

KKluaverb Package Documentation

Kosei Kawaguchi a.k.a. KKT_εX

Version 2.0.1 (2026/01/16)

目次

1 Outline 3

2 Acknowledgements / Credit 3

3 Usage 3

 3.1 Basic Usage of Lua-enhanced `\verb` 3

 3.2 In Verbatim Environments 3

 3.3 Optional Usage 3

 3.4 Behavior in TOC, Index etc. 4

 3.5 Basic Usage of Lua-enhanced `lstlisting` 4

1 Outline

The `KKluaverb` is a LaTeX package which provides a Lua-enhanced `\verb` command, `\KKverb`. It can be used in the table of contents, index, underline commands (e.g., `\underlineKK` provided by the `luwa-ul` package), `tblr` environment and so on.

2 Acknowledgements / Credit

In developing this package, I made use of an algorithm which is used in “`bxcrawstr`” (by Takayuki YATO)¹⁾.

3 Usage

3.1 Basic Usage of Lua-enhanced `\verb`

As mentioned in the “Outline” section, this package mainly provides enhanced `\verb` command. The usage is not much different from the normal one.

| Input |
|---|
| <code>\KKverb \def\TEST{Test Text.} </code> |

| Output |
|------------------------------------|
| <code>\def\TEST{Test Text.}</code> |

3.2 In Verbatim Environments

When this package is loaded, the Lua-scanning which enables `\KKverb` to detokenize its argument is activated. However, when `\KKverb` used in verbatim environments such as “`lstlistings`” and “`tcblistings`”, the argument cannot be decoded properly. In such cases, you should use `\KKvScanOff` which deactivates the scanning process used internally in `KKverb.lua`. Of course, `\KKvScanOn` re-activates the process.

3.3 Optional Usage

In addition to the main function of `\KKverb`, some optional functions are provided.

1) package: <https://gist.github.com/zr-tex8r/c7901658a866adfdc3cd66b6dfa86997>
article: <https://zrbabbler.hatenablog.com/entry/20181222/1545495849>

`\KKvOpChange` can change font and color of the output of `\KKverb`. Also, it can activate or deactivate the scanner. The usage is as follows:

Input

```

1  {\KKvOpChange{color=blue, font=\gtfamily, enabled=true}%
2  \KKverb|\def\TEST{Test Text.}|}

```

Output

```

\def\TEST{Test Text.}

```

表 1: Default Values of `\KKvOpChange`

| Key | Default Value | Description |
|---------|------------------------|--|
| font | <code>\ttfamily</code> | Sets the font for the verbatim text. |
| color | color | Sets the text color (requires <code>xcolor</code>). |
| enabled | true | Activates or deactivates the scanning engine. |

Additionally, the `enabled` option functions the same as `\KKvScanOn` and `\KKvScanOff`.

3.4 Behavior in TOC, Index etc.

One of the strong points of `\KKverb` is that it can be used in table of contents, index, and footnotes and more. While the traditional `\verb` command always causes an error in such contexts, `\KKverb` does not. This is achieved through the following logic, using the TOC as an example.

When the TeX system generates the TOC, it produces an auxiliary file with the extension `.toc`. In this file, entry data is typically stored in an expanded state.

However, this package requires the raw text data to be passed unexpanded. This is because the text must be intercepted and processed by `process_input_buffer` before TeX's tokenizer converts it into tokens. To resolve this, I implemented a mechanism that automatically prepends the `\unexpanded` command to the starter flag when writing to auxiliary files:

```

1 % In TOC
2 \noexpand\KKlvStart*<encoded tests>\noexpand\KKlvEnd*%

```

Then, the content of the `.toc` file is processed by TeX in the usual way subsequently, because the `\unexpanded` is no longer present.

3.5 Basic Usage of Lua-enhanced `lstlisting`

`\KKcodeS` and `\KKcodeE` provide an environment-like output which is very close to the `lstlisting`. The usage is very simple. The code between `\KKcodeS` and `\KKcodeE` is rendered as

raw texts.

If you want to attach linenumbers to the left of the each line, `\KKcodeS+` will meet the request. Please note that you do not need to type `\KKcodeE+`. `\KKcodeS+` and `\KKcodeE` is the proper pair.

Input

```
1 \KKcodeS+
2 % A sample command
3 \long\def\myprog#1{%
4   \ifx#1\empty
5     \relax
6   \else
7     \message{Processing...}%
8     #1
9   \fi
10 }
11 \KKcodeE
```

Output

```
1 % A sample command
2 \long\def\myprog#1{%
3   \ifx#1\empty
4     \relax
5   \else
6     \message{Processing...}%
7     #1
8   \fi
9 }
```