precattl — Prepare special catcodes from token list [*]

user202729

Released 2022/07/07

Abstract

Allow users to write code that contains tokens with unusual catcodes.

1 Motivation

This package allows developers to quickly prototype writing code by specifying the exact token list to be executed.

For example, classically if you want to define a space gobbler you would do this

```
\lowercase{\def\spacegobbler} {}
```

which is equivalent to the token list `\def\spacegobbler`⟨*explicit space token*⟩`{}`.

In such simple cases there's negligible benefit in using this package; nevertheless in some more complex cases such as the following...

```
\begingroup
\lccode ‘?=‘_
\lowercase{
    \endgroup
    \peek_analysis_map_inline:n {
        \expandafter \string #1
        \peek_analysis_map_break:
    }
    \c_begin_group_token ?
}

\lowercase{\lowercase{\def\twospacegobbler} } {}

\catcode‘\^^M\active
\edef^^M{\string^^M}
```

it would be beneficial to have simpler construct to express the code.

A side benefit is that there's no need to do explicit `\begingroup` ... `\endgroup` (and keep track of when `\endgroup` should be executed), and no risk of accidentally pretokenizing the token list/it can be used inside argument of anything.

---

[*] This file describes version v0.0.0, last revised 2022/07/07.

2 Limitation

- There's some performance hit. I think it's about 5 times slower than `\tl_analysis_map_inline:nn` on the same token list.

  Nevertheless this is not a big problem, as it's possible to precompile the token list and include it in the generated `.sty` file.

- It's not allowed to pass `\outer` tokens into the macro directly; however, generating `\outer` tokens with `\cC` and `\cA` is supported.

3 Syntax

`\precattl_exec:n`

`\precattl_exec:n {⟨token list⟩}`

Executes the `{⟨token list⟩}` after preprocessing it in the following manner:

- `\cC{⟨control sequence name⟩}` is replaced by that control sequence.
- `\cFrozenRelax` is replaced with the frozen relax control sequence.
- `\cA{⟨tokens⟩}`, `\cO{⟨tokens⟩}`, etc. are replaced with sequence of tokens with catcode active/other respectively.

In particular, `{⟨tokens⟩}` or `{⟨control sequence name⟩}` might consist of normal "characters" or control sequences. Control sequences will have its csname appended. (nevertheless don't use the null control sequence here)

The full list of supported catcodes is the same as l3regex package: `\cB`, `\cE`, `\cM`, `\cT`, `\cP`, `\cU`, `\cD`, `\cS`, `\cL`, `\cO`, `\cA`.

As an example, the examples above can be executed as following:

```
\precattl_exec:n {
    \def\spacegobbler\cS\ {}

    \peek_analysis_map_inline:n {
        \expandafter \string #1
        \peek_analysis_map_break:
    }
    \c_begin_group_token \cO\_

    \def\twospacegobbler\cS\ \cS\ {}

    \def \cA\^^M {\cO\^^M}
}
```

More details: the token list following one of the `\c⟨character⟩` token might either be a single token, or a braced group. In the latter case their string representation without any `\escapechar` will be concatenated together.

For example:

- `\cL{12~34}` in `\ExplSyntaxOn` mode results in the 5 tokens with catcode letter and char code 1, 2, ⟨*space*⟩, 3, 4 respectively.

- `\cO\abc` results in 3 tokens `abc` with catcode other.

- `\cO{\ab\cd\ef}` results in 6 tokens `abcdef`. Note that this is different from the `\detokenized` value (that is, the spaces are removed) even when `\escapechar=-1`.

- `\cC{\ab\cd\ef}` results in the control sequence `\abcdef`.

- `\cO\\` results in a single token with char code \ and catcode other.

- `\cA\^^M` results in a single token with catcode active and char code 13. (because normal TEX processing rules transform the token `\^^M` to a control sequence with name length $= 1$ before passing it to the function `\precattl_exec:n`)

Remark: don't put explicit space token right after a `\cO` or similar. (nevertheless this is nontrivial to trigger because of how TEX works normally)

Warning: a \␣ at the end of a line will be interpreted as an escape sequence containing the `\endlinechar`.

`\precattlExec` LATEX2-style synonym for the function above.

`\precattl_set:Nn`

`\precattl_set:Nn` ⟨*tl var*⟩ {⟨*token list*⟩}

Same as above, but instead of executing the processed {⟨*token list*⟩}, the result is stored into ⟨*tl var*⟩.

Mostly equivalent to `\precattl_exec:n {\tl_set:Nn` ⟨*tl var*⟩ {⟨*token list*⟩}}.

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.