Package **`highlightlatex`** manual

Vincent Kuhlmann
`vincent.kuhlmann@hotmail.com`

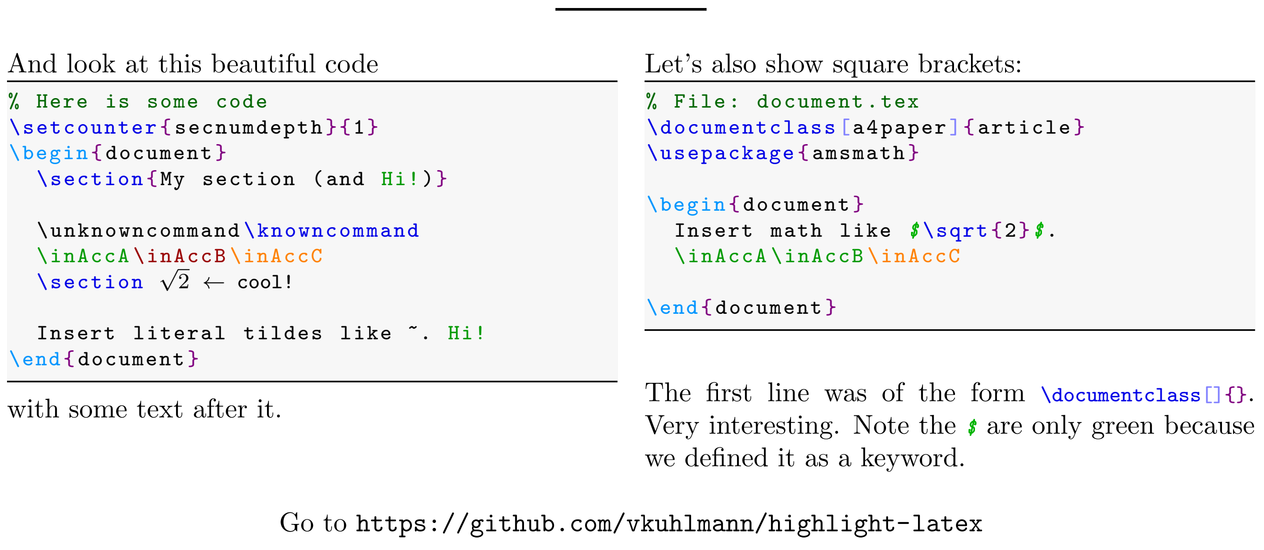
March 29, 2021

Abstract

The `highlightlatex` package provides colored syntax highlighting for LaTeX source code, without aid from outside LaTeX. The aim is to make this accessible enough so people stop falling back to plain verbatim for this. For this, it builds further on the generic ‘listings’ package. An example output is shown in Figure 1 .

Repository

`https://github.com/vkuhlmann/highlight-latex`



And look at this beautiful code

```
% Here is some code
\setcounter{secnumdepth}{1}
\begin{document}
  \section{My section (and Hi!)}

  \unknowncommand\knowncommand
  \inAccA\inAccB\inAccC
  \section √2 ← cool!

  Insert literal tildes like ~. Hi!
\end{document}
```

with some text after it.

Let’s also show square brackets:

```
% File: document.tex
\documentclass[a4paper]{article}
\usepackage{amsmath}

\begin{document}
  Insert math like $\sqrt{2}$.
  \inAccA\inAccB\inAccC

\end{document}
```

The first line was of the form `\documentclass[]{}`. Very interesting. Note the `$` are only green because we defined it as a keyword.

Go to `https://github.com/vkuhlmann/highlight-latex`

Figure 1: Output of ‘demo.tex’.

Contents

1 Getting started

Include the package using `\usepackage{highlightlatex}`. If not recognized, refresh your installation's package repository and set it to install `highlightlatex`. There are two ways in which code can be shown.

1.1 Inline style

EXAMPLE

```
Your file begins with a line of the form \hll|\documentclass[]{}|. The
square brackets ...
```

RESULT

Your file begins with a line of the form `\documentclass[]{}`. The square brackets ...

The first non-space character following `\hll` delimits the argument to this command.

1.2 Block style

EXAMPLE

```
Your basic document now looks like
\begin{highlightblock}[gobble=4]
    \documentclass[a4paper]{article}

    \begin{document}
        Hello world!
    \end{document}
\end{highlightblock}
```

RESULT

Your basic document now looks like

```
\documentclass[a4paper]{article}

\begin{document}
    Hello world!
\end{document}
```

gobble. The `gobble` parameter specifies how many characters need to be stripped of the start of each line. In the example, we had an indentation of four spaces for the literal code. Without the gobble, those spaces would be shown. See Subsection 3.3 for setting a global default, and how tabs are interpreted.

linewidth. The gray box extends to the width of the page. To set it to only use 60%, use `linewidth=0.6\linewidth`, or for a width of 20 characters, set `linewidth=10em`.

There are more keys you can provide. Check the `listings` package documentation [1] for options available to the `lstlisting`-environment and `lstset` command.

beamer. Troubles using beamer? Or troubles with it working at some places, and failing at others? Check Section 4.

[1] `https://www.ctan.org/pkg/listings`

2 Macro `\updatehighlight`

2.1 Examples

2.1.1 Adding a command to a highlighting rule

By default, only some LaTeX commands will be highlighted in blue. If there are others you need, like `\tableofcontents` and `\figref`, update the highlighting rules:

EXAMPLE
```
\updatehighlight{
    name = default,
    add = {
        \tableofcontents, \figref
    }
}
```

The change will only affect code after it. I recommend issuing `updatehighlight` in your preamble (before the `\begin{document}`). In some situations you might want to change things mid-document. That's possible too.

2.1.2 Coloring a command differently

As shown in `demo.tex`, you can put any command or keyword you want to highlight in a different color. You do this with

EXAMPLE
```
\updatehighlight{
    % The name allows you to modify the style later.
    name = spotlight,
    color = orange,
    add = {
        \tableofcontents
    }
}
```

You can use the `xcolor` syntax for describing colors as well. If you find the orange too bright, you can replace it with `orange!90!black`: 90% orange, remaining is black. For more information on color definitions and name, refer to LaTeX/Colors on Wikibooks [2].

2.2 Specification

The argument to `\updatehighlight` is a key-value list. Keys are processed in order and can appear multiple times. If a value contains special characters, like a '=' or ',', enclose the value with braces. If you want to create some space in your code, don't use a blank line, they are interpreted by LaTeX as a paragraph end. You can use an empty line with only a percent sign (`%`) instead. The processing is done using the package xkeyval.

You can merge any two `\updatehighlight` in one. No need to close and reopen `\updatehighlight` for each highlighting rule.

name = ⟨`characters`⟩,

Sets the named rule to be modified. If a rule with the name does not exist, a new rule will be created.

There are two default named rules. The first one is `default`, which includes a bunch of basic commands, and has by default a dark blue color. The second one is `structure`, which consists of `\begin` and `\end` and prints them in light blue.

classoffset = ⟨`integer`⟩,

Sets the `listings` classoffset manually. Try to avoid this. Use **name** to refer to existing rules instead.

[2] `https://en.wikibooks.org/wiki/LaTeX/Colors`

add `= ⟨list⟩,`

Adds a command (`\mycommand`) or keyword (`Hi!`) to the current rule. The value can contain multiple commands and keywords by seperating them with comma's. The value needs to be surrounded with braces.

remove `= ⟨list⟩,`

Removes a commands or keywords from the current rule. The value can contain multiple commands and keywords by seperating them with comma's. The value needs to be surrounded with braces.

clear `,`

Removes all commands and keywords from the current rule.

EXAMPLE

```
\updatehighlight{
    name = default,
    clear
}
```

color `= ⟨color⟩,`

Specifies the current rule's style to color text in the given color. This can also be done using the **style** option, for example `color=red` is equivalent to `style=\color{red}`.

style `= ⟨LaTeX code⟩,`

Specifies the current rule's style. A rule can have only one style. If you specify a style after `add` or `remove`, this starts a new (unnamed) rule. In practice, the only style which will probably work for you is just a color. For that, using the 'color' key is just a bit easier and neater. But hey, you have the option to set whatever style you want. :)

3 Global settings

Global parameters can either be set as package option, or through invocation of the `\hllconfigure`-command.

DEFAULTS

```
\hllconfigure{
    frame=lines,
    tabsize=4,
    gobble=0,
    backgroundcolor=gray!6!white,
    bracecolor=red!50!blue,
    bracketcolor=blue!50!white,
    commentcolor=green!40!black,
    alsoletter={$@_!|?$},
}
```

The argument to `\hllconfigure` and the value of the package options is a key-value list. Keys are processed in order and can appear multiple times. If a value contains special characters, like a '=' or ',', enclose the value with braces. If you want to create some space in your code, don't use a blank line, they are interpreted by LaTeX as a paragraph end. You can use an empty line with only a percent sign (`%`) instead. The processing is done using the package xkeyval.

The options available are specified in the following subsections.

3.1 Block appearance

frame = ⟨choice⟩, [= lines]

Specifies the frame you want around code blocks. My favorites are `lines` and `none`. Check the listings package documentation [3] for all possibilities.

The square brackets in the syntax definition above mean you can specify '`frame,`' and that this is equivalent to '`frame=lines,`'.

noframe,

Equivalent to `frame=none`.

debugframe,

Shows a colored frame around boxes loaded using the `saveblock`+`\useblock` scheme. This can be useful for debugging unexpected offsets. Not for production use.

backgroundcolor = ⟨color⟩,

Sets the background color of code blocks.

3.2 General appearance

mathdollar = ⟨key-value list⟩, [= on]

Configures styling of code between dollar signs.

The command takes a key-value list. The allowed keys are

on,

Enables mathdollar styling.

[3] `https://www.ctan.org/pkg/listings`

off,

Disables mathdollar styling.

style = ⟨LaTeX code⟩,

Sets style.

color = ⟨color⟩,

Sets style to color text with this color. Hence `color=XXX` is equivalent to `style=\color{XXX}`.

cumulative = ⟨boolean⟩, [= true]

Applies normal coloring rules on top of the style provided.

By default, **mathdollar** is off and has default color `green!40!black` and cumulative enabled. Each invocation of the **mathdollar** key is implicitly preceded with the option `on`.

bracecolor = ⟨color⟩,

Sets the color for displaying curly braces (`{` and `}`).

bracketcolor = ⟨color⟩,

Sets the color for displaying square brackets (`[` and `]`).

commentcolor = ⟨color⟩,

Sets the color for displaying comments, that is a percent sign (`%`) and the text following it.

3.3 Other

styleanywhere,

Overrides the default behavior that `style` starts a new highlighting rule after commands like `add` and `remove`.

tabsize = ⟨integer⟩,

Sets the amount of space characters a tab character is converted to, when no non-tab character has preceded on the line. Behavior of tab characters is left undefined by this package if a non-tab character has preceded on the line.

gobble = ⟨integer⟩,

Sets the default amount of characters that will be stripped of code lines within a code block. This is useful to combat indentation used for structuring your `.tex` file to show up in the code block.

gobbletabs = ⟨integer⟩,

Sets the gobble in amount of **tabsize**'s. The amount of **tabsize**'s is evaluated at the start of each code block. The result of `tabsize=4,gobbletabs=2` is equivalent to `gobble=8`. However, when you indent with tabs, this option allows you to quickly change the **tabsize** without having to update **gobble** everywhere.

alsoletter = `⟨characters⟩,`

Specifies the character list of special characters to be treated like ‘letter’. This makes them usable in highlighting rules. However, beware using special characters can conflict with other typesetting routines. Setting this option overrides the previous value.

4 Fragility

4.1 Introduction

When passing command arguments around, or storing environment content, LaTeX interprets all characters. This includes seeing `\maketitle` in `\hll|\maketitle|` as a real command. To prevent this behavior, everything from `\verb`, to the `verbatim`-environment, to the `listings` package (which this package depends on) temporarily changes the interpretation of characters that are still to be read. The blackslash before maketitle in `\hll|\maketitle|` will be read as 'just text' (a *letter* technically).

This reliance on interpretation being done at the last moment is called 'fragility'. When content gets interpreted early, like the `frame`-environment in `beamer` does, this fragility is broken. A `frame`-environment allows a `fragile` option to be set, which probably solves this, but I would avoid this. In situations where fragility is broken, you need to either *escape* the displayed code, or *pre-processing* the displayed code outside a fragile breaking situation.

4.2 Escaping

Escaping is done by preceding certain special characters with a backslash. For example, `\hll|\documentclass[]{}|` becomes `\hll|\\documentclass[]\{\}|`.

4.3 Pre-processing

For large code blocks, escaping is undesirable. Therefore we pre-process the code when we have fragility preserved, and load the result in our fragile breaking situation. The storing is done through the `saveblock`-environment. This environment takes a name under which the result will be stored. A name can be re-used and will silently overwrite any existing save.

EXAMPLE

```
\begin{saveblock}{basicfigure}
    \begin{highlightblock}[linewidth=0.6\textwidth]
        \begin{figure}
            \includegraphics
            [width=0.9\linewidth]
            {myPlot.pdf}

            \caption{My plot}
            \label{fig:myplot}
        \end{figure}
    \end{highlightblock}
\end{saveblock}
```

To load the result, use the `\useblock` or `\consumeblock` command with the save name provided as argument. For the example, these are `\useblock{basicfigure}` and `\consumeblock{basicfigure}` respectively.

The `\consumeblock` variant deletes the result from memory after use. This is useful if you have many different save names occupied. Not deleting them keeps them loaded till the PDF has fully generated. Note `\consumeblock` can result in unexpected behavior, for example animations in a beamer frame might need the code line to be executed multiple times. Use `\useblock` when you can't guarantee the last use of a block.

There is a separate demo for fragile breaking situations. You can find it as `deamerdemo.tex`.

5 Adding extra space

By default, `highlightlatex` follows an approach where it minimizes spacing. This gives you full control over how tight or spacious your document looks. Just use commands like `\medskip` to add extra spacing. The package doesn't currently include an option to do it everywhere automatically.

6 License & Credits

The package is available under MIT License. See LICENSE.txt.

Thanks for minor fixes:

gemmaro

For any bug, feature request, unclarity, or whatever else related to this package, you're welcome to open an issue or pull-request. Issues can be opened on

URL
```
https://github.com/vkuhlmann/highlight-latex/issues
```

Thanks for contributing!