The minted package: Highlighted source code in LaTeX

Konrad Rudolph
konrad_rudolph@madrat.net

January 8, 2010

Contents

1 Introduction

minted is a package that allows formatting source code in LaTeX. For example:

```
\begin{minted}{language}
  code
\end{minted}
```

will highlight a piece of code in a chosen language. The display can be customized by a number of arguments and colour schemes.

Unlike some other packages, most notably listings, minted requires the installation of an additional software, Pygments. This may seem like a disadvantage but there are advantages, as well:

Pygments provides far superior syntax highlighting compared to conventional packages. For example, listings basically only highlights strings, comments and

keywords. Pygments, on the other hand, can be completely customized to highlight any token kind the source language might support. This might include special formatting sequences inside strings, numbers, different kinds of identifiers and exotic constructs such as HTML tags.

Some languages make this especially desirable. Consider the following Ruby code as an extreme, but at the same time typical, example:

```ruby
class Foo
    def init
        pi = Math::PI
        @var = "Pi is approx. #{pi}"
    end
end
```

Here we have three different colours for identifiers (four, if you count keywords) and escapes from inside strings, none of which pose a problem to Pygments.

Additionally, installing Pygments is actually incredibly easy (see the next section).

2 Installation

Pygments is written in Python so make sure that at least Python 2.6 is installed on you system:

```
$ python --version
Python 2.6.2
```

If that's not the case, you can download it from the website or use your operating system's package manager.

You can then install Pygments using the following simple command:

```
$ sudo easy_install Pygments
```

(If you've already got Pygments installed, be advised that minted requires at least version 1.2.)

3 Basic usage

3.1 Preliminary

Since minted makes calls to the outside world (i.e. Pygments), you need to tell the LaTeX processor about this by passing it the `-shell-escape` option or it won't allow such calls. In effect, instead of calling the processor like this:

```
$ latex input
```

you need to call it like this:

```
$ latex -shell-escape input
```

The same holds for other processors, such as `pdflatex` or `xelatex`.

3.2 Formatting source code

minted Using minted is straightforward. For example, to highlight a Python source code, we might use the following code snippet (result on the right):

```
\begin{minted}{python}
def boring(args = None):
    pass
\end{minted}
```

```
def boring(args = None):
    pass
```

Optionally, the environment accepts a number of options in `key=value` notation, which are described in more detail below.

\mint For one-line source codes, you can alternatively use a shorthand notation similar to `\verb`:

```
\mint{python}|import this|
```

```
import this
```

The complete syntax is `\mint[`⟨*options*⟩`]{`⟨*language*⟩`}/code/` Where the code delimither `/`, like with `\verb`, can be almost any punctuation character. Again, this command supports a number of options described below.

\inputminted Finally, there's the comment `\inputminted` command to read and format whole files. Its syntax is `\inputminted[`⟨*options*⟩`]{`⟨*language*⟩`}{`⟨*filename*⟩`}`.

3.3 Using different styles

Instead of using the default style you may choose an another stylesheet provided \usemintedstyle by Pygments by its name. For example, this document uses the "`trac`" style. To do this, put the following into the prelude of your document:

\usemintedstyle{name}

To get a list of all available stylesheets, execute the following command on the command line:

```
$ pygmentize -L styles
```

Creating own styles is also very easy. Just follow the instructions provided on the website.

3.4 Supported languages

Pygments at the moment supports over 150 different programming languages, template languages and other markup languages. To see an exhaustive list of the currently supported languages, use the command

```
$ pygmentize -L lexers
```

4 Options

4.1 Usage

All minted highlight commands accept the same set of options. Options are specified as a comma-separated list of `key=value` pairs. For example, we can specify that the lines should be numbered:

```
\begin{minted}[linenos=true]{c++}
#include <iostream>
int main() {
    std::cout << "Hello "
              << "world"
              << std::endl;
}
\end{minted}
```

```
1 #include <iostream>
2 int main() {
3     std::cout << "Hello "
4               << "world"
5               << std::endl;
6 }
```

An option value of `true` may also be omitted entirely (including the =). To customize the display of the line numbers further, override the `\theFancyVerbLine` command. Consult the fancyvrb documentation for details.

Here's another example: we want to use the LaTeX math mode inside comments:

```
\begin{minted}[mathescape]{python}
# Returns $\sum_{i=1}^{n}i$
def sum_from_one_to(n):
    r = range(1, n + 1)
    return sum(r)
\end{minted}
```

```
# Returns ∑_{i=1}^{n} i
def sum_from_one_to(n):
    r = range(1, n + 1)
    return sum(r)
```

To make your LaTeX code more readable you might want to indent the code inside a `minted` environment. The option `gobble` removes these unnecessary whitespace characters from the output:

```
\begin{minted}[gobble=2,showspaces]{python}
  def boring(args = None):
      pass
\end{minted}

versus

\begin{minted}[showspaces]{python}
  def boring(args = None):
      pass
\end{minted}
```

```
def␣boring(args␣=␣None):
␣␣␣␣pass

versus

␣␣def␣boring(args␣=␣None):
␣␣␣␣␣␣pass
```

4.2 Available options

Following is a full list of available options. For more detailed option descriptions please refer to the fancyvrb documentation, except where noted otherwise.

baselinestretch (`auto`|dimension) : Value to use as for baselinestretch inside the listing (default: `auto`).

firstline (integer) : First line to be shown (default: `1`). All lines before that line are ignored and do not appear in the output.

firstnumber (`auto`|integer) : Line number of the first line (default: `auto` = 1).

frame (`none`|`leftline`|`topline`|`bottomline`|`lines`|`single`) : The type of frame to put around the source code listing (default: `none`).

framerule (dimension) : Width of the frame (default: `0.4pt`).

framesep (dimension) : Distance between frame and content (default: `\fboxsep`).

gobble (integer) : Remove the first n characters from each input line (default: `0`).

label (`[string]string`) : Assigns the strings as labels to the listing (default: *none*).

lastline (integer) : Last line to be shown (default: *last line of input*).

linenos (boolean) : Enables line numbers (default `false`).

mathescape (boolean) : Enable LaTeX math mode inside comments (default: `false`). Usage as in package listings.

numberblanklines (boolean) : Enables or disables numbering of blank lines (default: `true`).

numbersep (dimension) : Gap between numbers and start of line (default: `12pt`).

resetmargins (boolean) : Resets the left margin inside other environments (default: `false`).

rulecolor (color command) : The color of the frame (default: *black*)

samepage (boolean) : Forces the whole listing to appear on the same page, even if it doesn't fit (default: `false`).

showspaces (boolean) : Enables visible spaces: `visible␣spaces` (default: `false`).

stepnumber (integer) : Interval at which line numbers appear (default: `1`).

texcl (boolean) : Enables LaTeX code inside comments (default: `false`). Usage as in package listings.

xleftmargin (dimension) : Indentation to add before the listing (default: `0`).

xrightmargin (dimension) : Indentation to add after the listing (default: `0`).

5 To do list

- Add check for pygmentize installation and version.
- Allow multiple stylesheets in one file.
- Allow quotes in fancyvrb arguments.

6 Implementation

6.1 Option processing

`\minted@resetoptions` Reset options.

```
1 \newcommand\minted@resetoptions{}
```

`\minted@defopt` Define an option internally and register it with in the `\minted@resetoptions` command.

```
\newcommand\minted@defopt[1]{
  \expandafter\def\expandafter\minted@resetoptions\expandafter{%
    \minted@resetoptions
    \@namedef{minted@opt@#1}{}}}
```

`\minted@opt` Actually use (i.e. read) an option value. Options are passed to `\detokenize` so that `\immediate\write18` will work properly.

```
\newcommand\minted@opt[1]{
  \expandafter\detokenize%
    \expandafter\expandafter\expandafter{\csname minted@opt@#1\endcsname}}
```

`\minted@define@opt` Define a generic option.

```
\newcommand\minted@define@opt[2]{
  \minted@defopt{#1}
  \define@key{minted@opt}{#1}{\@namedef{minted@opt@#1}{#2}}}
```

`\minted@define@switch` Define an option switch (values are either `true` or `false`, and `true` may be omitted, e.g. `foobar` is the same as `foobar=true`).

```
\newcommand\minted@define@switch[2]{
  \minted@defopt{#1}
  \define@booleankey{minted@opt}{#1}{
    \@namedef{minted@opt@#1}{#2}}
   {\@namedef{minted@opt@#1}{}}}
```

`\minted@define@extra` Extra options are passed on to fancyvrb.

```
\minted@defopt{extra}
\newcommand\minted@define@extra[1]{
  \define@key{minted@opt}{#1}{
    \expandafter\def\expandafter\minted@opt@extra\expandafter{%
      \minted@opt@extra,#1=##1}}}
```

`inted@define@extra@switch`

```
\newcommand\minted@define@extra@switch[1]{
  \define@booleankey{minted@opt}{#1}{
    \expandafter\def\expandafter\minted@opt@extra\expandafter{%
      \minted@opt@extra,#1}}
     {\expandafter\def\expandafter\minted@opt@extra\expandafter{%
      \minted@opt@extra,#1=false}}}
```

Actual option definitions.

```
\minted@define@switch{texcl}{-P texcomments}
\minted@define@switch{mathescape}{-P mathescape}
\minted@define@switch{linenos}{-P linenos}
\minted@define@opt{gobble}{-F gobble:n=#1}
\minted@define@extra{frame}
\minted@define@extra{framesep}
\minted@define@extra{framerule}
\minted@define@extra{rulecolor}
\minted@define@extra{numbersep}
\minted@define@extra{stepnumber}
```

```
\minted@define@extra{firstline}
\minted@define@extra{lastline}
\minted@define@extra{baselinestretch}
\minted@define@extra{label}
\minted@define@extra{xleftmargin}
\minted@define@extra{xrightmargin}
\minted@define@extra@switch{numberblanklines}
\minted@define@extra@switch{showspaces}
\minted@define@extra@switch{resetmargins}
\minted@define@extra@switch{samepage}
```

6.2 Internal helpers

`\minted@savecode` Save a code to be pygmentized to a file.

```
\newwrite\minted@code
\newcommand\minted@savecode[1]{
  \immediate\openout\minted@code\jobname.pyg
  \immediate\write\minted@code{#1}
  \immediate\closeout\minted@code}
```

`\minted@pygmentize` Pygmentize the file given as first argument (default: `\jobname.pyg`) using the options provided.

```
\newcommand\minted@pygmentize[2][\jobname.pyg]{
  \def\minted@cmd{pygmentize -l #2 -f latex -F tokenmerge \minted@opt{gobble}
    \minted@opt{texcl} \minted@opt{mathescape} \minted@opt{linenos}
    -P "verboptions=\minted@opt{extra}" -o \jobname.out.pyg #1}
  %\immediate\typeout{\minted@cmd} % For debugging.
  \immediate\write18{\minted@cmd}
  \input{\jobname.out.pyg}
  \immediate\write18{rm \jobname.out.pyg}}
```

`\minted@usedefaultstyle` Include the default stylesheet.

```
\newcommand\minted@usedefaultstyle{\usemintedstyle{default}}
```

6.3 Public API

`\usemintedstyle` Include stylesheet.

```
\newcommand\usemintedstyle[1]{
  \renewcommand\minted@usedefaultstyle{}
  \immediate\write18{pygmentize -S #1 -f latex > \jobname.pyg}
  \input{\jobname.pyg}}
```

`\mint` Highlight a small piece of verbatim code. Usage:

```
\mint[options]{language}/code/
```

where / is an arbitrary delimiter, much like for `\verb` and fancyvrb's `\Verb`.

```
\newcommand\mint[3][]{
  \DefineShortVerb{#3}
  \minted@resetoptions
  \setkeys{minted@opt}{#1}
  \SaveVerb[aftersave={
```

```
    \UndefineShortVerb{#3}
    \minted@savecode{\FV@SV@minted@verb}
    \minted@pygmentize{#2}
    \immediate\write18{rm \jobname.pyg}}]{minted@verb}#3}
```

`\minted` Highlight a longer piece of code inside a verbatim environment. Usage:

```
\begin{minted}[options]{language}
  code
\end{minted}
```

```
\newcommand\minted@proglang[1]{}
\newenvironment{minted}[2][]
  {\VerbatimEnvironment
   \renewcommand{\minted@proglang}[1]{#2}
   \minted@resetoptions
   \setkeys{minted@opt}{#1}
   \begin{VerbatimOut}{\jobname.pyg}}%
  {\end{VerbatimOut}
   \minted@pygmentize{\minted@proglang{}}
   \immediate\write18{rm \jobname.pyg}}
```

`\inputminted` Highlight an external source file. Usage:

```
\inputminted[options]{language}{path}
```

```
\newcommand\inputminted[3][]{
  \minted@resetoptions
  \setkeys{minted@opt}{#1}
  \minted@pygmentize[#3]{#2}}
```

6.4 Epilogue

Load default stylesheet – but only if user has not yet loaded a custom stylesheet in the preamble.

```
\AtBeginDocument{
  \minted@usedefaultstyle}
```

Check whether LaTeX was invoked with `-shell-escape` option.

```
\AtEndOfPackage{
  \ifeof18\PackageError{minted}{You must invoke LaTeX with the -shell-escape
    flag}
  {Pass the -shell-escape flag to LaTeX. Refer to the minted.sty documentation
  for more information.}\fi}
```