prettytok — Pretty-print token lists*

user202729

Released 2023/04/18

Abstract

Pretty-print token lists for debugging purposes.
Can be used to replace `\tl_analysis_show:n`.

1 Motivation

TEX/LATEX's default mechanism for debug-printing – that is, the following:

- `\showtokens{...}`,
- `\show...`,
- `\typeout{\unexpanded{...}}`,
- `\tl_show:n`,
- etc.

has a few limitations:

- `\show` and similar is considered an error and stops the TEX run. (this point is partially fixable, see the package writer's question https://tex.stackexchange.com/q/621919/250119)
- If there's some unprintable character in the output (for example, `^^J`, `^^M`, `^^I` – literal tab character), it's not easy to distinguish between them.
- If some token has unexpected catcode (most commonly, letter versus other), it's not easy to distinguish as well.
- They does not work in expansion-only context.

  (apart from `\msg_expandable_error:nn`, but this one suffers from the first problem as well)

`\tl_analysis_show:n` attempts to fix the third problem, but is very, very verbose and does not fix the other problems.

This package aims to fix all of them.

(although the expandable debug printing functions are LuaTEX-only.)

And some more additional (expandable in LuaTEX) functions to inspect the content of the input stream at a particular moment in time.

*This file describes version v0.2.0, last revised 2023/04/18.

2 Usage

2.1 A complete example

For a full example, the following code, which prints several things, both using the LaTeX 2_ε interface and using the expl3-style interface, using both expandable and unexpandable command (the former requires LuaTeX, as mentioned in 2.5):

```
\documentclass{article}
\usepackage{prettytok}

\prettyN{123{4 5}\6}
\prettyN{#&^_:}
\edef\mytest{\prettyeN{\error}}

\prettyN {very long long long long long long long long long long long
↪  long long long argument \argument\argument\test123456}
\prettyeN {very long long long long long long long long long long long
↪  long long long argument \argument\argument\test123456}

\ExplSyntaxOn
\pretty:n {#&^_:}
\ExplSyntaxOff

\begin {document}
\prettyW abc \prettystop
\end {document}
```

The corresponding output (in the current version) is:

```
> 123{4␣5}\6
> #&^_:
> \error
> very␣long␣long␣long␣long␣long␣long␣long␣long␣long␣long␣long␣long␣long
> .. ␣long␣argument␣\argument \argument \test 123456
> very␣long␣long␣long␣long␣long␣long␣long␣long␣long␣long␣long␣long␣long
> .. ␣long␣argument␣\argument \argument \test 123456
> #&^_:
> abc␣
```

2.2 Loading the package

```
\usepackage{prettytok}
```

2.3 Options

There are several options that can be passed to the package.

Usage example: `\usepackage[mode=term-shell]{prettytok}`.

`mode=` Specify the working mode of the package, that is, where the output is displayed. It can either be:

- `term-8bit`: this is the default.

  Assume XTerm-compatible system, output to the terminal.

  Requires `-8bit` option on engines other than LuaTeX, see the following link: https://tex.stackexchange.com/q/168460/250119.

  Besides, this drops the distinction between the catcode of some tokens (for example `{}$#&^_` are all shown as the same color as "special catcode"), which is available in the tooltip in the HTML version.

  This might work on Windows, although the package writer have not tested it. Refer to https://stackoverflow.com/q/2048509/5267751 and https://superuser.com/q/413073/577463.

- `term-shell`: To output colored text to the terminal, `--8bit` flag is needed otherwise the terminal escape codes will be changed to `^^[` etc. A workaround, using TeX Live's behavior when the file name has the form `|...`, is provided with this option.

  Requires `--shell-escape` flag. May not be very reliable.

  Refer to https://tex.stackexchange.com/a/670572/250119 for more details.

- `html`: output to a HTML file named for example `pretty-abc.html` (although this can be customized, refer to `html-file-name=`) if the main TeX file is named `abc.tex`.

  Open the file in any browser to view the result.

  Using this option, the output will not be cluttered with the traceback/other TeX default output.

  By default, the output refreshes whenever the TeX file is recompiled. The behavior can be customized with `html-refresh-strategy=` and `html-refresh-duration=`.

Currently, it's not supported to print the debug output to the PDF itself, because if the TeX program stops with error / has some error that corrupts the PDF output, the output will even with corrupted more by the debug print.

2.3.1 HTML configuration

These options are only meaningful if `mode=html`.

`html-file-name=` The output file name. Defaults to `pretty-`⟨*jobname*⟩`.html`, as mentioned above.

`html-refresh-strategy=` The auto-refresh strategy. Allowed values are 0-4. 0 is no refresh.
Which value works best depends on the particular browser.
If you're using Google Chrome to view the output HTML, invoking the browser from the command-line with `--allow-file-access-from-files` flag may be useful.

`html-refresh-duration=` The duration between two consecutive refresh check, in milliseconds. Defaults to 1000.

2.3.2 Terminal configuration

These options are only meaningful if `mode=term-8bit` or `mode=term-shell`.

`term-prefix=`
`term-prefix-more=`

Strings consist of prefixes to be output before each terminal line.

This might be useful for log-filtering/output-filtering tools such as `texfot` to recognize the output line.

Defaults to `>␣` and `>␣..␣` respectively.

`term-wrap-limit=` Estimated line length limit. Set this to a little smaller than your terminal width.

Defaults to 70.

`term-shell-decode-cmd=` Only meaningful with `mode=term-shell`.

Normally you would not need to explicitly pass this option, unless something does not work.

By default, a file named `prettytok-decode-8bit.py` should be included in your TEX distribution, and the package runs `kpsewhich prettytok-decode-8bit.py` to find the location of that file in order to execute it. However, if by any reason this does not work, you can specify an explicit command such as `python3 /full/path/to/prettytok-decode-8bit.py` to override it.

Passing blank value invokes the default behavior (runs `kpsewhich`).

Alternatively, you can also choose to explicitly pass the path in order to save a call to `kpsewhich` to make the program a bit faster.

If you *really* want to, special characters may be passed by prefixing them with `\`. But `\"` won't work anyway (as far as the package writer know, this is impossible in non-LuaTEX engines).

TEXhackers note: The path is interpreted by detokenizing the value in `escapechar=-1`.

`term-shell-decode-cmd-print=`

If `mode=term-shell`, print out the command correspond to `term-shell-decode-cmd` on the console, for debugging purpose.

Example output: `The value of term-shell-decode-cmd is: [[[./prettytok-decode-8bit.py]]]`

2.4 Main function

`\pretty:n`
`\pretty:(x|o|V)`

`\pretty:n {`⟨*token list*⟩`}`

Print the content of ⟨*token list*⟩.

This is a simple replacement of the functions above. (`\tl_analysis_show:n`, etc.)

\pretty:w `\pretty:w ⟨token list⟩ \prettystop`

Print the content of ⟨*token list*⟩.

The purpose of this function is that it can be inserted "anywhere" in order to *inspect the input stream* without affecting how the function works.

Note that the input stream will be tokenized and has catcode frozen.

For example

```
\ExplSyntaxOn
\def \f #1 {\prettye:w 789}
\f 123456 \prettystop
```

will print out `78923456`.

For now, it must be brace-balanced. Use `\prettye:w` instead if this is a problem.

\prettystop ⋆ `\prettystop`

Only used as a delimiter for `:w` functions. For convenience, this function is defined to do nothing.

\prettyshow:N `\prettyshow:N ⟨token⟩`
\prettyshow:c `\prettyshow:c {⟨control sequence name⟩}`

Show the meaning of a `N`-type argument.

\pretty:N `\pretty:N ⟨token⟩`
\pretty:c `\pretty:c {⟨control sequence name⟩}`

Print ⟨*token*⟩.

It may also be useful to use `\pretty:V` to print a token list variable's value, or `\prettyshow:N` to print a control sequence's meaning.

\pretty:nn `\pretty:nn {⟨token list⟩} {⟨token list⟩}`
\pretty:nnn `\pretty:nnn {⟨token list⟩} {⟨token list⟩} {⟨token list⟩}`

Print multiple token lists. Its effect is similar to multiple consecutive calls to `\pretty:n`.

2.5 Expandable interface (LuaTEX only)

\prettye:n ⋆ `\prettye:n {⟨token list⟩}`

Print the token list, similar to `\pretty:n`, but is fully expandable.

\prettye:w ⋆ `\prettye:w ⟨tokens⟩ \prettystop`

Print the tokens until `\prettystop` is seen. Useful for inspecting the content of the input stream.

As a debugging tool, it's possible to execute `\everyeof{\prettystop}` to avoid runaway printing in weird catcode environments.

Currently some implementation details (it can be fixed, but the package writer does not have an immediate need for it, see https://tex.stackexchange.com/q/335994/250119) means control sequences not in the hash table will be destroyed. Use with care.

`\prettye:nn` ⋆ `\prettye:nn {⟨token list⟩} {⟨token list⟩}`
`\prettye:nnn` ⋆ `\prettye:nnn {⟨token list⟩} {⟨token list⟩} {⟨token list⟩}`

Similar to multiple consecutive calls to `\prettye:n`.

`\prettye:nw` ⋆ `\prettye:nw {⟨callback⟩} ⟨tokens⟩ \prettystop`

Same as above, but has a callback, that is, code that is put in the input stream after the content is printed.

Useful if you want to fine-tune what is printed exactly. (`\prettye:w ⟨callback⟩ ⟨tokens⟩` is functionally the same, but the callback is also printed, which will clutter the debug output)

For example

```
1 \prettye:nw {\somecode ...} 123 \prettystop
```

will print `123`, then after some expansion steps results in the input stream contain `\somecode ... 123 \prettystop`.

`\prettye:nnw` ⋆ `\prettye:nnw {⟨callback⟩} {⟨token list⟩} ⟨tokens⟩ \prettystop`
`\prettye:nnnw` ⋆ `\prettye:nnnw {⟨callback⟩} {⟨token list⟩} {⟨token list⟩} ⟨tokens⟩ \prettystop`

Similar to call(s) to `\prettye:n` followed by a call to `\prettye:nw`.

For example

```
1 \prettye:nnw {\somecode ...} {123} 456 \prettystop
```

will print `123456`, then, after some expansion steps, `\somecode ... 456 \prettystop` remains in the input stream.

2.6 Lua programming interface

`prettyprint` `prettyprint(⟨content⟩)`

Print the content, which should be a table of token objects.

For convenience, you can pass multiple arguments. Strings are also supported.

2.7 LaTeX 2ε interface

`\prettyN` `\prettyN {⟨token list⟩}`
`\prettyX` `\prettyX {⟨token list⟩}`
`\prettyO` `\prettyO {⟨token list⟩}`
`\prettyV` `\prettyV ⟨tl var⟩`
`\prettyW` `\prettyW ⟨token list⟩ \prettystop`
`\prettyshowN` `\prettyshowN ⟨token⟩`
`\prettyshowC` `\prettyshowC {⟨control sequence name⟩}`

Alias of the correspondingly-named commands.

`\prettyeN` ⋆ `\prettyeN {⟨token list⟩}`
`\prettyeW` ⋆ `\prettyeW ⟨tokens⟩ \prettystop`

Alias of the correspondingly-named commands. Only available in LuaTeX.

3 Implementation

Unfortunately, the implementation is not typesetted in TEX. Read the `.sty` file.

Remark: it's possible to do expandable printing in other engines as well by, for example, turning on `\tracingmacros`, parse the token list somehow (and use some not-always-exact logic to distinguish normal character and active character with same meaning; then grep the resulting log file for special markers.

But that would be very, very slow and slows down everything else by turning on logging. Just use LuaTEX for debugging.

There's another option of recompiling the engine and adding some expandable primitive for debug logging...

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.

H

`html-file-name=` . . . 3
`html-refresh-duration=` . . . 4
`html-refresh-strategy=` . . . 3

M

`mode=` . . . 3

P

pretty commands:
`\pretty:N` . . . 5
`\pretty:n` . . . 4, 5
`\pretty:nn` . . . 5
`\pretty:nnn` . . . 5
`\pretty:w` . . . 5
prettye commands:
`\prettye:n` . . . 5, 6
`\prettye:nn` . . . 6
`\prettye:nnn` . . . 6
`\prettye:nnnw` . . . 6
`\prettye:nnw` . . . 6
`\prettye:nw` . . . 6
`\prettye:w` . . . 5, 6
`\prettyeN` . . . 6
`\prettyeW` . . . 6
`\prettyN` . . . 6
`\prettyO` . . . 6
`prettyprint` . . . 6
prettyshow commands:
`\prettyshow:N` . . . 5
`\prettyshowC` . . . 6
`\prettyshowN` . . . 6
`\prettystop` . . . 5, 6
`\prettyV` . . . 6
`\prettyW` . . . 6
`\prettyX` . . . 6

T

`term-prefix-more=` . . . 4
`term-prefix=` . . . 4
`term-shell-decode-cmd-print=` . . . 4
`term-shell-decode-cmd=` . . . 4
`term-wrap-limit=` . . . 4
TEX and LATEX 2ε commands:
`\prettyN` . . . 6
`\prettyO` . . . 6
`\prettyshowC` . . . 6
`\prettyshowN` . . . 6
`\prettyV` . . . 6
`\prettyX` . . . 6