Fwd_list_base, 1447](#)
- [std::_Fwd_list_const_iterator, 1449](#)
- [std::_Fwd_list_iterator, 1450](#)
- [std::_Fwd_list_node, 1451](#)
- [std::_Fwd_list_node_base, 1453](#)
- [std::_Index_tuple, 1454](#)
- [std::_List_base, 1454](#)
- [std::_List_const_iterator, 1456](#)
- [std::_List_iterator, 1457](#)
- [std::_List_node, 1458](#)
- [_M_data, 1460](#)
- [std::_List_node_base, 1460](#)
- [std::_Maybe_get_result_type, 1461](#)

-
- std::_Maybe_unary_or_binary_function, 1462
 std::_Maybe_unary_or_binary_function< _Res, _T1 >, 1462
 argument_type, 1463
 result_type, 1464
 std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >, 1464
 first_argument_type, 1465
 result_type, 1465
 second_argument_type, 1465
 std::_Maybe_wrap_member_pointer, 1466
 std::_Maybe_wrap_member_pointer< _Tp _Class::*, >, 1466
 std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const >, 1467
 std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) const volatile >, 1468
 std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) volatile >, 1469
 std::_Mem_fn< _Res(_Class::*)(_ArgTypes...) >, 1470
 std::_Mu< _Arg, false, false >, 1472
 std::_Mu< _Arg, false, true >, 1472
 std::_Mu< _Arg, true, false >, 1473
 std::_Mu< reference_wrapper< _Tp >, false, false >, 1473
 std::_Placeholder, 1474
 std::_Reference_wrapper_base, 1475
 std::_Safe_tuple_element, 1476
 std::_Safe_tuple_element_impl, 1476
 std::_Safe_tuple_element_impl< __i, _Tuple, false >, 1477
 std::_Temporary_buffer, 1478
 _Temporary_buffer, 1479
 begin, 1479
 end, 1479
 requested_size, 1479
 size, 1480
 std::_Tuple_impl< _Idx >, 1480
 std::_Tuple_impl< _Idx, _Head, _Tail..., >, 1481
 std::_Vector_base, 1482
 std::_Weak_result_type, 1483
 std::_Weak_result_type_impl, 1484
 std::_Weak_result_type_impl< _Res(*)(_ArgTypes...) >, 1485
 std::_Weak_result_type_impl< _Res(&)(_ArgTypes...) >, 1484
 std::_Weak_result_type_impl< _Res(_ArgTypes...) >, 1485
 std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >, 1486
 std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile >, 1486
 std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) volatile >, 1487
 std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) >, 1488
 std::add_lvalue_reference, 1488
 std::add_rvalue_reference, 1489
 std::adopt_lock_t, 1489
 std::aligned_storage, 1490
 std::allocator, 1490
 std::allocator< void >, 1492
 std::allocator_arg_t, 1493
 std::atomic, 1493
 std::atomic< _Tp * >, 1494
 std::atomic< bool >, 1495
 std::atomic< char >, 1496
 std::atomic< char16_t >, 1496
 std::atomic< char32_t >, 1497
 std::atomic< int >, 1498
 std::atomic< long >, 1499
 std::atomic< long long >, 1500
 std::atomic< short >, 1500
 std::atomic< signed char >, 1501
 std::atomic< unsigned char >, 1502
 std::atomic< unsigned int >, 1503
 std::atomic< unsigned long >, 1504
 std::atomic< unsigned long long >, 1505
 std::atomic< unsigned short >, 1505
 std::atomic< void * >, 1506
 std::atomic< wchar_t >, 1507
-

- std::auto_ptr, 1508
 - ~auto_ptr, 1510
 - auto_ptr, 1509, 1510
 - element_type, 1509
 - get, 1511
 - operator*, 1511
 - operator->, 1511
 - operator=, 1512
 - release, 1512
 - reset, 1513
- std::auto_ptr_ref, 1513
- std::back_insert_iterator, 1514
 - back_insert_iterator, 1516
 - container_type, 1515
 - difference_type, 1515
 - iterator_category, 1516
 - operator*, 1517
 - operator++, 1517
 - operator=, 1517
 - pointer, 1516
 - reference, 1516
 - value_type, 1516
- std::bad_alloc, 1518
 - what, 1518
- std::bad_cast, 1519
 - what, 1520
- std::bad_exception, 1520
 - what, 1521
- std::bad_function_call, 1521
 - what, 1522
- std::bad_typeid, 1522
 - what, 1523
- std::basic_filebuf, 1523
 - ~basic_filebuf, 1529
 - _M_buf, 1549
 - _M_buf_locale, 1549
 - _M_buf_size, 1549
 - _M_create_pback, 1529
 - _M_destroy_pback, 1529
 - _M_ext_buf, 1549
 - _M_ext_buf_size, 1550
 - _M_ext_next, 1550
 - _M_in_beg, 1550
 - _M_in_cur, 1550
 - _M_in_end, 1551
 - _M_mode, 1551
 - _M_out_beg, 1551
 - _M_out_cur, 1552
 - _M_out_end, 1552
 - _M_pback, 1553
 - _M_pback_cur_save, 1553
 - _M_pback_end_save, 1553
 - _M_pback_init, 1553
 - _M_reading, 1554
 - _M_set_buffer, 1530
 - __streambuf_type, 1527
- basic_filebuf, 1529
- char_type, 1527
- close, 1530
- eback, 1530
- egptr, 1531
- epptr, 1531
- gbump, 1532
- getloc, 1532
- gptr, 1532
- imbue, 1533
- in_avail, 1533
- int_type, 1528
- is_open, 1534
- off_type, 1528
- open, 1534, 1535
- overflow, 1535
- pbackfail, 1536
- pbase, 1536
- pbump, 1537
- pos_type, 1528
- pptr, 1537
- pubimbue, 1537
- pubseekoff, 1538
- pubseekpos, 1538
- pubsetbuf, 1539
- pubsync, 1539
- sbumpc, 1539
- seekoff, 1540
- seekpos, 1540
- setbuf, 1540, 1541
- setg, 1541
- setp, 1542
- sgetc, 1542
- sgetn, 1542
- showmanyc, 1543
- sngetc, 1543

- sputbackc, 1544
- sputc, 1544
- sputn, 1545
- stossc, 1545
- sungetc, 1545
- sync, 1546
- traits_type, 1528
- uflow, 1546
- underflow, 1547
- xsgetn, 1547
- xspn, 1548
- std::basic_fstream, 1554
 - ~basic_fstream, 1570
 - _M_gcount, 1615
 - _M_getloc, 1571
 - _M_write, 1571
 - __ctype_type, 1563
 - __num_get_type, 1563
 - __num_put_type, 1563
 - adjustfield, 1615
 - app, 1615
 - ate, 1615
 - bad, 1571
 - badbit, 1616
 - basefield, 1616
 - basic_fstream, 1569, 1570
 - beg, 1616
 - binary, 1616
 - boolalpha, 1617
 - char_type, 1564
 - clear, 1572
 - close, 1572
 - copyfmt, 1572
 - cur, 1617
 - dec, 1617
 - end, 1617
 - eof, 1573
 - eofbit, 1617
 - event, 1569
 - event_callback, 1564
 - exceptions, 1573
 - fail, 1574
 - failbit, 1618
 - fill, 1575
 - fixed, 1618
 - flags, 1575, 1576
 - floatfield, 1618
 - flush, 1576
 - fmtflags, 1565
 - gcount, 1576
 - get, 1577–1580
 - getline, 1580
 - getloc, 1581
 - good, 1582
 - goodbit, 1618
 - hex, 1619
 - ignore, 1582, 1583
 - imbue, 1584
 - in, 1619
 - init, 1584
 - int_type, 1565, 1566
 - internal, 1619
 - iostate, 1566
 - is_open, 1584
 - iword, 1585
 - left, 1619
 - narrow, 1585
 - oct, 1619
 - off_type, 1566, 1567
 - open, 1586
 - openmode, 1567
 - operator void *, 1586
 - operator<<, 1587–1593
 - operator>>, 1594–1600
 - out, 1620
 - peek, 1600
 - pos_type, 1567, 1568
 - precision, 1600, 1601
 - put, 1601
 - putback, 1602
 - pword, 1602
 - rdbuf, 1603
 - rdstate, 1604
 - read, 1604
 - readsome, 1605
 - register_callback, 1605
 - right, 1620
 - scientific, 1620
 - seekdir, 1568
 - seekg, 1606
 - seekp, 1607
 - setf, 1608

- setstate, 1609
- showbase, 1620
- showpoint, 1620
- showpos, 1620
- skipws, 1621
- sync, 1609
- sync_with_stdio, 1610
- tellg, 1610
- tellp, 1611
- tie, 1611
- traits_type, 1568, 1569
- trunc, 1621
- unget, 1612
- unitbuf, 1621
- unsetf, 1612
- uppercase, 1621
- widen, 1613
- width, 1613
- write, 1614
- xalloc, 1614
- std::basic_ifstream, 1621
 - ~basic_ifstream, 1633
 - _M_gcount, 1668
 - _M_getloc, 1634
 - __ctype_type, 1628
 - __num_get_type, 1628
 - __num_put_type, 1628
- adjustfield, 1668
- app, 1669
- ate, 1669
- bad, 1634
- badbit, 1669
- basefield, 1669
- basic_ifstream, 1633
- beg, 1670
- binary, 1670
- boolalpha, 1670
- char_type, 1629
- clear, 1634
- close, 1635
- copyfmt, 1635
- cur, 1670
- dec, 1670
- end, 1670
- eof, 1636
- eofbit, 1671
- event, 1632
- event_callback, 1629
- exceptions, 1636
- fail, 1637
- failbit, 1671
- fill, 1637, 1638
- fixed, 1671
- flags, 1638
- floatfield, 1671
- fmtflags, 1629
- gcount, 1639
- get, 1639–1642
- getline, 1642, 1643
- getloc, 1644
- good, 1644
- goodbit, 1671
- hex, 1672
- ignore, 1644–1646
- imbue, 1646
- in, 1672
- init, 1647
- int_type, 1630
- internal, 1672
- iostate, 1630
- is_open, 1647
- iword, 1647
- left, 1672
- narrow, 1648
- oct, 1673
- off_type, 1631
- open, 1648
- openmode, 1631
- operator void *, 1649
- operator>>, 1649–1656
- out, 1673
- peek, 1656
- pos_type, 1631
- precision, 1656, 1657
- putback, 1657
- pword, 1658
- rdbuf, 1658
- rdstate, 1659
- read, 1659
- readsome, 1660
- register_callback, 1661
- right, 1673

- scientific, 1673
- seekdir, 1632
- seekg, 1661, 1662
- setf, 1662, 1663
- setstate, 1663
- showbase, 1673
- showpoint, 1673
- showpos, 1674
- skipws, 1674
- sync, 1664
- sync_with_stdio, 1664
- tellg, 1665
- tie, 1665
- traits_type, 1632
- trunc, 1674
- unget, 1666
- unitbuf, 1674
- unsetf, 1666
- uppercase, 1674
- widen, 1667
- width, 1667
- xalloc, 1668
- std::basic_ios, 1675
 - ~basic_ios, 1684
 - _M_getloc, 1685
 - __ctype_type, 1679
 - __num_get_type, 1679
 - __num_put_type, 1679
- adjustfield, 1699
- app, 1699
- ate, 1699
- bad, 1685
- badbit, 1700
- basefield, 1700
- basic_ios, 1684
- beg, 1700
- binary, 1700
- boolalpha, 1701
- char_type, 1679
- clear, 1685
- copyfmt, 1685
- cur, 1701
- dec, 1701
- end, 1701
- eof, 1686
- eofbit, 1701
- event, 1684
- event_callback, 1680
- exceptions, 1686, 1687
- fail, 1687
- failbit, 1702
- fill, 1688
- fixed, 1702
- flags, 1688, 1689
- floatfield, 1702
- fmtflags, 1680
- getloc, 1689
- good, 1689
- goodbit, 1702
- hex, 1703
- imbue, 1690
- in, 1703
- init, 1690
- int_type, 1681
- internal, 1703
- iostate, 1681
- isword, 1690
- left, 1703
- narrow, 1691
- oct, 1703
- off_type, 1682
- openmode, 1682
- operator void *, 1691
- out, 1704
- pos_type, 1682
- precision, 1692
- pword, 1693
- rdbuf, 1693, 1694
- rdstate, 1694
- register_callback, 1694
- right, 1704
- scientific, 1704
- seekdir, 1683
- setf, 1695
- setstate, 1696
- showbase, 1704
- showpoint, 1704
- showpos, 1704
- skipws, 1705
- sync_with_stdio, 1696
- tie, 1697
- traits_type, 1683

- trunc, 1705
- unitbuf, 1705
- unsetf, 1697
- uppercase, 1705
- widen, 1698
- width, 1698
- xalloc, 1699
- std::basic_istream, 1705
- ~basic_istream, 1720
- _M_gcount, 1764
- _M_getloc, 1720
- _M_write, 1720
- __ctype_type, 1713
- __num_get_type, 1713
- __num_put_type, 1713, 1714
- adjustfield, 1764
- app, 1764
- ate, 1764
- bad, 1721
- badbit, 1764
- basefield, 1765
- basic_istream, 1720
- beg, 1765
- binary, 1765
- boolalpha, 1765
- char_type, 1714
- clear, 1721
- copyfmt, 1722
- cur, 1766
- dec, 1766
- end, 1766
- eof, 1722
- eofbit, 1766
- event, 1719
- event_callback, 1715
- exceptions, 1722, 1723
- fail, 1723
- failbit, 1766
- fill, 1724
- fixed, 1767
- flags, 1725
- floatfield, 1767
- flush, 1725
- fmtflags, 1715
- gcount, 1726
- get, 1726–1729
- getline, 1729, 1730
- getloc, 1731
- good, 1731
- goodbit, 1767
- hex, 1767
- ignore, 1731–1733
- imbue, 1733
- in, 1768
- init, 1734
- int_type, 1716
- internal, 1768
- iostate, 1716
- iword, 1734
- left, 1768
- narrow, 1734
- oct, 1768
- off_type, 1717
- openmode, 1717
- operator void *, 1735
- operator<<, 1735–1742
- operator>>, 1742–1748
- out, 1768
- peek, 1749
- pos_type, 1718
- precision, 1749
- put, 1750
- putback, 1750
- pword, 1751
- rdbuf, 1751, 1752
- rdstate, 1752
- read, 1753
- readsome, 1753
- register_callback, 1754
- right, 1769
- scientific, 1769
- seekdir, 1718
- seekg, 1754, 1755
- seekp, 1756
- setf, 1757
- setstate, 1757
- showbase, 1769
- showpoint, 1769
- showpos, 1769
- skipws, 1769
- sync, 1758
- sync_with_stdio, 1759

- tellg, 1759
- tellp, 1759
- tie, 1760
- traits_type, 1719
- trunc, 1769
- unget, 1761
- unitbuf, 1770
- unsetf, 1761
- uppercase, 1770
- widen, 1761
- width, 1762
- write, 1763
- xalloc, 1763
- std::basic_istream, 1770
 - ~basic_istream, 1781
 - _M_gcount, 1815
 - _M_getloc, 1782
 - __ctype_type, 1776
 - __num_get_type, 1776
 - __num_put_type, 1777
- adjustfield, 1815
- app, 1815
- ate, 1815
- bad, 1782
- badbit, 1816
- basefield, 1816
- basic_istream, 1781
- beg, 1816
- binary, 1816
- boolalpha, 1816
- char_type, 1777
- clear, 1782
- copyfmt, 1783
- cur, 1817
- dec, 1817
- end, 1817
- eof, 1783
- eofbit, 1817
- event, 1781
- event_callback, 1777
- exceptions, 1783, 1784
- fail, 1784
- failbit, 1817
- fill, 1785
- fixed, 1818
- flags, 1786
- floatfield, 1818
- fmtflags, 1778
- gcount, 1786
- get, 1787–1789
- getline, 1790, 1791
- getloc, 1791
- good, 1791
- goodbit, 1818
- hex, 1818
- ignore, 1792, 1793
- imbue, 1793
- in, 1819
- init, 1794
- int_type, 1778
- internal, 1819
- iostate, 1779
- iword, 1794
- left, 1819
- narrow, 1794
- oct, 1819
- off_type, 1779
- openmode, 1779
- operator void *, 1795
- operator>>, 1795–1802
- out, 1819
- peek, 1802
- pos_type, 1780
- precision, 1802, 1803
- putback, 1803
- pword, 1804
- rdbuf, 1804, 1805
- rdstate, 1805
- read, 1806
- readsome, 1806
- register_callback, 1807
- right, 1820
- scientific, 1820
- seekdir, 1780
- seekg, 1807, 1808
- setf, 1809
- setstate, 1810
- showbase, 1820
- showpoint, 1820
- showpos, 1820
- skipws, 1820
- sync, 1810

- sync_with_stdio, 1811
- tellg, 1811
- tie, 1812
- traits_type, 1780
- trunc, 1820
- unget, 1812
- unitbuf, 1821
- unsetf, 1813
- uppercase, 1821
- widen, 1813
- width, 1814
- xalloc, 1814
- std::basic_istream::sentry, 1821
- operator bool, 1823
- sentry, 1822
- traits_type, 1822
- std::basic_istream, 1823
- ~basic_istream, 1835
- _M_gcount, 1870
- _M_getloc, 1836
- __ctype_type, 1830
- __num_get_type, 1830
- __num_put_type, 1830
- adjustfield, 1870
- app, 1870
- ate, 1870
- bad, 1836
- badbit, 1871
- basefield, 1871
- basic_istream, 1835
- beg, 1871
- binary, 1871
- boolalpha, 1872
- char_type, 1831
- clear, 1836
- copyfmt, 1837
- cur, 1872
- dec, 1872
- end, 1872
- eof, 1837
- eofbit, 1872
- event, 1834
- event_callback, 1831
- exceptions, 1838
- fail, 1839
- failbit, 1873
- fill, 1839, 1840
- fixed, 1873
- flags, 1840
- floatfield, 1873
- fmtflags, 1831
- gcount, 1841
- get, 1841–1844
- getline, 1844, 1845
- getloc, 1846
- good, 1846
- goodbit, 1873
- hex, 1874
- ignore, 1846, 1847
- imbue, 1848
- in, 1874
- init, 1848
- int_type, 1832
- internal, 1874
- iostate, 1832
- iword, 1849
- left, 1874
- narrow, 1849
- oct, 1874
- off_type, 1833
- openmode, 1833
- operator void *, 1850
- operator>>, 1850–1856
- out, 1875
- peek, 1857
- pos_type, 1833
- precision, 1857, 1858
- putback, 1858
- pword, 1859
- rdbuf, 1859
- rdstate, 1860
- read, 1860
- readsome, 1861
- register_callback, 1862
- right, 1875
- scientific, 1875
- seekdir, 1834
- seekg, 1862, 1863
- setf, 1863, 1864
- setstate, 1864
- showbase, 1875
- showpoint, 1875

- showpos, 1875
- skipws, 1875
- str, 1865
- sync, 1865
- sync_with_stdio, 1866
- tellg, 1866
- tie, 1867
- traits_type, 1834
- trunc, 1876
- unget, 1867
- unitbuf, 1876
- unsetf, 1868
- uppercase, 1876
- widen, 1868
- width, 1869
- xalloc, 1869
- std::basic_ofstream, 1876
 - ~basic_ofstream, 1888
 - _M_getloc, 1888
 - _M_write, 1889
 - __ctype_type, 1883
 - __num_get_type, 1883
 - __num_put_type, 1883
- adjustfield, 1914
- app, 1914
- ate, 1914
- bad, 1889
- badbit, 1915
- basefield, 1915
- basic_ofstream, 1887, 1888
- beg, 1915
- binary, 1915
- boolalpha, 1916
- char_type, 1883
- clear, 1889
- close, 1890
- copyfmt, 1890
- cur, 1916
- dec, 1916
- end, 1916
- eof, 1891
- eofbit, 1916
- event, 1887
- event_callback, 1883
- exceptions, 1891
- fail, 1892
- failbit, 1917
- fill, 1892, 1893
- fixed, 1917
- flags, 1893
- floatfield, 1917
- flush, 1894
- fmtflags, 1884
- getloc, 1894
- good, 1894
- goodbit, 1917
- hex, 1918
- imbue, 1895
- in, 1918
- init, 1895
- int_type, 1885
- internal, 1918
- iostate, 1885
- is_open, 1895
- iword, 1896
- left, 1918
- narrow, 1896
- oct, 1918
- off_type, 1885
- open, 1897
- openmode, 1886
- operator void *, 1897
- operator<<, 1898–1904
- out, 1919
- pos_type, 1886
- precision, 1905
- put, 1905
- pword, 1906
- rdbuf, 1906
- rdstate, 1907
- register_callback, 1907
- right, 1919
- scientific, 1919
- seekdir, 1886
- seekp, 1908
- setf, 1909
- setstate, 1909
- showbase, 1919
- showpoint, 1919
- showpos, 1919
- skipws, 1920
- sync_with_stdio, 1910

- tellp, 1910
- tie, 1911
- traits_type, 1887
- trunc, 1920
- unitbuf, 1920
- unsetf, 1912
- uppercase, 1920
- widen, 1912
- width, 1912, 1913
- write, 1913
- xalloc, 1914
- std::basic_ostream, 1920
 - ~basic_ostream, 1931
 - _M_getloc, 1931
 - _M_write, 1932
 - __ctype_type, 1926
 - __num_get_type, 1926
 - __num_put_type, 1926
 - adjustfield, 1956
 - app, 1956
 - ate, 1957
 - bad, 1932
 - badbit, 1957
 - basefield, 1957
 - basic_ostream, 1931
 - beg, 1957
 - binary, 1958
 - boolalpha, 1958
 - char_type, 1926
 - clear, 1932
 - copyfmt, 1933
 - cur, 1958
 - dec, 1958
 - end, 1958
 - eof, 1933
 - eofbit, 1958
 - event, 1931
 - event_callback, 1927
 - exceptions, 1934
 - fail, 1935
 - failbit, 1959
 - fill, 1935, 1936
 - fixed, 1959
 - flags, 1936
 - floatfield, 1959
 - flush, 1937
 - fmtflags, 1927
 - getloc, 1937
 - good, 1937
 - goodbit, 1959
 - hex, 1960
 - imbue, 1938
 - in, 1960
 - init, 1938
 - int_type, 1928
 - internal, 1960
 - iostate, 1928
 - iword, 1938
 - left, 1960
 - narrow, 1939
 - oct, 1961
 - off_type, 1929
 - openmode, 1929
 - operator void *, 1939
 - operator<<, 1940–1946
 - out, 1961
 - pos_type, 1930
 - precision, 1946, 1947
 - put, 1947
 - pword, 1948
 - rdbuf, 1948, 1949
 - rdstate, 1949
 - register_callback, 1950
 - right, 1961
 - scientific, 1961
 - seekdir, 1930
 - seekp, 1950
 - setf, 1951
 - setstate, 1952
 - showbase, 1961
 - showpoint, 1961
 - showpos, 1962
 - skipws, 1962
 - sync_with_stdio, 1952
 - tellp, 1953
 - tie, 1953
 - traits_type, 1930
 - trunc, 1962
 - unitbuf, 1962
 - unsetf, 1954
 - uppercase, 1962
 - widen, 1954

- width, 1955
- write, 1955
- xalloc, 1956
- std::basic_ostream::sentry, 1963
 - ~sentry, 1963
 - operator bool, 1964
 - sentry, 1963
- std::basic_ostringstream, 1964
 - ~basic_ostringstream, 1976
 - _M_getloc, 1976
 - _M_write, 1977
 - __ctype_type, 1971
 - __num_get_type, 1971
 - __num_put_type, 1971
 - adjustfield, 2002
 - app, 2002
 - ate, 2002
 - bad, 1977
 - badbit, 2002
 - basefield, 2002
 - basic_ostringstream, 1975
 - beg, 2003
 - binary, 2003
 - boolalpha, 2003
 - char_type, 1971
 - clear, 1977
 - copyfmt, 1978
 - cur, 2003
 - dec, 2003
 - end, 2004
 - eof, 1978
 - eofbit, 2004
 - event, 1975
 - event_callback, 1971
 - exceptions, 1979
 - fail, 1980
 - failbit, 2004
 - fill, 1980
 - fixed, 2004
 - flags, 1981
 - floatfield, 2004
 - flush, 1982
 - fmtflags, 1972
 - getloc, 1982
 - good, 1982
 - goodbit, 2005
 - hex, 2005
 - imbue, 1983
 - in, 2005
 - init, 1983
 - int_type, 1973
 - internal, 2005
 - iostate, 1973
 - iword, 1983
 - left, 2006
 - narrow, 1984
 - oct, 2006
 - off_type, 1973
 - openmode, 1974
 - operator void *, 1984
 - operator<<, 1985–1991
 - out, 2006
 - pos_type, 1974
 - precision, 1991, 1992
 - put, 1992
 - pword, 1992
 - rdbuf, 1993, 1994
 - rdstate, 1994
 - register_callback, 1994
 - right, 2006
 - scientific, 2006
 - seekdir, 1974
 - seekp, 1994, 1995
 - setf, 1995, 1996
 - setstate, 1996
 - showbase, 2006
 - showpoint, 2007
 - showpos, 2007
 - skipws, 2007
 - str, 1997
 - sync_with_stdio, 1997
 - tellp, 1998
 - tie, 1998, 1999
 - traits_type, 1975
 - trunc, 2007
 - unitbuf, 2007
 - unsetf, 1999
 - uppercase, 2007
 - widen, 1999
 - width, 2000
 - write, 2000
 - xalloc, 2001

-
- std::basic_regex, 2008
 - ~basic_regex, 2013
 - assign, 2013–2016
 - basic_regex, 2010–2012
 - flags, 2016
 - getloc, 2016
 - imbue, 2017
 - mark_count, 2017
 - operator=, 2017, 2018
 - swap, 2018
 - std::basic_streambuf, 2019
 - ~basic_streambuf, 2025
 - _M_buf_locale, 2041
 - _M_in_beg, 2042
 - _M_in_cur, 2042
 - _M_in_end, 2042
 - _M_out_beg, 2043
 - _M_out_cur, 2043
 - _M_out_end, 2043
 - __streambuf_type, 2023
 - basic_streambuf, 2025
 - char_type, 2023
 - eback, 2026
 - egptr, 2026
 - eptr, 2026
 - gbump, 2027
 - getloc, 2027
 - gptr, 2027
 - imbue, 2028
 - in_avail, 2028
 - int_type, 2023
 - off_type, 2024
 - overflow, 2029
 - pbackfail, 2029
 - pbase, 2030
 - pbump, 2030
 - pos_type, 2024
 - pptr, 2030
 - pubimbue, 2031
 - pubseekoff, 2031
 - pubseekpos, 2032
 - pubsetbuf, 2032
 - pubsync, 2032
 - sbumpc, 2033
 - seekoff, 2033
 - seekpos, 2033
 - setbuf, 2034
 - setg, 2034
 - setp, 2035
 - sgetc, 2035
 - sgetn, 2035
 - showmanyc, 2036
 - sngetc, 2036
 - sputbackc, 2037
 - sputc, 2037
 - sputn, 2038
 - stossc, 2038
 - sungetc, 2038
 - sync, 2039
 - traits_type, 2024
 - uflow, 2039
 - underflow, 2039
 - xsgetn, 2040
 - xsputn, 2041
 - std::basic_string, 2044
 - ~basic_string, 2053
 - append, 2053–2056
 - assign, 2056–2059
 - at, 2060
 - back, 2061
 - basic_string, 2050–2052
 - begin, 2061
 - c_str, 2061
 - capacity, 2062
 - cbegin, 2062
 - cend, 2062
 - clear, 2062
 - compare, 2063–2066
 - copy, 2066
 - crbegin, 2067
 - crend, 2067
 - data, 2067
 - empty, 2068
 - end, 2068
 - erase, 2068, 2069
 - find, 2070, 2071
 - find_first_not_of, 2072, 2073
 - find_first_of, 2074, 2075
 - find_last_not_of, 2076, 2077
 - find_last_of, 2078, 2079
 - front, 2079, 2080
 - get_allocator, 2080
-

- insert, 2080–2084
- length, 2085
- max_size, 2085
- npos, 2102
- operator+=, 2085, 2086
- operator=, 2087, 2088
- push_back, 2089
- rbegin, 2089, 2090
- rend, 2090
- replace, 2090–2096
- reserve, 2097
- resize, 2098
- rfind, 2098–2100
- shrink_to_fit, 2100
- size, 2101
- substr, 2101
- swap, 2102
- std::basic_stringbuf, 2102
 - _M_buf_locale, 2126
 - _M_in_beg, 2126
 - _M_in_cur, 2126
 - _M_in_end, 2126
 - _M_mode, 2127
 - _M_out_beg, 2127
 - _M_out_cur, 2127
 - _M_out_end, 2128
 - __streambuf_type, 2106
- basic_stringbuf, 2108
- char_type, 2106
- eback, 2108
- egptr, 2109
- epptr, 2109
- gbump, 2109
- getloc, 2110
- gptr, 2110
- imbue, 2110
- in_avail, 2111
- int_type, 2106
- off_type, 2107
- overflow, 2111
- pbackfail, 2112
- pbase, 2113
- pbump, 2113
- pos_type, 2107
- pptr, 2113
- pubimbue, 2114
- pubseekoff, 2114
- pubseekpos, 2114
- pubsetbuf, 2115
- pubsync, 2115
- sbumpc, 2115
- seekoff, 2116
- seekpos, 2116
- setbuf, 2117
- setg, 2117
- setp, 2118
- sgetc, 2118
- sgetn, 2119
- showmanyc, 2119
- snextc, 2120
- sputbackc, 2120
- sputc, 2120
- sputn, 2121
- stossc, 2121
- str, 2121, 2122
- sungetc, 2122
- sync, 2123
- traits_type, 2107
- uflow, 2123
- underflow, 2123
- xsgetn, 2124
- xsgptr, 2125
- std::basic_stringstream, 2128
 - ~basic_stringstream, 2144
 - _M_gcount, 2188
 - _M_getloc, 2144
 - _M_write, 2145
 - __ctype_type, 2137
 - __num_get_type, 2137
 - __num_put_type, 2137
- adjustfield, 2188
- app, 2189
- ate, 2189
- bad, 2145
- badbit, 2189
- basefield, 2189
- basic_stringstream, 2143, 2144
- beg, 2189
- binary, 2190
- boolalpha, 2190
- char_type, 2138
- clear, 2145

- copyfmt, 2146
- cur, 2190
- dec, 2190
- end, 2190
- eof, 2146
- eofbit, 2190
- event, 2143
- event_callback, 2138
- exceptions, 2147
- fail, 2148
- failbit, 2191
- fill, 2148, 2149
- fixed, 2191
- flags, 2149
- floatfield, 2191
- flush, 2150
- fmtflags, 2139
- gcount, 2150
- get, 2150–2153
- getline, 2154, 2155
- getloc, 2155
- good, 2155
- goodbit, 2191
- hex, 2192
- ignore, 2156, 2157
- imbue, 2157
- in, 2192
- init, 2158
- int_type, 2139, 2140
- internal, 2192
- iostate, 2140
- isword, 2158
- left, 2192
- narrow, 2159
- oct, 2193
- off_type, 2140, 2141
- openmode, 2141
- operator void *, 2159
- operator<<, 2159–2166
- operator>>, 2166–2172
- out, 2193
- peek, 2173
- pos_type, 2141, 2142
- precision, 2173
- put, 2174
- putback, 2174
- pwd, 2175
- rdbuf, 2175, 2176
- rdstate, 2176
- read, 2177
- readsome, 2177
- register_callback, 2178
- right, 2193
- scientific, 2193
- seekdir, 2142
- seekg, 2178, 2179
- seekp, 2180
- setf, 2181
- setstate, 2181
- showbase, 2193
- showpoint, 2193
- showpos, 2194
- skipws, 2194
- str, 2182
- sync, 2183
- sync_with_stdio, 2183
- tellg, 2183
- tellp, 2184
- tie, 2184, 2185
- traits_type, 2142, 2143
- trunc, 2194
- unget, 2185
- unitbuf, 2194
- unsetf, 2186
- uppercase, 2194
- widen, 2186
- width, 2186, 2187
- write, 2187
- xalloc, 2188
- std::bernoulli_distribution, 2195
- bernoulli_distribution, 2196
- max, 2196
- min, 2196
- operator(), 2196
- p, 2196
- param, 2196, 2197
- reset, 2197
- result_type, 2195
- std::bernoulli_distribution::param_type, 2197
- std::bidirectional_iterator_tag, 2198
- std::binary_function, 2199

- first_argument_type, 2200
- result_type, 2200
- second_argument_type, 2200
- std::binary_negate, 2201
 - first_argument_type, 2201
 - result_type, 2202
 - second_argument_type, 2202
- std::binder1st, 2202
 - argument_type, 2203
 - result_type, 2203
- std::binder2nd, 2204
 - argument_type, 2205
 - result_type, 2205
- std::binomial_distribution, 2205
 - max, 2207
 - min, 2207
 - operator<<, 2209
 - operator>>, 2209
 - operator(), 2207
 - operator==, 2209
 - p, 2208
 - param, 2208
 - reset, 2208
 - result_type, 2207
 - t, 2209
- std::binomial_distribution::param_type, 2210
- std::bitset, 2211
 - all, 2217
 - any, 2217
 - bitset, 2215, 2216
 - count, 2217
 - flip, 2217, 2218
 - none, 2218
 - operator<<, 2219
 - operator<=, 2219
 - operator>>, 2219
 - operator>=, 2219
 - operator~, 2221
 - operator^=, 2221
 - operator==, 2219
 - operator&=, 2218
 - reset, 2221, 2222
 - set, 2222
 - size, 2222
 - test, 2223
 - to_string, 2223
 - to_ulong, 2223
- std::bitset::reference, 2224
- std::cauchy_distribution, 2225
 - max, 2226
 - min, 2226
 - operator(), 2226
 - param, 2227
 - reset, 2227
 - result_type, 2226
- std::cauchy_distribution::param_type, 2227
- std::char_traits, 2228
- std::char_traits< __gnu_cxx::character< V, I, S > >, 2230
- std::char_traits< char >, 2231
- std::char_traits< wchar_t >, 2232
- std::chi_squared_distribution, 2233
 - max, 2235
 - min, 2235
 - operator<<, 2236
 - operator>>, 2236
 - operator(), 2235
 - operator==, 2236
 - param, 2235
 - reset, 2236
 - result_type, 2234
- std::chi_squared_distribution::param_type, 2237
- std::chrono, 734
 - duration_cast, 738
 - hours, 737
 - microseconds, 737
 - milliseconds, 737
 - minutes, 737
 - nanoseconds, 737
 - seconds, 737
 - time_point_cast, 738
- std::chrono::duration, 2238
- std::chrono::duration_values, 2239
- std::chrono::system_clock, 2240
- std::chrono::time_point, 2241
- std::chrono::treat_as_floating_point, 2242
- std::codecvt, 2242
 - do_out, 2245
 - in, 2245

- out, [2246](#)
- unshift, [2247](#)
- std::codecvt< _InternT, _ExternT, encoding_state >, [2248](#)
 - do_out, [2251](#)
 - in, [2251](#)
 - out, [2252](#)
 - unshift, [2253](#)
- std::codecvt< char, char, mbstate_t >, [2254](#)
 - do_out, [2256](#)
 - in, [2256](#)
 - out, [2257](#)
 - unshift, [2258](#)
- std::codecvt< wchar_t, char, mbstate_t >, [2259](#)
 - do_out, [2261](#)
 - in, [2261](#)
 - out, [2262](#)
 - unshift, [2263](#)
- std::codecvt_base, [2264](#)
- std::codecvt_byname, [2265](#)
 - do_out, [2268](#)
 - in, [2268](#)
 - out, [2269](#)
 - unshift, [2270](#)
- std::collate, [2271](#)
 - ~collate, [2275](#)
 - char_type, [2274](#)
 - collate, [2274](#)
 - compare, [2275](#)
 - do_compare, [2275](#)
 - do_hash, [2276](#)
 - do_transform, [2276](#)
 - hash, [2277](#)
 - id, [2278](#)
 - string_type, [2274](#)
 - transform, [2277](#)
- std::collate_byname, [2278](#)
 - char_type, [2281](#)
 - compare, [2281](#)
 - do_compare, [2281](#)
 - do_hash, [2282](#)
 - do_transform, [2282](#)
 - hash, [2283](#)
 - id, [2284](#)
 - string_type, [2281](#)
 - transform, [2283](#)
 - complex, [2284](#)
 - complex, [2286](#)
 - operator+=, [2286](#)
 - operator-=, [2286](#)
 - value_type, [2286](#)
 - std::condition_variable, [2287](#)
 - std::condition_variable_any, [2288](#)
 - std::conditional, [2289](#)
 - std::const_mem_fun1_ref_t, [2289](#)
 - first_argument_type, [2291](#)
 - result_type, [2291](#)
 - second_argument_type, [2291](#)
 - std::const_mem_fun1_t, [2291](#)
 - first_argument_type, [2293](#)
 - result_type, [2293](#)
 - second_argument_type, [2293](#)
 - std::const_mem_fun_ref_t, [2293](#)
 - argument_type, [2295](#)
 - result_type, [2295](#)
 - std::const_mem_fun_t, [2295](#)
 - argument_type, [2297](#)
 - result_type, [2297](#)
 - std::ctype, [2297](#)
 - char_type, [2300](#)
 - do_is, [2301](#)
 - do_narrow, [2302](#)
 - do_scan_is, [2303](#)
 - do_scan_not, [2303](#)
 - do_tolower, [2304](#)
 - do_toupper, [2305](#)
 - do_widen, [2305](#), [2306](#)
 - id, [2312](#)
 - is, [2306](#), [2307](#)
 - narrow, [2307](#), [2308](#)
 - scan_is, [2309](#)
 - scan_not, [2309](#)
 - tolower, [2309](#), [2310](#)
 - toupper, [2310](#), [2311](#)
 - widen, [2311](#), [2312](#)
 - std::ctype< char >, [2313](#)
 - ~ctype, [2316](#)
 - char_type, [2316](#)
 - classic_table, [2317](#)
 - ctype, [2316](#)

- do_narrow, [2317](#)
- do_tolower, [2318](#)
- do_toupper, [2319](#)
- do_widen, [2319](#), [2320](#)
- id, [2326](#)
- is, [2320](#), [2321](#)
- narrow, [2321](#), [2322](#)
- scan_is, [2322](#)
- scan_not, [2323](#)
- table, [2323](#)
- table_size, [2326](#)
- tolower, [2323](#), [2324](#)
- toupper, [2324](#), [2325](#)
- widen, [2325](#), [2326](#)
- std::ctype< wchar_t >, [2327](#)
 - ~ctype, [2331](#)
 - char_type, [2330](#)
 - ctype, [2331](#)
 - do_is, [2331](#), [2332](#)
 - do_narrow, [2333](#), [2334](#)
 - do_scan_is, [2335](#)
 - do_scan_not, [2336](#)
 - do_tolower, [2337](#), [2338](#)
 - do_toupper, [2339](#), [2340](#)
 - do_widen, [2340](#), [2341](#)
 - id, [2347](#)
 - is, [2342](#)
 - narrow, [2343](#)
 - scan_is, [2344](#)
 - scan_not, [2344](#)
 - tolower, [2345](#)
 - toupper, [2345](#), [2346](#)
 - widen, [2346](#), [2347](#)
- std::ctype_base, [2348](#)
- std::ctype_byname, [2349](#)
 - char_type, [2351](#)
 - do_is, [2352](#)
 - do_narrow, [2352](#), [2353](#)
 - do_scan_is, [2354](#)
 - do_scan_not, [2354](#)
 - do_tolower, [2355](#)
 - do_toupper, [2356](#)
 - do_widen, [2356](#), [2357](#)
 - id, [2363](#)
 - is, [2357](#), [2358](#)
 - narrow, [2358](#), [2359](#)
 - scan_is, [2360](#)
 - scan_not, [2360](#)
 - tolower, [2360](#), [2361](#)
 - toupper, [2361](#), [2362](#)
 - widen, [2362](#), [2363](#)
- std::ctype_byname< char >, [2364](#)
 - char_type, [2366](#)
 - classic_table, [2367](#)
 - do_narrow, [2367](#)
 - do_tolower, [2368](#)
 - do_toupper, [2369](#)
 - do_widen, [2369](#), [2370](#)
 - id, [2376](#)
 - is, [2370](#), [2371](#)
 - narrow, [2371](#), [2372](#)
 - scan_is, [2372](#)
 - scan_not, [2373](#)
 - table, [2373](#)
 - table_size, [2376](#)
 - tolower, [2373](#), [2374](#)
 - toupper, [2374](#), [2375](#)
 - widen, [2375](#), [2376](#)
- std::decay, [2377](#)
- std::decimal, [738](#)
 - decimal32_to_long_long, [749](#)
- std::decimal::decimal128, [2377](#)
 - decimal128, [2379](#)
- std::decimal::decimal32, [2379](#)
 - decimal32, [2381](#)
- std::decimal::decimal64, [2381](#)
 - decimal64, [2383](#)
- std::default_delete, [2383](#)
- std::defer_lock_t, [2384](#)
- std::deque, [2385](#)
 - ~deque, [2394](#)
 - _M_fill_initialize, [2394](#)
 - _M_initialize_map, [2395](#)
 - _M_new_elements_at_back, [2395](#)
 - _M_new_elements_at_front, [2395](#)
 - _M_pop_back_aux, [2396](#)
 - _M_pop_front_aux, [2396](#)
 - _M_push_back_aux, [2396](#)
 - _M_push_front_aux, [2396](#)
 - _M_range_check, [2397](#)
 - _M_range_initialize, [2397](#)
 - _M_reallocate_map, [2398](#)

- `_M_reserve_elements_at_back`, 2398
- `_M_reserve_elements_at_front`, 2398
- `_M_reserve_map_at_back`, 2399
- `_M_reserve_map_at_front`, 2399
- `assign`, 2399, 2400
- `at`, 2400, 2401
- `back`, 2401, 2402
- `begin`, 2402
- `cbegin`, 2402
- `cend`, 2402
- `clear`, 2403
- `crbegin`, 2403
- `crend`, 2403
- `deque`, 2391–2393
- `emplace`, 2403
- `empty`, 2404
- `end`, 2404
- `erase`, 2404, 2405
- `front`, 2405, 2406
- `get_allocator`, 2406
- `insert`, 2406–2408
- `max_size`, 2408
- `operator=`, 2408, 2409
- `pop_back`, 2410
- `pop_front`, 2411
- `push_back`, 2411
- `push_front`, 2411
- `rbegin`, 2412
- `rend`, 2412
- `resize`, 2413
- `shrink_to_fit`, 2413
- `size`, 2413
- `swap`, 2414
- `std::discard_block_engine`, 2414
 - `base`, 2418
 - `discard`, 2418
 - `discard_block_engine`, 2416, 2417
 - `max`, 2418
 - `min`, 2418
 - `operator<<`, 2420
 - `operator>>`, 2420
 - `operator()`, 2418
 - `operator==`, 2420
 - `result_type`, 2416
 - `seed`, 2419
- `std::discrete_distribution`, 2421
 - `max`, 2423
 - `min`, 2423
 - `operator<<`, 2424
 - `operator>>`, 2424
 - `operator()`, 2423
 - `param`, 2423
 - `probabilities`, 2424
 - `reset`, 2424
 - `result_type`, 2422
- `std::discrete_distribution::param_type`, 2425
- `std::divides`, 2426
 - `first_argument_type`, 2427
 - `result_type`, 2427
 - `second_argument_type`, 2427
- `std::domain_error`, 2428
 - `what`, 2428
- `std::enable_if`, 2429
- `std::enable_shared_from_this`, 2429
- `std::equal_to`, 2430
 - `first_argument_type`, 2432
 - `result_type`, 2432
 - `second_argument_type`, 2432
- `std::error_category`, 2432
- `std::error_code`, 2433
- `std::error_condition`, 2434
- `std::exception`, 2434
 - `what`, 2436
- `std::exponential_distribution`, 2436
 - `exponential_distribution`, 2438
 - `lambda`, 2438
 - `max`, 2438
 - `min`, 2438
 - `operator()`, 2438
 - `param`, 2439
 - `reset`, 2439
 - `result_type`, 2437
- `std::exponential_distribution::param_type`, 2440
- `std::extreme_value_distribution`, 2440
 - `a`, 2442
 - `b`, 2442
 - `max`, 2442
 - `min`, 2442

- operator(), 2442
- param, 2443
- reset, 2443
- result_type, 2441
- std::extreme_value_distribution::param_type, 2443
- std::fisher_f_distribution, 2444
 - max, 2446
 - min, 2446
 - operator<<, 2447
 - operator>>, 2447
 - operator(), 2446
 - operator==, 2447
 - param, 2446
 - reset, 2447
 - result_type, 2445
- std::fisher_f_distribution::param_type, 2448
- std::forward_iterator_tag, 2449
- std::forward_list, 2450
 - ~forward_list, 2456
 - assign, 2457
 - before_begin, 2458
 - begin, 2458
 - cbegin, 2459
 - cend, 2459
 - clear, 2459
 - emplace_after, 2460
 - emplace_front, 2460
 - empty, 2460
 - end, 2461
 - erase_after, 2461, 2462
 - forward_list, 2454–2456
 - front, 2462
 - get_allocator, 2462
 - insert_after, 2463, 2464
 - max_size, 2465
 - merge, 2465
 - operator=, 2466
 - pop_front, 2467
 - push_front, 2467
 - remove, 2467
 - remove_if, 2468
 - resize, 2468
 - reverse, 2469
 - sort, 2469
 - splice_after, 2469, 2470
 - swap, 2471
 - unique, 2471
- std::fpos, 2472
 - fpos, 2473
 - operator streamoff, 2473
 - operator+, 2473
 - operator+=, 2473
 - operator-, 2473
 - operator=, 2474
 - state, 2474
- std::front_insert_iterator, 2474
 - container_type, 2476
 - difference_type, 2476
 - front_insert_iterator, 2477
 - iterator_category, 2476
 - operator*, 2477
 - operator++, 2477
 - operator=, 2478
 - pointer, 2476
 - reference, 2476
 - value_type, 2477
- std::function< _Res(_ArgTypes...)>, 2478
 - function, 2481, 2482
 - operator bool, 2482
 - operator(), 2482
 - operator=, 2483–2485
 - swap, 2485
 - target, 2485, 2486
 - target_type, 2486
- std::future, 2486
 - _M_get_result, 2488
 - future, 2488
 - get, 2489
- std::future< _Res & >, 2489
 - _M_get_result, 2492
 - future, 2491
 - get, 2492
- std::future< void >, 2492
 - _M_get_result, 2495
 - future, 2494
 - get, 2495
- std::future_error, 2495
 - what, 2496

- std::gamma_distribution, 2496
 - alpha, 2499
 - beta, 2499
 - gamma_distribution, 2498
 - max, 2499
 - min, 2499
 - operator<=, 2501
 - operator>=, 2501
 - operator(), 2499
 - operator==, 2501
 - param, 2500
 - reset, 2500
 - result_type, 2498
- std::gamma_distribution::param_type, 2502
- std::geometric_distribution, 2503
 - max, 2504
 - min, 2504
 - operator(), 2504
 - p, 2504
 - param, 2505
 - reset, 2505
 - result_type, 2504
- std::geometric_distribution::param_type, 2505
- std::greater, 2506
 - first_argument_type, 2508
 - result_type, 2508
 - second_argument_type, 2508
- std::greater_equal, 2508
 - first_argument_type, 2510
 - result_type, 2510
 - second_argument_type, 2510
- std::gslice, 2510
- std::gslice_array, 2511
- std::has_nothrow_copy_assign, 2513
- std::has_nothrow_copy_constructor, 2513
- std::has_nothrow_default_constructor, 2514
- std::has_trivial_copy_assign, 2514
- std::has_trivial_copy_constructor, 2515
- std::has_trivial_default_constructor, 2515
- std::has_trivial_destructor, 2515
- std::hash, 2516
- std::hash< __debug::bitset< _Nb > >, 2517
- std::hash< __debug::vector< bool, _Alloc > >, 2518
- std::hash< __gnu_cxx::throw_value_limit >, 2519
 - argument_type, 2520
 - result_type, 2520
- std::hash< __gnu_cxx::throw_value_random >, 2520
 - argument_type, 2522
 - result_type, 2522
- std::hash< __profile::bitset< _Nb > >, 2522
- std::hash< __profile::vector< bool, _Alloc > >, 2523
- std::hash< __shared_ptr< _Tp, _Lp > >, 2523
 - argument_type, 2525
 - result_type, 2525
- std::hash< _Tp * >, 2525
- std::hash< error_code >, 2526
- std::hash< shared_ptr< _Tp > >, 2526
 - argument_type, 2528
 - result_type, 2528
- std::hash< string >, 2528
- std::hash< thread::id >, 2529
- std::hash< u16string >, 2529
- std::hash< u32string >, 2530
- std::hash< unique_ptr< _Tp, _Dp > >, 2531
 - argument_type, 2532
 - result_type, 2532
- std::hash< wstring >, 2532
- std::hash<::bitset< _Nb > >, 2533
- std::hash<::vector< bool, _Alloc > >, 2533
- std::independent_bits_engine, 2534
 - base, 2537
 - discard, 2537
 - independent_bits_engine, 2536, 2537
 - max, 2538
 - min, 2538
 - operator>=, 2540
 - operator(), 2538
 - operator==, 2540
 - result_type, 2535

- seed, 2539
- std::indirect_array, 2541
 - operator<=, 2543
 - operator>=, 2543
 - operator*=, 2542
 - operator^=, 2543
 - operator+=, 2543
 - operator-=, 2543
 - operator/=, 2543
 - operator%=, 2542
 - operator&=, 2542
- std::initializer_list, 2544
- std::input_iterator_tag, 2544
- std::insert_iterator, 2545
 - container_type, 2547
 - difference_type, 2547
 - insert_iterator, 2548
 - iterator_category, 2547
 - operator*, 2548
 - operator++, 2548
 - operator=, 2549
 - pointer, 2547
 - reference, 2547
 - value_type, 2548
- std::invalid_argument, 2550
 - what, 2550
- std::ios_base, 2551
 - ~ios_base, 2557
 - _M_getloc, 2557
 - adjustfield, 2563
 - app, 2563
 - ate, 2563
 - badbit, 2564
 - basefield, 2564
 - beg, 2564
 - binary, 2564
 - boolalpha, 2565
 - cur, 2565
 - dec, 2565
 - end, 2565
 - eofbit, 2565
 - event, 2557
 - event_callback, 2554
 - failbit, 2566
 - fixed, 2566
 - flags, 2557, 2558
 - floatfield, 2566
 - fmtflags, 2554
 - getloc, 2558
 - goodbit, 2566
 - hex, 2567
 - imbue, 2558
 - in, 2567
 - internal, 2567
 - iostate, 2555
 - lword, 2559
 - left, 2567
 - oct, 2567
 - openmode, 2556
 - out, 2568
 - precision, 2559, 2560
 - pword, 2560
 - register_callback, 2560
 - right, 2568
 - scientific, 2568
 - seekdir, 2556
 - setf, 2561
 - showbase, 2568
 - showpoint, 2568
 - showpos, 2568
 - skipws, 2569
 - sync_with_stdio, 2561
 - trunc, 2569
 - unitbuf, 2569
 - unsetf, 2562
 - uppercase, 2569
 - width, 2562
 - xalloc, 2563
- std::ios_base::failure, 2569
 - what, 2570
- std::is_base_of, 2571
- std::is_bind_expression, 2571
- std::is_bind_expression< _Bind< _-
Signature > >, 2572
- std::is_bind_expression< _Bind_result<
_Result, _Signature > >, 2572
- std::is_constructible, 2573
- std::is_convertible, 2574
- std::is_error_code_enum, 2574
- std::is_error_condition_enum, 2575
- std::is_explicitly_convertible, 2575
- std::is_lvalue_reference, 2576

-
- `std::is_nothrow_constructible`, 2577
 - `std::is_placeholder`, 2577
 - `std::is_placeholder< _Placeholder< _-
 Num > >`, 2578
 - `std::is_pod`, 2578
 - `std::is_reference`, 2579
 - `std::is_rvalue_reference`, 2579
 - `std::is_signed`, 2580
 - `std::is_standard_layout`, 2580
 - `std::is_trivial`, 2580
 - `std::is_unsigned`, 2581
 - `std::istream_iterator`, 2581
 - `difference_type`, 2583
 - `istream_iterator`, 2584
 - `iterator_category`, 2583
 - `pointer`, 2583
 - `reference`, 2583
 - `value_type`, 2583
 - `std::istreambuf_iterator`, 2584
 - `char_type`, 2586
 - `difference_type`, 2586
 - `equal`, 2588
 - `int_type`, 2586
 - `istream_type`, 2586
 - `istreambuf_iterator`, 2587, 2588
 - `iterator_category`, 2586
 - `operator*`, 2588
 - `operator++`, 2588
 - `pointer`, 2586
 - `reference`, 2587
 - `streambuf_type`, 2587
 - `traits_type`, 2587
 - `value_type`, 2587
 - `std::iterator`, 2589
 - `difference_type`, 2590
 - `iterator_category`, 2590
 - `pointer`, 2590
 - `reference`, 2590
 - `value_type`, 2591
 - `std::iterator_traits< _Tp * >`, 2591
 - `std::iterator_traits< const _Tp * >`, 2592
 - `std::length_error`, 2593
 - `what`, 2593
 - `std::less`, 2594
 - `first_argument_type`, 2595
 - `result_type`, 2595
 - `std::less_equal`, 2595
 - `first_argument_type`, 2597
 - `result_type`, 2597
 - `second_argument_type`, 2597
 - `std::linear_congruential_engine`, 2597
 - `discard`, 2600
 - `increment`, 2603
 - `linear_congruential_engine`, 2599
 - `max`, 2600
 - `min`, 2600
 - `modulus`, 2603
 - `multiplier`, 2603
 - `operator<<`, 2602
 - `operator>>`, 2602
 - `operator()`, 2600
 - `operator==`, 2602
 - `result_type`, 2599
 - `seed`, 2601
 - `std::list`, 2604
 - `_M_create_node`, 2611
 - `assign`, 2611, 2612
 - `back`, 2612, 2613
 - `begin`, 2613
 - `cbegin`, 2613
 - `cend`, 2613
 - `clear`, 2614
 - `crbegin`, 2614
 - `crend`, 2614
 - `emplace`, 2614
 - `empty`, 2615
 - `end`, 2615
 - `erase`, 2615, 2616
 - `front`, 2616, 2617
 - `get_allocator`, 2617
 - `insert`, 2617–2619
 - `list`, 2608–2610
 - `max_size`, 2619
 - `merge`, 2619, 2620
 - `operator=`, 2620, 2621
 - `pop_back`, 2621
 - `pop_front`, 2622
 - `push_back`, 2622
 - `push_front`, 2622
 - `rbegin`, 2623
 - `remove`, 2623
-

- remove_if, 2623
- rend, 2624
- resize, 2624, 2625
- reverse, 2625
- size, 2625
- sort, 2625, 2626
- splice, 2626, 2627
- swap, 2627
- unique, 2628
- std::locale, 2629
 - ~locale, 2633
 - all, 2636
 - category, 2631
 - classic, 2633
 - collate, 2636
 - combine, 2633
 - ctype, 2637
 - global, 2633
 - has_facet, 2635
 - locale, 2631, 2632
 - messages, 2637
 - monetary, 2637
 - name, 2634
 - none, 2637
 - numeric, 2638
 - operator(), 2634
 - operator=, 2635
 - operator==, 2635
 - time, 2638
 - use_facet, 2636
- std::locale::facet, 2638
 - ~facet, 2640
 - facet, 2640
- std::locale::id, 2640
 - has_facet, 2641
 - id, 2641
 - use_facet, 2642
- std::lock_guard, 2642
- std::logic_error, 2643
 - logic_error, 2644
 - what, 2644
- std::logical_and, 2644
 - first_argument_type, 2646
 - result_type, 2646
 - second_argument_type, 2646
- std::logical_not, 2646
 - argument_type, 2648
 - result_type, 2648
- std::logical_or, 2648
 - first_argument_type, 2650
 - result_type, 2650
 - second_argument_type, 2650
- std::lognormal_distribution, 2650
 - max, 2652
 - min, 2652
 - operator<<, 2653
 - operator>>, 2654
 - operator(), 2652
 - operator==, 2653
 - param, 2652, 2653
 - reset, 2653
 - result_type, 2652
- std::lognormal_distribution::param_type, 2654
- std::make_signed, 2655
- std::make_unsigned, 2655
- std::map, 2656
 - at, 2661
 - begin, 2662
 - cbegin, 2662
 - cend, 2662
 - clear, 2663
 - count, 2663
 - crbegin, 2663
 - crend, 2663
 - empty, 2664
 - end, 2664
 - equal_range, 2664, 2665
 - erase, 2666
 - find, 2667
 - get_allocator, 2668
 - insert, 2668–2670
 - key_comp, 2670
 - lower_bound, 2670, 2671
 - map, 2659–2661
 - max_size, 2671
 - operator=, 2671, 2672
 - rbegin, 2673
 - rend, 2673, 2674
 - size, 2674
 - swap, 2674
 - upper_bound, 2675

- value_comp, 2675
- std::mask_array, 2676
 - operator<=, 2678
 - operator>=, 2678
 - operator*=, 2678
 - operator^=, 2678
 - operator+=, 2678
 - operator-=, 2678
 - operator/=, 2678
 - operator%=, 2677
 - operator&=, 2677
- std::match_results, 2679
 - ~match_results, 2684
 - begin, 2684
 - cbegin, 2684
 - cend, 2684
 - empty, 2685
 - end, 2685
 - format, 2685, 2686
 - get_allocator, 2686
 - length, 2686
 - match_results, 2683
 - max_size, 2687
 - operator=, 2687
 - position, 2687
 - prefix, 2688
 - size, 2688
 - str, 2689
 - suffix, 2689
 - swap, 2689
- std::mem_fun1_ref_t, 2690
 - first_argument_type, 2691
 - result_type, 2691
 - second_argument_type, 2691
- std::mem_fun1_t, 2691
 - first_argument_type, 2693
 - result_type, 2693
 - second_argument_type, 2693
- std::mem_fun_ref_t, 2693
 - argument_type, 2695
 - result_type, 2695
- std::mem_fun_t, 2695
 - argument_type, 2697
 - result_type, 2697
- std::messages, 2697
 - ~messages, 2701
- char_type, 2700
- id, 2701
- messages, 2700
- string_type, 2700
- std::messages_base, 2701
- std::messages_byname, 2702
 - char_type, 2705
 - id, 2705
 - string_type, 2705
- std::minus, 2705
 - first_argument_type, 2707
 - result_type, 2707
 - second_argument_type, 2707
- std::modulus, 2707
 - first_argument_type, 2709
 - result_type, 2709
 - second_argument_type, 2709
- std::money_base, 2709
- std::money_get, 2711
 - ~money_get, 2714
 - char_type, 2713
 - do_get, 2714
 - get, 2715, 2716
 - id, 2716
 - iter_type, 2713
 - money_get, 2714
 - string_type, 2713
- std::money_put, 2717
 - ~money_put, 2719
 - char_type, 2719
 - do_put, 2720
 - id, 2722
 - iter_type, 2719
 - money_put, 2719
 - put, 2721, 2722
 - string_type, 2719
- std::moneypunct, 2723
 - ~moneypunct, 2727
 - char_type, 2725
 - curr_symbol, 2727
 - decimal_point, 2727
 - do_curr_symbol, 2727
 - do_decimal_point, 2728
 - do_frac_digits, 2728
 - do_grouping, 2729
 - do_neg_format, 2729

- do_negative_sign, 2729
- do_pos_format, 2730
- do_positive_sign, 2730
- do_thousands_sep, 2731
- frac_digits, 2731
- grouping, 2731
- id, 2735
- intl, 2735
- moneypunct, 2726
- neg_format, 2732
- negative_sign, 2733
- pos_format, 2733
- positive_sign, 2734
- string_type, 2725
- thousands_sep, 2734
- std::moneypunct_byname, 2735
 - char_type, 2738
 - curr_symbol, 2739
 - decimal_point, 2739
 - do_curr_symbol, 2739
 - do_decimal_point, 2740
 - do_frac_digits, 2740
 - do_grouping, 2740
 - do_neg_format, 2741
 - do_negative_sign, 2741
 - do_pos_format, 2742
 - do_positive_sign, 2742
 - do_thousands_sep, 2743
 - frac_digits, 2743
 - grouping, 2743
 - id, 2747
 - intl, 2747
 - neg_format, 2744
 - negative_sign, 2745
 - pos_format, 2745
 - positive_sign, 2746
 - string_type, 2738
 - thousands_sep, 2746
- std::move_iterator, 2747
- std::multimap, 2748
 - begin, 2754
 - cbegin, 2754
 - cend, 2754
 - clear, 2754
 - count, 2755
 - crbegin, 2755
 - crend, 2755
 - empty, 2755
 - end, 2756
 - equal_range, 2756, 2757
 - erase, 2757, 2758
 - find, 2759
 - get_allocator, 2759
 - insert, 2760, 2761
 - key_comp, 2761
 - lower_bound, 2762
 - max_size, 2763
 - multimap, 2751–2753
 - operator=, 2763, 2764
 - rbegin, 2764
 - rend, 2764, 2765
 - size, 2765
 - swap, 2765
 - upper_bound, 2766
 - value_comp, 2766
- std::multiplies, 2767
 - first_argument_type, 2768
 - result_type, 2768
 - second_argument_type, 2768
- std::multiset, 2768
 - begin, 2773
 - cbegin, 2773
 - cend, 2774
 - clear, 2774
 - count, 2774
 - crbegin, 2774
 - crend, 2775
 - empty, 2775
 - end, 2775
 - equal_range, 2775, 2776
 - erase, 2776, 2777
 - find, 2778
 - get_allocator, 2779
 - insert, 2779, 2780
 - key_comp, 2781
 - lower_bound, 2781
 - max_size, 2782
 - multiset, 2771–2773
 - operator<, 2785
 - operator=, 2782, 2783
 - operator==, 2785
 - rbegin, 2783

- rend, 2783
- size, 2783
- swap, 2784
- upper_bound, 2784
- value_comp, 2785
- std::mutex, 2786
- std::negate, 2787
 - argument_type, 2788
 - result_type, 2788
- std::negative_binomial_distribution, 2788
 - k, 2790
 - max, 2790
 - min, 2790
 - operator<<, 2792
 - operator>>, 2792
 - operator(), 2790
 - operator==, 2792
 - p, 2790
 - param, 2791
 - reset, 2791
 - result_type, 2790
- std::negative_binomial_ -
 - distribution::param_type, 2793
- std::nested_exception, 2793
- std::normal_distribution, 2794
 - max, 2796
 - mean, 2796
 - min, 2796
 - normal_distribution, 2796
 - operator<<, 2798
 - operator>>, 2799
 - operator(), 2797
 - operator==, 2799
 - param, 2797
 - reset, 2798
 - result_type, 2796
 - stddev, 2798
- std::normal_distribution::param_type, 2799
- std::not_equal_to, 2800
 - first_argument_type, 2802
 - result_type, 2802
 - second_argument_type, 2802
- std::num_get, 2802
 - ~num_get, 2806
- char_type, 2805
- do_get, 2806–2812
- get, 2813–2821
- id, 2822
- iter_type, 2805
- num_get, 2806
- std::num_put, 2822
 - ~num_put, 2826
 - char_type, 2825
 - do_put, 2826–2829
 - id, 2837
 - iter_type, 2825
 - num_put, 2826
 - put, 2830–2836
- std::numeric_limits, 2837
 - denorm_min, 2839
 - digits, 2840
 - digits10, 2841
 - epsilon, 2839
 - has_denorm, 2841
 - has_denorm_loss, 2841
 - has_infinity, 2841
 - has_quiet_NaN, 2841
 - has_signaling_NaN, 2841
 - infinity, 2839
 - is_bounded, 2841
 - is_exact, 2842
 - is_iec559, 2842
 - is_integer, 2842
 - is_modulo, 2842
 - is_signed, 2842
 - is_specialized, 2843
 - lowest, 2839
 - max, 2839
 - max_digits10, 2843
 - max_exponent, 2843
 - max_exponent10, 2843
 - min, 2840
 - min_exponent, 2843
 - min_exponent10, 2843
 - quiet_NaN, 2840
 - radix, 2844
 - round_error, 2840
 - round_style, 2844
 - signaling_NaN, 2840
 - tinyness_before, 2844

- traps, 2844
- std::numeric_limits< bool >, 2845
- std::numeric_limits< char >, 2846
- std::numeric_limits< char16_t >, 2847
- std::numeric_limits< char32_t >, 2848
- std::numeric_limits< double >, 2850
- std::numeric_limits< float >, 2851
- std::numeric_limits< int >, 2852
- std::numeric_limits< long >, 2854
- std::numeric_limits< long double >, 2855
- std::numeric_limits< long long >, 2856
- std::numeric_limits< short >, 2858
- std::numeric_limits< signed char >, 2859
- std::numeric_limits< unsigned char >, 2860
- std::numeric_limits< unsigned int >, 2862
- std::numeric_limits< unsigned long >, 2863
- std::numeric_limits< unsigned long long >, 2864
- std::numeric_limits< unsigned short >, 2866
- std::numeric_limits< wchar_t >, 2867
- std::numpunct, 2868
 - ~numpunct, 2872
 - char_type, 2871
 - decimal_point, 2872
 - do_decimal_point, 2873
 - do_falsename, 2873
 - do_grouping, 2873
 - do_thousands_sep, 2874
 - do_truename, 2874
 - falsename, 2875
 - grouping, 2875
 - id, 2876
 - numpunct, 2871, 2872
 - string_type, 2871
 - thousands_sep, 2875
 - truename, 2876
- std::numpunct_byname, 2877
 - char_type, 2879
 - decimal_point, 2879
 - do_decimal_point, 2879
 - do_falsename, 2880
 - do_grouping, 2880
 - do_thousands_sep, 2880
 - do_truename, 2881
 - falsename, 2881
 - grouping, 2881
 - id, 2883
 - string_type, 2879
 - thousands_sep, 2882
 - truename, 2882
- std::once_flag, 2883
- call_once, 2884
- std::ostream_iterator, 2884
 - char_type, 2885
 - difference_type, 2885
 - iterator_category, 2886
 - operator=, 2888
 - ostream_iterator, 2887
 - ostream_type, 2886
 - pointer, 2886
 - reference, 2886
 - traits_type, 2886
 - value_type, 2886
- std::ostreambuf_iterator, 2888
 - char_type, 2890
 - difference_type, 2890
 - failed, 2892
 - iterator_category, 2890
 - operator*, 2892
 - operator++, 2892
 - operator=, 2892
 - ostream_type, 2890
 - ostreambuf_iterator, 2891
 - pointer, 2890
 - reference, 2890
 - streambuf_type, 2890
 - traits_type, 2891
 - value_type, 2891
- std::out_of_range, 2893
- what, 2893
- std::output_iterator_tag, 2894
- std::overflow_error, 2895
- what, 2895
- std::owner_less< shared_ptr< _Tp > >, 2896
 - first_argument_type, 2896
 - result_type, 2896

- second_argument_type, 2897
- std::owner_less< weak_ptr< _Tp > >, 2897
 - first_argument_type, 2898
 - result_type, 2898
 - second_argument_type, 2898
- std::packaged_task< _Res(_- ArgTypes...)>, 2898
- std::pair, 2899
 - first, 2902
 - first_type, 2901
 - pair, 2901
 - second, 2902
 - second_type, 2901
- std::piecewise_constant_distribution, 2902
 - densities, 2904
 - intervals, 2904
 - max, 2904
 - min, 2904
 - operator<<, 2906
 - operator>>, 2906
 - operator(), 2904
 - param, 2905
 - reset, 2905
 - result_type, 2904
- std::piecewise_constant_distribution::param_type, 2907
- std::piecewise_linear_distribution, 2908
 - densities, 2909
 - intervals, 2909
 - max, 2910
 - min, 2910
 - operator<<, 2911
 - operator>>, 2911
 - operator(), 2910
 - param, 2910
 - reset, 2911
 - result_type, 2909
- std::piecewise_linear_distribution::param_type, 2912
- std::placeholders, 749
- std::plus, 2913
 - first_argument_type, 2914
 - result_type, 2914
 - second_argument_type, 2914
- std::pointer_to_binary_function, 2915
 - first_argument_type, 2916
 - result_type, 2916
 - second_argument_type, 2916
- std::pointer_to_unary_function, 2916
 - argument_type, 2918
 - result_type, 2918
- std::poisson_distribution, 2918
 - max, 2920
 - mean, 2920
 - min, 2920
 - operator<<, 2922
 - operator>>, 2922
 - operator(), 2920
 - operator==, 2922
 - param, 2921
 - reset, 2921
 - result_type, 2919
- std::poisson_distribution::param_type, 2923
- std::priority_queue, 2923
 - empty, 2926
 - pop, 2926
 - priority_queue, 2925
 - push, 2926
 - size, 2927
 - top, 2927
- std::promise, 2928
- std::promise< _Res & >, 2928
- std::promise< void >, 2929
- std::queue, 2930
 - back, 2932
 - c, 2934
 - empty, 2932
 - front, 2932
 - pop, 2933
 - push, 2933
 - queue, 2932
 - size, 2933
- std::random_access_iterator_tag, 2934
- std::random_device, 2935
 - result_type, 2936
- std::range_error, 2937
 - what, 2937

- std::ratio, 2938
- std::ratio_add, 2939
- std::ratio_divide, 2939
- std::ratio_equal, 2940
- std::ratio_multiply, 2940
- std::ratio_not_equal, 2941
- std::ratio_subtract, 2941
- std::raw_storage_iterator, 2942
 - difference_type, 2943
 - iterator_category, 2943
 - pointer, 2943
 - reference, 2943
 - value_type, 2944
- std::recursive_mutex, 2944
- std::recursive_timed_mutex, 2945
- std::reference_wrapper, 2945
- std::regex_constants, 750
 - __match_flag, 752
 - __syntax_option, 752
 - awk, 754
 - basic, 754
 - collate, 755
 - ECMAScript, 755
 - egrep, 755
 - error_backref, 753
 - error_badbrace, 753
 - error_badrepeat, 753
 - error_brace, 753
 - error_brack, 753
 - error_collate, 753
 - error_complexity, 753
 - error_ctype, 753
 - error_escape, 754
 - error_paren, 754
 - error_range, 754
 - error_space, 754
 - error_stack, 754
 - error_type, 752
 - extended, 755
 - format_default, 755
 - format_first_only, 756
 - format_no_copy, 756
 - format_sed, 756
 - grep, 757
 - icase, 757
 - match_any, 757
 - match_continuous, 757
 - match_default, 757
 - match_flag_type, 751
 - match_not_bol, 757
 - match_not_bow, 758
 - match_not_eol, 758
 - match_not_eow, 758
 - match_not_null, 758
 - match_prev_avail, 758
 - nosubs, 759
 - optimize, 759
 - syntax_option_type, 752
- std::regex_error, 2947
 - code, 2948
 - regex_error, 2948
 - what, 2948
- std::regex_iterator, 2949
 - operator*, 2951
 - operator++, 2951
 - operator->, 2952
 - operator=, 2952
 - operator==, 2952
 - regex_iterator, 2950
- std::regex_token_iterator, 2953
 - operator*, 2957
 - operator++, 2957
 - operator->, 2958
 - operator=, 2958
 - operator==, 2958
 - regex_token_iterator, 2954–2956
- std::regex_traits, 2959
 - getloc, 2960
 - imbue, 2960
 - length, 2961
 - lookup_classname, 2961
 - lookup_collatename, 2962
 - regex_traits, 2960
 - transform, 2963
 - transform_primary, 2964
 - translate, 2964
 - translate_nocase, 2964
- std::rel_ops, 759
 - operator<=, 760
 - operator>, 760
 - operator>=, 761
- std::remove_reference, 2965

- std::reverse_iterator, 2966
 - base, 2969
 - difference_type, 2967
 - iterator_category, 2967
 - operator*, 2969
 - operator+, 2970
 - operator++, 2970
 - operator+=, 2970
 - operator-, 2971
 - operator->, 2972
 - operator--, 2971
 - operator=, 2972
 - pointer, 2968
 - reference, 2968
 - reverse_iterator, 2968, 2969
 - value_type, 2968
- std::runtime_error, 2973
 - runtime_error, 2974
 - what, 2974
- std::seed_seq, 2974
 - result_type, 2975
 - seed_seq, 2975
- std::set, 2975
 - allocator_type, 2978
 - begin, 2984
 - cbegin, 2984
 - cend, 2984
 - clear, 2984
 - const_iterator, 2978
 - const_pointer, 2979
 - const_reference, 2979
 - const_reverse_iterator, 2979
 - count, 2985
 - crbegin, 2985
 - crend, 2985
 - difference_type, 2979
 - empty, 2985
 - end, 2986
 - equal_range, 2986
 - erase, 2987, 2988
 - find, 2988, 2989
 - get_allocator, 2989
 - insert, 2989–2991
 - iterator, 2979
 - key_comp, 2991
 - key_compare, 2980
 - key_type, 2980
 - lower_bound, 2991, 2992
 - max_size, 2992
 - operator=, 2992, 2993
 - pointer, 2980
 - rbegin, 2993
 - reference, 2980
 - rend, 2994
 - reverse_iterator, 2980
 - set, 2981–2983
 - size, 2994
 - size_type, 2981
 - swap, 2994
 - upper_bound, 2994, 2995
 - value_comp, 2995
 - value_compare, 2981
 - value_type, 2981
- std::shared_future, 2996
 - _M_get_result, 2998
 - get, 2998
 - shared_future, 2997
- std::shared_future< _Res & >, 2998
 - _M_get_result, 3001
 - get, 3001
 - shared_future, 3000
- std::shared_future< void >, 3001
 - _M_get_result, 3004
 - shared_future, 3003
- std::shared_ptr, 3004
 - allocate_shared, 3011
 - shared_ptr, 3006–3011
- std::shuffle_order_engine, 3012
 - base, 3015
 - discard, 3015
 - max, 3015
 - min, 3016
 - operator<<, 3017
 - operator>>, 3018
 - operator(), 3016
 - operator==, 3017
 - result_type, 3013
 - seed, 3016, 3017
 - shuffle_order_engine, 3014, 3015
- std::slice, 3018
- std::slice_array, 3019
 - operator<=, 3021

- operator>=, 3022
- operator*=, 3021
- operator^=, 3022
- operator+=, 3021
- operator-=, 3021
- operator/=, 3021
- operator%=, 3021
- operator&=, 3021
- std::stack, 3022
 - empty, 3025
 - pop, 3025
 - push, 3025
 - size, 3025
 - stack, 3024
 - top, 3025, 3026
- std::student_t_distribution, 3026
 - max, 3028
 - min, 3028
 - operator<<, 3029
 - operator>>, 3029
 - operator(), 3028
 - operator==, 3029
 - param, 3028
 - reset, 3029
 - result_type, 3027
- std::student_t_distribution::param_type, 3030
- std::sub_match, 3031
 - compare, 3033, 3034
 - first, 3035
 - first_type, 3033
 - length, 3034
 - operator string_type, 3034
 - second, 3035
 - second_type, 3033
 - str, 3035
- std::system_error, 3036
 - what, 3037
- std::this_thread, 761
 - get_id, 762
 - sleep_for, 762
 - sleep_until, 762
 - yield, 762
- std::thread, 3037
 - native_handle, 3038
- std::thread::id, 3038
- std::time_base, 3039
- std::time_get, 3040
 - ~time_get, 3043
 - char_type, 3043
 - date_order, 3044
 - do_date_order, 3044
 - do_get_date, 3044
 - do_get_monthname, 3045
 - do_get_time, 3045
 - do_get_weekday, 3046
 - do_get_year, 3047
 - get_date, 3047
 - get_monthname, 3048
 - get_time, 3049
 - get_weekday, 3049
 - get_year, 3050
 - id, 3051
 - iter_type, 3043
 - time_get, 3043
- std::time_get_byname, 3051
 - char_type, 3054
 - date_order, 3054
 - do_date_order, 3055
 - do_get_date, 3055
 - do_get_monthname, 3056
 - do_get_time, 3056
 - do_get_weekday, 3057
 - do_get_year, 3058
 - get_date, 3058
 - get_monthname, 3059
 - get_time, 3060
 - get_weekday, 3060
 - get_year, 3061
 - id, 3062
 - iter_type, 3054
- std::time_put, 3062
 - ~time_put, 3065
 - char_type, 3064
 - do_put, 3065
 - id, 3067
 - iter_type, 3064
 - put, 3066, 3067
 - time_put, 3065
- std::time_put_byname, 3068
 - char_type, 3069
 - do_put, 3070

- id, 3072
- iter_type, 3069
- put, 3070, 3071
- std::timed_mutex, 3072
- std::tr1, 762
- std::tr1::__detail, 770
- std::tr1::__is_member_pointer_helper, 3073
- std::tr1::add_const, 3074
- std::tr1::add_cv, 3074
- std::tr1::add_pointer, 3075
- std::tr1::add_volatile, 3075
- std::tr1::alignment_of, 3076
- std::tr1::array, 3077
- std::tr1::bad_weak_ptr, 3078
- what, 3079
- std::tr1::extent, 3080
- std::tr1::has_virtual_destructor, 3081
- std::tr1::integral_constant, 3082
- std::tr1::is_abstract, 3084
- std::tr1::is_arithmetic, 3085
- std::tr1::is_array, 3086
- std::tr1::is_class, 3087
- std::tr1::is_compound, 3088
- std::tr1::is_const, 3089
- std::tr1::is_empty, 3090
- std::tr1::is_enum, 3092
- std::tr1::is_floating_point, 3093
- std::tr1::is_function, 3094
- std::tr1::is_fundamental, 3095
- std::tr1::is_integral, 3095
- std::tr1::is_member_function_pointer, 3096
- std::tr1::is_member_object_pointer, 3097
- std::tr1::is_object, 3098
- std::tr1::is_pointer, 3099
- std::tr1::is_polymorphic, 3099
- std::tr1::is_same, 3101
- std::tr1::is_scalar, 3102
- std::tr1::is_union, 3102
- std::tr1::is_void, 3104
- std::tr1::is_volatile, 3104
- std::tr1::rank, 3105
- std::tr1::remove_all_extents, 3107
- std::tr1::remove_const, 3107
- std::tr1::remove_cv, 3108
- std::tr1::remove_extent, 3108
- std::tr1::remove_pointer, 3109
- std::tr1::remove_volatile, 3109
- std::try_to_lock_t, 3110
- std::tuple, 3110
- std::tuple< _T1 >, 3111
- std::tuple< _T1, _T2 >, 3112
- std::tuple_element< 0, tuple< _Head, _Tail...> >, 3113
- std::tuple_element< __i, tuple< _Head, _Tail...> >, 3114
- std::tuple_size< tuple< _Elements...> >, 3114
- std::type_info, 3115
- ~type_info, 3116
- name, 3116
- std::unary_function, 3117
- argument_type, 3118
- result_type, 3118
- std::unary_negate, 3118
- argument_type, 3120
- result_type, 3120
- std::underflow_error, 3121
- what, 3121
- std::uniform_int_distribution, 3122
- max, 3123
- min, 3123
- operator(), 3124
- param, 3124
- reset, 3124
- result_type, 3123
- uniform_int_distribution, 3123
- std::uniform_int_distribution::param_type, 3125
- std::uniform_real_distribution, 3126
- max, 3127
- min, 3127
- operator(), 3127
- param, 3128
- reset, 3128
- result_type, 3127
- uniform_real_distribution, 3127
- std::uniform_real_distribution::param_type, 3129
- std::unique_lock, 3129
- std::unique_ptr, 3131

- std::unordered_map, 3134
- std::unordered_multimap, 3136
- std::unordered_multiset, 3139
- std::unordered_set, 3141
- std::uses_allocator, 3144
- std::valarray, 3145
 - valarray, 3148
- std::vector, 3148
 - ~vector, 3155
 - _M_allocate_and_copy, 3155
 - _M_range_check, 3155
 - assign, 3156
 - at, 3157
 - back, 3158
 - begin, 3158
 - capacity, 3159
 - cbegin, 3159
 - cend, 3159
 - clear, 3159
 - crbegin, 3160
 - crend, 3160
 - data, 3160
 - emplace, 3160
 - empty, 3161
 - end, 3161
 - erase, 3161, 3162
 - front, 3162, 3163
 - insert, 3163–3165
 - max_size, 3165
 - operator=, 3165, 3166
 - pop_back, 3167
 - push_back, 3168
 - rbegin, 3168
 - rend, 3168
 - reserve, 3169
 - resize, 3169, 3170
 - shrink_to_fit, 3170
 - size, 3170
 - swap, 3170
 - vector, 3152–3154
- std::vector< bool, _Alloc >, 3171
- std::weak_ptr, 3175
- std::weibull_distribution, 3176
 - a, 3178
 - b, 3178
 - max, 3178
 - min, 3178
 - operator(), 3178
 - param, 3178, 3179
 - reset, 3179
 - result_type, 3177
- std::weibull_distribution::param_type, 3179
- stdatomic.h, 3507
- stddev
 - std::normal_distribution, 2798
- stdexcept, 3507
- stdio_filebuf
 - __gnu_cxx::stdio_filebuf, 957, 958
- stdio_filebuf.h, 3508
- stdio_sync_filebuf.h, 3509
- stl_algo.h, 3509
- stl_algobase.h, 3522
- stl_bvector.h, 3525
- stl_construct.h, 3526
- stl_deque.h, 3527
 - _GLIBCXX_DEQUE_BUF_SIZE, 3530
- stl_function.h, 3531
- stl_heap.h, 3533
- stl_iterator.h, 3535
- stl_iterator_base_funcs.h, 3539
- stl_iterator_base_types.h, 3540
- stl_list.h, 3542
- stl_map.h, 3543
- stl_multimap.h, 3544
- stl_multiset.h, 3545
- stl_numeric.h, 3546
- stl_pair.h, 3547
- stl_queue.h, 3549
- stl_raw_storage_iter.h, 3550
- stl_relops.h, 3550
- stl_set.h, 3551
- stl_stack.h, 3552
- stl_tempbuf.h, 3553
- stl_tree.h, 3554
- stl_uninitialized.h, 3555
- stl_vector.h, 3557
- stossc
 - __gnu_cxx::enc_filebuf, 903
 - __gnu_cxx::stdio_filebuf, 975

- __gnu_cxx::stdio_sync_filebuf, 1003
 - std::basic_filebuf, 1545
 - std::basic_streambuf, 2038
 - std::basic_stringbuf, 2121
- str
 - std::basic_istream, 1865
 - std::basic_ostringstream, 1997
 - std::basic_stringbuf, 2121, 2122
 - std::basic_stringstream, 2182
 - std::match_results, 2689
 - std::sub_match, 3035
- stream_iterator.h, 3559
- streambuf, 3559
 - io, 62
- streambuf.tcc, 3560
- streambuf_iterator.h, 3561
- streambuf_type
 - std::istreambuf_iterator, 2587
 - std::ostreambuf_iterator, 2890
- streamoff
 - std, 604
- streampos
 - std, 604
- streamsize
 - std, 604
- stride
 - numeric_arrays, 109, 110
- string, 3562
 - strings, 278
- string_type
 - std::collate, 2274
 - std::collate_byname, 2281
 - std::messages, 2700
 - std::messages_byname, 2705
 - std::money_get, 2713
 - std::money_put, 2719
 - std::moneypunct, 2725
 - std::moneypunct_byname, 2738
 - std::numpunct, 2871
 - std::numpunct_byname, 2879
- stringbuf
 - io, 62
- stringfwd.h, 3565
- Strings, 277
- strings
 - string, 278
 - u16string, 278
 - u32string, 278
 - wstring, 278
- stringstream
 - io, 62
- substr
 - __gnu_cxx::__versa_string, 858
 - __gnu_debug::basic_string, 1101
 - std::basic_string, 2101
- subtractive_rng
 - __gnu_cxx::subtractive_rng, 1010
- suffix
 - std::match_results, 2689
- sum
 - numeric_arrays, 110
- sungetc
 - __gnu_cxx::enc_filebuf, 903
 - __gnu_cxx::stdio_filebuf, 976
 - __gnu_cxx::stdio_sync_filebuf, 1003
 - std::basic_filebuf, 1545
 - std::basic_streambuf, 2038
 - std::basic_stringbuf, 2122
- swap
 - __gnu_cxx, 376
 - __gnu_cxx::__versa_string, 859
 - __gnu_debug::basic_string, 1101
 - mutating_algorithms, 148
 - regex, 252, 253
 - std, 688–690
 - std::basic_regex, 2018
 - std::basic_string, 2102
 - std::deque, 2414
 - std::forward_list, 2471
 - std::function< _Res(_- ArgTypes...)>, 2485
 - std::list, 2627
 - std::map, 2674
 - std::match_results, 2689
 - std::multimap, 2765
 - std::multiset, 2784
 - std::set, 2994
 - std::vector, 3170
- swap_ranges
 - mutating_algorithms, 148

-
- sync
 - __gnu_cxx::enc_filebuf, 903
 - __gnu_cxx::stdio_filebuf, 976
 - __gnu_cxx::stdio_sync_filebuf, 1003
 - std::basic_filebuf, 1546
 - std::basic_fstream, 1609
 - std::basic_ifstream, 1664
 - std::basic_iostream, 1758
 - std::basic_istream, 1810
 - std::basic_istreamstream, 1865
 - std::basic_streambuf, 2039
 - std::basic_stringbuf, 2123
 - std::basic_stringstream, 2183
 - sync_with_stdio
 - std::basic_fstream, 1610
 - std::basic_ifstream, 1664
 - std::basic_ios, 1696
 - std::basic_iostream, 1759
 - std::basic_istream, 1811
 - std::basic_istreamstream, 1866
 - std::basic_ofstream, 1910
 - std::basic_ostream, 1952
 - std::basic_ostreamstream, 1997
 - std::basic_stringstream, 2183
 - std::ios_base, 2561
 - syntax_option_type
 - std::regex_constants, 752
 - system_error, 3565
 - t
 - std::binomial_distribution, 2209
 - table
 - std::ctype< char >, 2323
 - std::ctype_byname< char >, 2373
 - table_size
 - std::ctype< char >, 2326
 - std::ctype_byname< char >, 2376
 - tag_and_trait.hpp, 3567
 - tags.h, 3569
 - tan
 - complex_numbers, 53
 - tanh
 - complex_numbers, 53
 - target
 - std::function< _Res(_ArgTypes...)>, 2485, 2486
 - target_type
 - std::function< _Res(_ArgTypes...)>, 2486
 - tellg
 - std::basic_fstream, 1610
 - std::basic_ifstream, 1665
 - std::basic_iostream, 1759
 - std::basic_istream, 1811
 - std::basic_istreamstream, 1866
 - std::basic_stringstream, 2183
 - tellp
 - std::basic_fstream, 1611
 - std::basic_iostream, 1759
 - std::basic_ofstream, 1910
 - std::basic_ostream, 1953
 - std::basic_ostreamstream, 1998
 - std::basic_stringstream, 2184
 - temporary_buffer
 - __gnu_cxx::temporary_buffer, 1012
 - terminate
 - exceptions, 37
 - terminate_handler
 - exceptions, 35
 - test
 - std::bitset, 2223
 - tgmath.h, 3571
 - thousands_sep
 - std::moneypunct, 2734
 - std::moneypunct_byname, 2746
 - std::numpunct, 2875
 - std::numpunct_byname, 2882
 - thread, 3571
 - Threads, 75
 - throw_allocator.h, 3572
 - throw_with_nested
 - exceptions, 37
 - tie
 - std::basic_fstream, 1611
 - std::basic_ifstream, 1665
 - std::basic_ios, 1697
 - std::basic_iostream, 1760
 - std::basic_istream, 1812
 - std::basic_istreamstream, 1867
 - std::basic_ofstream, 1911
-

- std::basic_ostream, 1953
 - std::basic_ostringstream, 1998, 1999
 - std::basic_stringstream, 2184, 2185
- Time, 38
- time
 - std::locale, 2638
- time_get
 - std::time_get, 3043
- time_members.h, 3575
- time_point_cast
 - std::chrono, 738
- time_put
 - std::time_put, 3065
- tinyness_before
 - std::__numeric_limits_base, 1413
 - std::numeric_limits, 2844
- TLB_size
 - __gnu_parallel::_Settings, 1225
- to_string
 - std::bitset, 2223
- to_ulong
 - std::bitset, 2223
- tolower
 - std, 690
 - std::__ctype_abstract_base, 1333
 - std::ctype, 2309, 2310
 - std::ctype< char >, 2323, 2324
 - std::ctype< wchar_t >, 2345
 - std::ctype_byname, 2360, 2361
 - std::ctype_byname< char >, 2373, 2374
- top
 - std::priority_queue, 2927
 - std::stack, 3025, 3026
- toupper
 - std, 690
 - std::__ctype_abstract_base, 1333, 1334
 - std::ctype, 2310, 2311
 - std::ctype< char >, 2324, 2325
 - std::ctype< wchar_t >, 2345, 2346
 - std::ctype_byname, 2361, 2362
 - std::ctype_byname< char >, 2374, 2375
- tr1_math_spec_func
 - assoc_laguerre, 113
 - assoc_legendre, 113
 - beta, 114
 - comp_ellint_1, 114
 - comp_ellint_2, 114
 - comp_ellint_3, 114
 - conf_hyperg, 114
 - cyl_bessel_i, 114
 - cyl_bessel_j, 115
 - cyl_bessel_k, 115
 - cyl_neumann, 115
 - ellint_1, 115
 - ellint_2, 115
 - ellint_3, 116
 - expint, 116
 - hermite, 116
 - hyperg, 116
 - laguerre, 116
 - legendre, 116
 - riemann_zeta, 117
 - sph_bessel, 117
 - sph_legendre, 117
 - sph_neumann, 117
- traits_type
 - __gnu_cxx::enc_filebuf, 889
 - __gnu_cxx::stdio_filebuf, 957
 - __gnu_cxx::stdio_sync_filebuf, 989
 - std::basic_filebuf, 1528
 - std::basic_fstream, 1568, 1569
 - std::basic_ifstream, 1632
 - std::basic_ios, 1683
 - std::basic_iostream, 1719
 - std::basic_istream, 1780
 - std::basic_istream::sentry, 1822
 - std::basic_istreamstream, 1834
 - std::basic_ofstream, 1887
 - std::basic_ostream, 1930
 - std::basic_ostringstream, 1975
 - std::basic_streambuf, 2024
 - std::basic_stringbuf, 2107
 - std::basic_stringstream, 2142, 2143
 - std::istreambuf_iterator, 2587
 - std::ostream_iterator, 2886
 - std::ostreambuf_iterator, 2891
- transform
 - mutating_algorithms, 148, 149
 - std::collate, 2277

- std::collate_byname, 2283
- std::regex_traits, 2963
- transform_minimal_n
 - __gnu_parallel::_Settings, 1225
- transform_primary
 - std::regex_traits, 2964
- translate
 - std::regex_traits, 2964
- translate_nocase
 - std::regex_traits, 2964
- traps
 - std::__numeric_limits_base, 1413
 - std::numeric_limits, 2844
- tree_policy.hpp, 3575
- tree_trace_base.hpp, 3576
- trie_policy.hpp, 3576
- true_type
 - metaprogramming, 124
- truename
 - std::numpunct, 2876
 - std::numpunct_byname, 2882
- trunc
 - std::basic_fstream, 1621
 - std::basic_ifstream, 1674
 - std::basic_ios, 1705
 - std::basic_iostream, 1769
 - std::basic_istream, 1820
 - std::basic_istreamstream, 1876
 - std::basic_ofstream, 1920
 - std::basic_ostream, 1962
 - std::basic_ostreamstream, 2007
 - std::basic_stringstream, 2194
 - std::ios_base, 2569
- try_lock
 - mutexes, 72
- tuple, 3576
- Type Traits, 118
- type_traits, 3579, 3582
- type_traits.h, 3585
- type_utils.hpp, 3586
- typeinfo, 3586
- typelist.h, 3587
- types.h, 3588
- types_traits.hpp, 3589
- u16streampos
 - std, 605
- u16string
 - strings, 278
- u32streampos
 - std, 605
- u32string
 - strings, 278
- uflow
 - __gnu_cxx::enc_filebuf, 904
 - __gnu_cxx::stdio_filebuf, 976
 - __gnu_cxx::stdio_sync_filebuf, 1004
 - std::basic_filebuf, 1546
 - std::basic_streambuf, 2039
 - std::basic_stringbuf, 2123
- uncaught_exception
 - exceptions, 37
- underflow
 - __gnu_cxx::enc_filebuf, 904
 - __gnu_cxx::stdio_filebuf, 977
 - __gnu_cxx::stdio_sync_filebuf, 1004
 - std::basic_filebuf, 1547
 - std::basic_streambuf, 2039
 - std::basic_stringbuf, 2123
- unexpected
 - exceptions, 38
- unexpected_handler
 - exceptions, 35
- unget
 - std::basic_fstream, 1612
 - std::basic_ifstream, 1666
 - std::basic_iostream, 1761
 - std::basic_istream, 1812
 - std::basic_istreamstream, 1867
 - std::basic_stringstream, 2185
- Uniform, 288
- uniform_int_distribution
 - std::uniform_int_distribution, 3123
- uniform_real_distribution
 - std::uniform_real_distribution, 3127
- uninitialized_copy
 - std, 691
- uninitialized_copy_n
 - SGIextensions, 25
 - std, 691

- uninitialized_fill
 - std, 691
- uninitialized_fill_n
 - std, 692
- unique
 - mutating_algorithms, 150
 - std::forward_list, 2471
 - std::list, 2628
- unique_copy
 - mutating_algorithms, 151
- unique_copy.h, 3590
- unique_copy_minimal_n
 - __gnu_parallel::_Settings, 1225
- unique_ptr.h, 3590
- unitbuf
 - std, 692
 - std::basic_fstream, 1621
 - std::basic_ifstream, 1674
 - std::basic_ios, 1705
 - std::basic_iostream, 1770
 - std::basic_istream, 1821
 - std::basic_istreamstream, 1876
 - std::basic_ofstream, 1920
 - std::basic_ostream, 1962
 - std::basic_ostreamstream, 2007
 - std::basic_stringstream, 2194
 - std::ios_base, 2569
- Unordered Associative, 30
- unordered_map, 3592, 3593
- unordered_map.h, 3595
- unordered_set, 3597, 3598
- unordered_set.h, 3599
- unsetf
 - std::basic_fstream, 1612
 - std::basic_ifstream, 1666
 - std::basic_ios, 1697
 - std::basic_iostream, 1761
 - std::basic_istream, 1813
 - std::basic_istreamstream, 1868
 - std::basic_ofstream, 1912
 - std::basic_ostream, 1954
 - std::basic_ostreamstream, 1999
 - std::basic_stringstream, 2186
 - std::ios_base, 2562
- unshift
 - std::__codecvt_abstract_base, 1320
 - std::codecvt, 2247
 - std::codecvt< _InternT, _ExternT, encoding_state >, 2253
 - std::codecvt< char, char, mbstate_t >, 2258
 - std::codecvt< wchar_t, char, mbstate_t >, 2263
 - std::codecvt_byname, 2270
- upper_bound
 - binary_search_algorithms, 201, 202
 - std::map, 2675
 - std::multimap, 2766
 - std::multiset, 2784
 - std::set, 2994, 2995
- uppercase
 - std, 692
 - std::basic_fstream, 1621
 - std::basic_ifstream, 1674
 - std::basic_ios, 1705
 - std::basic_iostream, 1770
 - std::basic_istream, 1821
 - std::basic_istreamstream, 1876
 - std::basic_ofstream, 1920
 - std::basic_ostream, 1962
 - std::basic_ostreamstream, 2007
 - std::basic_stringstream, 2194
 - std::ios_base, 2569
- use_facet
 - std, 693
 - std::locale, 2636
 - std::locale::id, 2642
- Utilities, 77
- utility, 3601
- valarray, 3602
 - numeric_arrays, 90, 91
 - std::valarray, 3148
- valarray_after.h, 3608
- valarray_array.h, 3622
- valarray_array.tcc, 3633
- valarray_before.h, 3634
- value
 - regex, 253
- value_comp
 - std::map, 2675
 - std::multimap, 2766

- [std::multiset](#), 2785
 - [std::set](#), 2995
- [value_compare](#)
 - [std::set](#), 2981
- [value_type](#)
 - [std::back_insert_iterator](#), 1516
 - [std::complex](#), 2286
 - [std::front_insert_iterator](#), 2477
 - [std::insert_iterator](#), 2548
 - [std::istream_iterator](#), 2583
 - [std::istreambuf_iterator](#), 2587
 - [std::iterator](#), 2591
 - [std::ostream_iterator](#), 2886
 - [std::ostreambuf_iterator](#), 2891
 - [std::raw_storage_iterator](#), 2944
 - [std::reverse_iterator](#), 2968
 - [std::set](#), 2981
- [vector](#), 3634, 3635
 - [std::__debug::vector](#), 1395
 - [std::vector](#), 3152–3154
- [vector.tcc](#), 3637
- [vstring.h](#), 3637
- [vstring.tcc](#), 3641
- [vstring_fwd.h](#), 3642
- [vstring_util.h](#), 3643
- [wcerr](#)
 - [std](#), 694
- [wcin](#)
 - [std](#), 694
- [wclog](#)
 - [std](#), 695
- [wcout](#)
 - [std](#), 695
- [wcregex_token_iterator](#)
 - [regex](#), 224
- [wcsub_match](#)
 - [regex](#), 224
- [wfilebuf](#)
 - [io](#), 62
- [wfstream](#)
 - [io](#), 62
- [what](#)
 - [__gnu_cxx::forced_error](#), 914
 - [__gnu_cxx::recursive_init_error](#), 939
- [std::bad_alloc](#), 1518
- [std::bad_cast](#), 1520
- [std::bad_exception](#), 1521
- [std::bad_function_call](#), 1522
- [std::bad_typeid](#), 1523
- [std::domain_error](#), 2428
- [std::exception](#), 2436
- [std::future_error](#), 2496
- [std::invalid_argument](#), 2550
- [std::ios_base::failure](#), 2570
- [std::length_error](#), 2593
- [std::logic_error](#), 2644
- [std::out_of_range](#), 2893
- [std::overflow_error](#), 2895
- [std::range_error](#), 2937
- [std::regex_error](#), 2948
- [std::runtime_error](#), 2974
- [std::system_error](#), 3037
- [std::tr1::bad_weak_ptr](#), 3079
- [std::underflow_error](#), 3121
- [widen](#)
 - [std::__ctype_abstract_base](#), 1334, 1335
 - [std::basic_fstream](#), 1613
 - [std::basic_ifstream](#), 1667
 - [std::basic_ios](#), 1698
 - [std::basic_iostream](#), 1761
 - [std::basic_istream](#), 1813
 - [std::basic_istreamstream](#), 1868
 - [std::basic_ofstream](#), 1912
 - [std::basic_ostream](#), 1954
 - [std::basic_ostringstream](#), 1999
 - [std::basic_stringstream](#), 2186
 - [std::ctype](#), 2311, 2312
 - [std::ctype< char >](#), 2325, 2326
 - [std::ctype< wchar_t >](#), 2346, 2347
 - [std::ctype_byname](#), 2362, 2363
 - [std::ctype_byname< char >](#), 2375, 2376
- [width](#)
 - [std::basic_fstream](#), 1613
 - [std::basic_ifstream](#), 1667
 - [std::basic_ios](#), 1698
 - [std::basic_iostream](#), 1762
 - [std::basic_istream](#), 1814
 - [std::basic_istreamstream](#), 1869

- std::basic_ofstream, 1912, 1913
- std::basic_ostream, 1955
- std::basic_ostringstream, 2000
- std::basic_stringstream, 2186, 2187
- std::ios_base, 2562
- wifstream
 - io, 63
- wios
 - io, 63
- wiostream
 - io, 63
- wistream
 - io, 63
- wistringstream
 - io, 63
- wofstream
 - io, 63
- workstealing.h, 3643
- wostream
 - io, 63
- wostringstream
 - io, 64
- wregex
 - regex, 224
- write
 - std::basic_fstream, 1614
 - std::basic_istream, 1763
 - std::basic_ofstream, 1913
 - std::basic_ostream, 1955
 - std::basic_ostringstream, 2000
 - std::basic_stringstream, 2187
- ws
 - std, 693
- wsregex_token_iterator
 - regex, 224
- wssub_match
 - regex, 224
- wstreambuf
 - io, 64
- wstreampos
 - std, 605
- wstring
 - strings, 278
- wstringbuf
 - io, 64
- wstringstream
 - io, 64
- xalloc
 - std::basic_fstream, 1614
 - std::basic_ifstream, 1668
 - std::basic_ios, 1699
 - std::basic_istream, 1763
 - std::basic_istream, 1814
 - std::basic_istream, 1869
 - std::basic_ofstream, 1914
 - std::basic_ostream, 1956
 - std::basic_ostringstream, 2001
 - std::basic_stringstream, 2188
 - std::ios_base, 2563
- xsgetn
 - __gnu_cxx::enc_filebuf, 905
 - __gnu_cxx::stdio_filebuf, 978
 - __gnu_cxx::stdio_sync_filebuf, 1005
 - std::basic_filebuf, 1547
 - std::basic_streambuf, 2040
 - std::basic_stringbuf, 2124
- xspn
 - __gnu_cxx::enc_filebuf, 905
 - __gnu_cxx::stdio_filebuf, 978
 - __gnu_cxx::stdio_sync_filebuf, 1005
 - std::basic_filebuf, 1548
 - std::basic_streambuf, 2041
 - std::basic_stringbuf, 2125
- yield
 - std::this_thread, 762