`KKluaverb` Package Documentation

Kosei Kawaguchi a.k.a. KKTeX

Version 2.1.2 (2026/01/27)

目次

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 devoloping this package, I made use of an algorithm which is used in “bxrawstr” (by Takayuki YATO)[1)].

3 Basic Usage of Lua-enhanced `\verb`

3.1 Fundamental Behavior

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

Input

```
\KKverb|\def\TEST{Test Text.}|
```

Output

```
\def\TEST{Test Text.}
```

In the argument of the command, any linebreaks and blank lines are ignored. However, every space are preserved and rendered exactly as it appears.

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 “tcblisting”, 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.

1) package: https://gist.github.com/zr-tex8r/c7901658a866adfcd3cd66b6dfa86997
article: https://zrbabbler.hatenablog.com/entry/20181222/1545495849

3.3 Optional Usage

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

`\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

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

Output

\def\TEST{Test Text.}

表 1: `\KKvOpChange`

Key	Default Value	Description
`font`	`\ttfamily`	Sets the font for the verbatim text.
`color`	`black`	Sets the text color.
`enabled`	`true`	Activates or deactivates the scanning engine.

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

Also, you can change delimiters by using `\KKvSetDelims` as follows in order to change the delimiters of the command:

Input

```
\KKvSetDelims{[<}{>]}

Changed the delimiters: \KKverb[<\def\foo\s>]
```

Output

Changed the delimiters: \def\foo\s

By default, the delimiters are set as "|". So you can reset them by this:

Input

```
\KKvSetDelims{|}{|}
```

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:

```
% In TOC
\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.

4 Basic Usage of Lua-enhanced `lstlisting`

4.1 Fundamental Behavior

`\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

```
\KKcodeS+
% A sample command
\long\def\myprog#1{%
  \ifx#1\empty
    \relax
  \else
    \message{Processing...}%
    #1
  \fi
}
\KKcodeE
```

Output

```
% A sample command
\long\def\myprog#1{%
  \ifx#1\empty
    \relax
  \else
    \message{Processing...}%
    #1
  \fi
}
```

When you use this Lua-based environment, please keep the following points in mind:

- All indents and spaces are displayed.
- Line breaks and blank lines are rendered exactly as they appear in the source.

4.1.1 `\KKvLNChange`

This command can change the style of the linenumbers.

表 2: `\KKvLNChange`

Key	Default Value	Description
`font`	`\ttfamily`	Sets the font for the linenumbers.
`color`	`black!80`	Sets the color of the linenumbers.
`size`	`\small`	Set the size of the linenumbers.
`start`	`1`	Set the start number.
`style`	`0`	Set the style.

Please note that you do not have to use style key when you use this package in a normal way. This is because style1 and style2 are completely covered by `\KKcodeS/E` or `\KKcodeS+/E` environment. The following example illustrates this point.

Input

```
{\KKvLNChange{style=1}
\KKverb|
% contents
|}

```

```
\KKcodeS
% contents
\KKcodeE
```

The two above produce identical output. To summarize,

`style=0` This is the normal usage of the `\KKverb`.

`style=1` This style corresponds to `\KKcodeS/E` environment.

`style=2` This style corresponds to `\KKcodeS+/E` environment.

4.2 Additional Description

Some corner cases should be payed attention.

4.2.1 Note#1

You cannot use `\KKcodeS` and `\KKcodeE` in the argument of `\KKverb` command. When the scanner is activated, `\KKcodeS` forcibly starts the encoding system, even it is wrapped by `\KKverb`. If you need to display `\KKcodeS` and `\KKcodeE` as they are, deactivate the scanner by `\KKvScanOff` and use `\verb`.

4.2.2 Note#2

Any text on the same line that falls "inside" the `\KKcodeS` and `\KKcodeE` boundaries is ignored, while text "outside" them is preserved. The following example will illustrate the behavior.

Input

```
Not ignored. \KKcodeS+ Ignored.
% Contents
Ignored. \KKcodeE Not ignored.
```

Output

Not ignored.

```
% Contents
```

Not ignored.

5 Color Mapping

5.1 Basic Behavior

This package provides color-mapping option. You can specify the color of the keyword-color, comment-color, comment-token, etc. In order to set a colormap, you should use `luacode*` environment as follows:

Input

```
\begin{luacode*}
  KKLuaVerb.presets["<name-of-style>"] = {
    map = {
      <color1> = { "<token1>", "<token2>",... },
      <color2> = { "<token1>", "<token2>",... },
      ...
    },

    options = {
      word_boundary = true or false,
      comment_char = "<comment-token>",
      comment_color = "<comment-color>",
      escape_char = "<escape-token>",
      delimiters = {
        { start = '<starter1>', stop = '<ender1>', color = "<patial-color1>" },
        { start = '<starter2>', stop = '<ender2>', color = "<patial-color2>" },
      },
      forced_tokens = {
        ["<forced-token1>"] = "<color2>",
        ["<forced-token2>"] = "<color2>",
      },
      word_components = "[<word-components>]"
    }
  }
\end{luacode*}
```

You can set multiple presets, and can change or activate by `\KKvUsePreset{<name-of-style>}` command.

5.2 The `word_boundary` Option

The option controls how the scanner identifies the limits of a keyword during the tokenization process. It determines whether a sequence of characters should be treated as a standalone keyword or as part of a larger string.

When Disabled: `false` The scanner performs a raw string search. Any occurrence of the target token is highlighted, regardless of its surrounding characters. For example, if the keyword is `if`, it will be highlighted even inside the word `different`.

When Enabled (Default): `true` This mode prevents unintended partial matching. It ensures that a keyword is highlighted only when it stands as a complete word. Crucially, symbols (such as `(`, `[`, `#`, etc.) are automatically treated as boundaries. For example, `\def#1` will result in `\def` being highlighted, whereas the `if` inside `different` will remain plain text.

5.3 Example

Input Setting

```
\begin{luacode*}
  KKLuaVerb.presets["TeXStyle"] = {
    map = {
      DarkCyan = { "{", "}" },
      OrangeRed = { "[", "]" },
      DeepPink = { "(", ")" },
      DarkGoldenrod = { "&", "$", "^", "_", "\\"},

      CornflowerBlue = {
        "\\documentclass", "\\usepackage", "\\begin", "\\end",
        "\\section", "\\subsection", "\\chapter",
        "\\makeatletter", "\\makeatother", "\\ExplSyntaxOn", "\\ExplSyntaxOff"
      },

      MediumPurple = {
        "\\def", "\\edef", "\\gdef", "\\xdef",
        "\\newcommand", "\\renewcommand", "\\providecommand",
        "\\let", "\\setcopy", "\\long", "\\global"
      },

      Magenta = {
        "\\if", "\\else", "\\fi", "\\ifx", "\\ifdefined",
```

```
        "\\ifnum", "\\ifdim", "\\loop", "\\repeat", "\\noexpand"
      },

      IndianRed = {
        "\\KKlvStart*", "\\KKlvEnd*"
      }
    },

    options = {
      word_boundary = true,
      comment_char = "%",
      comment_color = "ForestGreen",
      escape_char = "\\",
      delimiters = {
        { start = "$", stop = "$", color = "SpringGreen3" },
        { start = "\\[", stop = "\\]", color = "SpringGreen3" },
      },
      forced_tokens = {
        ["{"] = "DarkCyan",
        ["}"] = "DarkCyan",
      },
    }
  }
\end{luacode*}
```

Input | Actual Use

```
\KKvUsePreset{TeXStyle}

\KKcodeS+
\usepackage{xcolor}
\usepackage{KKluaverb}

\begin{document}
  Hello, KKTeX!
\end{document}
\KKcodeE

```

```
\KKcodeS
{ [ \ ( \% \$ # & _ ^ ) / ] }
Inline mathmode: $a + b = \frac{x}{y}$
Display-style mathmode:
\[a + b = \frac{x}{y}\]
\KKcodeE
```

Output

```
\usepackage{xcolor}
\usepackage{KKluaverb}

\begin{document}
  Hello, KKTeX!
\end{document}
{ [ \ ( \% \$ # & _ ^ ) / ] }
Inline mathmode: $a + b = \frac{x}{y}$
Display-style mathmode:
\[a + b = \frac{x}{y}\]
```

6 Text Mapping

By using `\KKvSetMap`, you can replace a certain text in the argument of `\KKverb` and `\KKcodeS/E` environment.

Input | Actual Use

```
\KKvSetMap ⟨{!}{⟩ EXCLAMATION}
\KKvSetMap ⟨{?}{⟩ QUESTION}

\KKverb|Wait, you really change them!?|

\KKvSetMap{!}{!}
\KKvSetMap{?}{?}
\KKverb|Oh, how can I reset them!?|
```

Output

```
Wait, you really change them 〈EXCLAMATION〉 〈QUESTION〉
Oh, how can I reset them!?
```

Using this, you can visualize all spaces in the argument of `\KKverb` and `\KKcodeS/E` environment. However, you have to use `\KKvSpaceForce` if you want to reset. By default, this package automatically replaces any spaces to Unicode character U+00A0 (Non-breaking Space). Therefore, you need to use the special command if you want to quit visualizing spaces.

Input Actual Use

```
\KKvSetMap{ }{<sp>}
\KKverb| Look! All spaces are visualized!! |

\KKvSpaceForce
\KKverb| Oh, spaces are no longer visualized. |
```

Output

```
<sp><sp>Look!<sp>All<sp>spaces<sp>are<sp>visualized!!<sp><sp>
  Oh, spaces are no longer visualized.
```