The minted package:
Highlighted source code in LaTeX

Geoffrey M. Poore
gpoore@gmail.com
github.com/gpoore/minted

翻译:virhuiai
virhuiai@qq.com

Originally created and maintained (2009–2013) by
Konrad Rudolph

v2.6 from 2021/12/24

摘要

minted is a package that facilitates expressive syntax highlighting using the powerful Pygments library. The package also provides options to customize the highlighted source code output.

minted 是一个使用强大的 Pygments 库来实现语法高亮的宏包。该宏包还提供了自定义高亮源代码输出的选项。

License

许可证

目录

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 appearance can be customized with a number of options and color schemes.

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

Pygments provides 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 kind of token 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:

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

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

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

1 介绍

minted 是一个允许在 LaTeX 中格式化源代码的包。例如：

```
begin{minted}{<language>}
 <code>
end{minted}
```

将会以选定的语言高亮显示一段代码。外观可以通过许多选项和配色方案进行定制。

与其他一些包不同，尤其是 listings，minted 需要安装额外的软件 Pygments。这可能看起来像是一个缺点，但也有重要的优点。

Pygments 相比传统的包提供了更好的语法高亮。例如，listings 基本上只高亮字符串、注释和关键字。而 Pygments 可以完全自定义以突出显示源语言可能支持的任何类型的标记。这可能包括字符串内的特殊格式序列、数字、不同类型的标识符以及 HTML 标签等特殊结构。

一些语言特别需要这样的功能。以以下 Ruby 代码为极端但典型的例子：

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

在这个例子中，我们有四种不同的标识符颜色（如果计算关键字，则有五种），以及字符串内的转义字符，这对于 Pygments 来说都不是问题。

此外，安装 Pygments 实际上非常简单（详见下一节）。

2 Installation

2.1 Prerequisites

Pygments is written in Python, so make sure that you have Python 2.6 or later installed on your system. This may be easily checked from the command line:

```
$ python --version
Python 2.7.5
```

If you don't have Python installed, you can download it from the Python website or use your operating system's package manager.

Some Python distributions include Pygments (see some of the options under "Alternative Implementations" on the Python site). Otherwise, you will need to install Pygments manually. This may be done by installing setuptools, which facilitates the distribution of Python applications. You can then install Pygments using the following command:

```
$ sudo easy_install Pygments
```

Under Windows, you will not need the `sudo`, but may need to run the command prompt as administrator. Pygments may also be installed with `pip`:

```
$ pip install Pygments
```

If you already have Pygments installed, be aware that the latest version is recommended (at least 1.4 or later). Some features, such as `escapeinside`, will only work with 2.0+. minted may work with versions as early as 1.2, but there are no guarantees.

2.2 Required packages

minted requires that the following packages be available and reasonably up to date on your system. All of these ship with recent TEX distributions.

2 安装

2.1 先决条件

Pygments 是用 Python 编写的，所以请确保您的系统上已安装 Python 2.6 或更高版本。您可以通过命令行轻松检查：

```
$ python --version
Python 2.7.5
```

如果您尚未安装 Python，可以从Python 网站下载，或使用您操作系统的软件包管理器。

某些 Python 发行版已经包含了 Pygments（请参考 Python 网站上的 "替代实现" 选项）。否则，您需要手动安装 Pygments。可以通过安装setuptools来完成，它可以方便地分发 Python 应用程序。然后，您可以使用以下命令安装 Pygments：

```
$ sudo easy_install Pygments
```

在 Windows 下，您不需要使用 `sudo`，但可能需要以管理员身份运行命令提示符。也可以使用 `pip`[1] 安装 Pygments：

```
$ pip install Pygments
```

如果您已经安装了 Pygments，请注意建议使用最新版本（至少为 1.4 或更高版本）。某些功能，如 `escapeinside`，只能在 2.0+ 版本中工作。minted 可能与早期版本（至少为 1.2）一起工作，但不能保证。

2.2 所需软件包

minted 需要以下软件包在您的系统上可用且相对较新。这些软件包都包含在最新的 TEX 发行版中。

[1] 译注：最新的使用 pip3 安装，2 版本的安装不了了！

- keyval
- kvoptions
- fancyvrb
- fvextra
- upquote
- float
- ifthen
- calc
- ifplatform
- pdftexcmds
- etoolbox
- xstring
- xcolor
- lineno
- framed
- shellesc[1]
- catchfile

[1]for luatex 0.87+

2.3 Installing minted

You can probably install minted with your TEX distribution's package manager. Otherwise, or if you want the absolute latest version, you can install it manually by following the directions below.

You may download `minted.sty` from the project's homepage. We have to install the file so that TEX is able to find it. In order to do that, please refer to the TEX FAQ. If you just want to experiment with the latest version, you could locate your current `minted.sty` in your TEX installation and replace it with the latest version. Or you could just put the latest `minted.sty` in the same directory as the file you wish to use it with.

3 Basic usage

3.1 Preliminary

Since minted makes calls to the outside world (that is, 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`.

2.3 安装 minted

您可以使用 TEX 发行版的软件包管理器安装 minted。否则，如果您想要安装最新版本，可以按照以下说明手动安装。

您可以从项目主页下载 `minted.sty`。我们需要将该文件安装到使 TEX 能够找到它的位置。为此，请参考TEX FAQ。如果您只是想尝试最新版本，可以找到您当前 TEX 安装中的 `minted.sty`，并将其替换为最新版本。或者，您可以将最新的 `minted.sty` 放在与要使用它的文件相同的目录中。

3 基本使用

3.1 准备工作

由于 minted 会调用外部程序（即 Pygments），您需要通过传递`-shell-escape`选项告诉 LATEX 处理器，否则它不会允许这样的调用。实际上，您需要像这样调用处理器：

```
$ latex input
```

您需要像这样调用它：

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

其他处理器，如`pdflatex`或`xelatex`，也是一样的。

You should be aware that using `-shell-escape` allows LaTeX to run potentially arbitrary commands on your system. It is probably best to use `-shell-escape` only when you need it, and to use it only with documents from trusted sources.

您应该知道，使用`-shell-escape`允许 LaTeX 在您的系统上运行潜在的任意命令。最好只在需要时使用`-shell-escape`，并且只与来自可信源的文档一起使用。

Working with OS X

If you are using minted with some versions/configurations of OS X, and are using caching with a large number of code blocks (> 256), you may receive an error like

```
OSError: [Errno 24] Too many open files:
```

This is due to the way files are handled by the operating system, combined with the way that caching works. To resolve this, you may use the OS X commands `launchctl limit maxfiles` or `ulimit -n` to increase the number of files that may be used.

在 OS X 上使用

如果您在某些版本/配置的 OS X 上使用 minted，并且使用大量代码块（> 256）进行缓存，您可能会遇到以下错误：

```
OSError: [Errno 24] Too many open files:
```

这是由操作系统处理文件的方式以及缓存工作方式造成的。要解决此问题，您可以使用 OS X 命令`launchctl limit maxfiles`或`ulimit -n`来增加可使用的文件数量。

3.2 A minimal complete example

The following file `minimal.tex` shows the basic usage of minted.

3.2 一个最小的完整示例

以下文件`minimal.tex`显示了 minted 的基本使用方法。

```latex
\documentclass{article}

\usepackage{minted}

\begin{document}
\begin{minted}{c}
int main() {
    printf("hello, world");
    return 0;
}
\end{minted}
\end{document}
```


```latex
\documentclass{article}

\usepackage{minted}

\begin{document}
\begin{minted}{c}
int main() {
    printf("hello, world");
    return 0;
}
\end{minted}
\end{document}
```

By compiling the source file like this:

```
$ pdflatex -shell-escape minimal
```

we end up with the following output in `minimal.pdf`:

```c
int main() {
    printf("hello, world");
    return 0;
}
```

3.3 Formatting source code

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

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

```python
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 a single line of source code, you can alternatively use a shorthand notation:

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

```python
import this
```

This typesets a single line of code using a command rather than an environment, so it saves a little typing, but its output is equivalent to that of the `minted` environment.

The code is delimited by a pair of identical characters, similar to how `\verb` works. The complete syntax is
`\mint[`⟨*options*⟩`]{`⟨*language*⟩`}`⟨*delim*⟩⟨*code*⟩⟨*delim*⟩
, where the code delimiter can be almost any punctuation character. The ⟨*code*⟩ may also be

通过这样编译源文件：

```
$ pdflatex -shell-escape minimal
```

我们得到了以下输出结果在`minimal.pdf`中：

```c
int main() {
    printf("hello, world");
    return 0;
}
```

3.3 格式化源代码

使用 minted 非常简单。例如，要高亮一些 Python 源代码，我们可以使用以下代码片段（右侧是结果）：

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

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

可选地，该环境接受以`key=value`符号表示的一些选项，下面将对其进行详细描述。

对于一行源代码，您也可以使用简写符号：

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

```python
import this
```

这使用了一个命令而不是一个环境来排版一行代码，因此它节省了一些输入，但其输出与`minted`环境的输出是等效的。

代码由一对相同的字符界定，类似于`\verb`的工作方式。完整的语法是
`\mint[`⟨*options*⟩`]{`⟨*language*⟩`}`⟨ *代码界定符*⟩⟨*code*⟩⟨ *代码界定符*⟩
，其中代码界定符可以是几乎任何标点字符。如果⟨*code*⟩ 本身不包含不匹配的花括号，则⟨*code*⟩ 也可以由匹配的花括号`{}`界定。同样，该命令支持一些下面描述

delimited with matched curly braces `{}`, so long as ⟨*code*⟩ itself does not contain unmatched curly braces. Again, this command supports a number of options described below.

Note that the `\mint` command **is not for inline use**. Rather, it is a shortcut for `minted` when only a single line of code is present. The `\mintinline` command is provided for inline use.

`\mintinline` Code can be typeset inline:

```
X\mintinline{python}{print(x**2)}X
```

X`print(x**2)`X

The syntax is `\mintinline[`⟨*options*⟩`]{`⟨*language*⟩`}`⟨*delim*⟩⟨*code*⟩⟨*delim*⟩. The delimiters can be a pair of characters, as for `\mint`. They can also be a matched pair of curly braces, `{}`.

The command has been carefully crafted so that in most cases it will function correctly when used inside other commands.[2]

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

3.4 Using different styles

`\usemintedstyle` Instead of using the default style you may choose another stylesheet provided by Pygments. This may be done via the following:

```
\usemintedstyle{name}
```

The full syntax is `\usemintedstyle[`⟨*language*⟩`]{`⟨*style*⟩`}`. The style may be set for the document as a whole (no language specified), or only for a particular language. Note that the style may also

[2] For example, `\mintinline` works in footnotes! The main exception is when the code contains the percent `%` or hash `#` characters, or unmatched curly braces.

的选

3.4 代码块选项

请注意，`\mint` 命令**不能用于内联使用**。相反，它是`minted`的快捷方式，仅适用于单行代码。`\mintinline`命令用于内联使用。

`\mintinline` 代码可以内联排版：

```
X\mintinline{python}{print(x**2)}X
```

Xprint(x**2)X

该命令的语法是 `\mintinline[`⟨*options*⟩`]{`⟨*language*⟩`}`⟨*delim*⟩⟨*code*⟩⟨*delim*⟩。分隔符可以是一对字符，就像 `\mint` 一样。也可以是一对匹配的花括号，`{}`。

该命令已经被精心设计，以便在大多数情况下，当在其他命令中使用时，它可以正确地工作。[2]

最后，还有一个 `\inputminted` 命令用于读取和格式化整个文件。其语法是 `\inputminted[`⟨*options*⟩`]{`⟨*language*⟩`}{`⟨*filename*⟩`}`。

3.5 使用不同的样式

`\usemintedstyle` 您可以选择使用 Pygments 提供的其他样式表，而不是使用默认样式。可以通过以下方式实现：

```
\usemintedstyle{name}
```

完整的语法是 `\usemintedstyle[`⟨*language*⟩`]{`⟨*style*⟩`}`。样式可以为整个文档设置（不指定语言），也可以为特定语言设置。请注意，样式也可以通过 `\setminted` 和

[2] 例如，`\mintinline` 可以在脚注中工作！唯一的例外是代码中包含百分号 `%` 或井号 `#` 字符，或者未匹配的花括号。

be set via `\setminted` and via the optional argument for each command and environment.[3]

To get a list of all available stylesheets, see the online demo at the Pygments website or execute the following command on the command line:

```
$ pygmentize -L styles
```

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

3.5 Supported languages

Pygments supports over 300 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
```

[3]Version 2.0 added the optional language argument and removed the restriction that the command be used in the preamble.

每个命令和环境的可选参数设置。[3]

要获取所有可用样式表的列表，请参阅 Pygments 网站上的在线演示，或在命令行上执行以下命令：

```
$ pygmentize -L styles
```

创建自己的样式也非常简单。只需按照 Pygments 网站上提供的说明进行操作。

3.6 支持的语言

Pygments 支持超过 300 种不同的编程语言、模板语言和其他标记语言。要查看当前支持的语言的详尽列表，请使用以下命令：

```
$ pygmentize -L lexers
```

[3]版本 2.0 添加了可选的语言参数，并删除了该命令必须在导言区使用的限制。

4 Floating listings

4 浮动的代码清单

minted provides the `listing` environment to wrap around a source code block. This puts the code into a floating box, with the default placement `tbp` like figures and tables. You can also provide a `\caption` and a `\label` for such a listing in the usual way (that is, as for the `figure` and `table` environments):

minted 提供了 `listing` 环境来包装源代码块。这将把代码放入一个浮动的框中，默认的位置是 `tbp`，就像图表一样。您还可以像使用 `figure` 和 `table` 环境一样，为这样的清单提供 `\caption` 和 `\label`：

`listing`

```
\begin{listing}[H]
  \mint{cl}/(car (cons 1 '(2)))/
  \caption{Example of a listing.}
  \label{lst:example}
\end{listing}

Listing \ref{lst:example} contains an example of a listing.
```

```
\begin{listing}[H]
  \mint{cl}/(car (cons 1 '(2)))/
  \caption{Example of a listing.}
  \label{lst:example}
\end{listing}

Listing \ref{lst:example} contains an example of a listing.
```

will yield:

将生成：

```
(car (cons 1 '(2)))
```

Listing 1: Example of a listing.

Listing ?? contains an example of a listing.

The default `listing` placement can be modified easily. When the package option `newfloat=false` (default), the float package is used to create the `listing` environment. Placement can be modified by redefining `\fps@listing`. For example,

您可以轻松地修改默认的 `listing` 位置。当包选项 `newfloat=false`（默认）时，使用 float 宏包来创建 `listing` 环境。可以通过重新定义 `\fps@listing` 来修改位置。例如，

```
\makeatletter
\renewcommand{\fps@listing}{htp}
\makeatother
```

When `newfloat=true`, the more powerful newfloat package is used to create the `listing` environment. In that case, newfloat commands are available to customize `listing`:

当 `newfloat=true` 时，使用更强大的 newfloat 宏包来创建 `listing` 环境。在这种情况下，可以使用 newfloat 命令来自定义 `listing`：

```
\SetupFloatingEnvironment{listing}{placement=htp}
```

`\listoflistings` The `\listoflistings` macro will insert a list of all (floated) listings in the document:

`\listoflistings` 命令将在文档中插入所有（浮动的）清单的列表：

`\listoflistings`

List of Listings

1 Example of a listing. 1

Customizing the `listing` environment

自定义 `listing` 环境

By default, the `listing` environment is created using the float package. In that case, the `\listingscaption` and `\listoflistingscaption` macros described below may be used to customize the caption and list of listings. If minted is loaded with the `newfloat` option, then the `listing` environment will be created with the more powerful newfloat package instead. newfloat is part of caption, which provides many options for customizing captions.

默认情况下，使用 float 宏包来创建 `listing` 环境。在这种情况下，可以使用下面描述的 `\listingscaption` 和 `\listoflistingscaption` 宏来自定义标题和清单列表。如果加载了带有 `newfloat` 选项的 minted 宏包，则 `listing` 环境将使用功能更强大的 newfloat 宏包创建。newfloat 是 caption 的一部分，它提供了许多自定义标题的选项。

When newfloat is used to create the `listing` environment, customization should be achieved using

当使用 newfloat 来创建 `listing` 环境时，可使用 `\SetupFloatingEnvironment`

newfloat's \SetupFloatingEnvironment command. For example, the string "Listing" in the caption could be changed to "Program code" using

```
\SetupFloatingEnvironment{listing}{name=Program code}
```

And "List of Listings" could be changed to "List of Program Code" with

```
\SetupFloatingEnvironment{listing}{listname=List of Program Code}
```

Refer to the newfloat and caption documentation for additional information.

\listingscaption (Only applies when package option newfloat is not used.) The string "Listing" in a listing's caption can be changed. To do this, simply redefine the macro \listingscaption, for example:

```
\renewcommand{\listingscaption}{Program code}
```

\listoflistingscaption (Only applies when package option newfloat is not used.) Likewise, the caption of the listings list, "List of Listings," can be changed by redefining \listoflistingscaption:

```
\renewcommand{\listoflistingscaption}{List of Program Code}
```

5 Options

5.1 Package options

chapter To control how LaTeX counts the listing floats, you can pass either the section or chapter option when loading the minted package. For example, the following will cause listings to be counted by chapter:

```
\usepackage[chapter]{minted}
```

命令来实现自定义。例如，可以使用以下方式将标题中的字符串 "Listing" 更改为 "程序代码"：

```
\SetupFloatingEnvironment{listing}{name= 程序代码}
```

并且可以使用以下方式将 "List of Listings" 更改为 "程序代码列表"：

```
\SetupFloatingEnvironment{listing}{listname= 程序代码列表}
```

请参考 newfloat 和 caption 文档以获取更多信息。

\listingscaption （仅适用于未使用包选项 newfloat 的情况。）可以更改清单标题中的字符串 "Listing"。只需重新定义宏 \listingscaption 即可，例如：

```
\renewcommand{\listingscaption}{程序代码}
```

\listoflistingscaption 同样，可以通过重新定义 \listoflistingscaption 来更改清单列表的标题 "List of Listings"（仅适用于未使用包选项 newfloat 的情况。）：

```
\renewcommand{\listoflistingscaption}{程序代码列表}
```

5 选项

5.1 宏包选项

chapter 在加载 minted 宏包时，可以通过传递section或chapter选项来控制 LaTeX 如何计数listing浮动体。例如，以下代码将使得列表按章节计数：

```
\usepackage[chapter]{minted}
```

cache=⟨*boolean*⟩ (默认值: true)

minted works by saving code to a temporary file, highlighting the code via Pygments and saving the output to another temporary file, and inputting the output into the LaTeX document. This process can become quite slow if there are several chunks of code to highlight. To avoid this, the package provides a `cache` option. This is on by default.

minted 通过将代码保存到临时文件中，使用 Pygments 对代码进行高亮，并将输出保存到另一个临时文件中，然后将输出插入到 LaTeX 文档中。如果需要高亮显示多个代码块，这个过程可能会变得非常慢。为了避免这种情况，该宏包提供了一个`cache`选项，默认情况下为开启状态。

cache=⟨*boolean*⟩ (default: true)

The `cache` option creates a directory `_minted-`⟨*jobname*⟩ in the document's root directory (this may be customized with the `cachedir` option).[4] Files of highlighted code are stored in this directory, so that the code will not have to be highlighted again in the future. In most cases, caching will significantly speed up document compilation.

`cache`选项会在文档的根目录下创建一个名为`_minted-`⟨*jobname*⟩ 的目录（可以使用`cachedir`选项自定义目录名）。[4] 高亮显示的代码文件将存储在该目录中，以便以后不需要再次进行高亮显示。在大多数情况下，缓存会显著加快文档的编译速度。

Cached files that are no longer in use are automatically deleted.[5]

不再使用的缓存文件会自动删除。[5]

cachedir=⟨*directory*⟩ (默认值: `_minted-`⟨*jobname*⟩)

This allows the directory in which cached files are stored to be specified. Paths should use forward slashes, even under Windows.

该选项允许指定存储缓存文件的目录。路径应该使用正斜杠，即使在 Windows 下也是如此。

cachedir=⟨*directory*⟩ (def: `_minted-`⟨*jobname*⟩)

Special characters must be escaped. For example, `cachedir=~/mintedcache` would not work because the tilde `~` would be converted into the LaTeX commands for a non-breaking space, rather than being treated literally. Instead, use `\string~/mintedcache`, `\detokenize{~/mintedcache}`, or an equivalent solution.

特殊字符必须进行转义。例如，`cachedir=~/mintedcache`是无效的，因为波浪号`~`会被转换为非换行空格，而不是按字面意义对待。相反，使用`\string~/mintedcache`、`\detokenize{~/mintedcache}`或类似的解决方案。

Paths may contain spaces, but only if the entire ⟨*directory*⟩ is wrapped in curly braces `{}`, and only if the spaces are quoted. For example,

路径可以包含空格，但只有在整个⟨*directory*⟩ 被放在花括号`{}`中，并且空格被引用时才可以。例如，

```
cachedir = {\detokenize{~/"minted cache"/"with spaces"}}
```

```
cachedir = {\detokenize{~/"minted cache"/"with spaces"}}
```

Note that the cache directory is relative to the `outputdir`, if an `outputdir` is specified.

请注意，如果指定了`outputdir`，则缓存目录是相对于`outputdir`的。

finalizecache=⟨*boolean*⟩ (默认值: false)

In some cases, it may be desirable to use minted in an environment in which `-shell-escape` is not allowed. A document might be submitted to a publisher or preprint server or used with an online service that does not support `-shell-escape`. This is possible as long as minted content does not need to be modified.

在某些情况下，可能希望在不允许`-shell-escape`的环境中使用 minted。例如，文档可能会被提交给出版商、预印版本服务器或与不支持`-shell-escape`的在线服务一起使用。只要不需要修改 minted 内容，就可以做到这一点。

finalizecache=⟨*boolean*⟩ (default: false)

[4] The directory is actually named using a "sanitized" copy of ⟨*jobname*⟩, in which spaces and asterisks have been replaced by underscores, and double quotation marks have been stripped. If the file name contains spaces, `\jobname` will contain a quote-wrapped name, except under older versions of MiKTeX which used the name with spaces replaced by asterisks. Using a "sanitized" ⟨*jobname*⟩ is simpler than accomodating the various escaping conventions.

[5] This depends on the main auxiliary file not being deleted or becoming corrupted. If that happens, you could simply delete the cache directory and start over.

[4] 实际上，该目录的命名是使用了"清理"过的⟨*jobname*⟩ 的副本，其中空格和星号被替换为下划线，双引号被删除。如果文件名包含空格，`\jobname` 将包含带引号的名称，除了在旧版本的 MiKTeX 中，该名称将使用将空格替换为星号的名称。使用"清理"过的⟨*jobname*⟩ 比适应各种转义约定更简单。

[5] 这取决于主辅助文件未被删除或损坏。如果发生这种情况，您只需删除缓存目录并重新开始。

Compiling with the `finalizecache` option prepares the cache for use in an environment without `-shell-escape`.[6] Once this has been done, the `finalizecache` option may be swapped for the `frozencache` option, which will then use the frozen (static) cache in the future, without needing `-shell-escape`.

使用`finalizecache`选项编译缓存以供在不需要`-shell-escape`的环境中使用。[6]完成此操作后，可以将`finalizecache`选项替换为`frozencache`选项，以后就可以在不需要`-shell-escape`的情况下使用冻结（静态）缓存。

`fontencoding=`⟨*encoding*⟩ (默认值: ⟨*doc encoding*⟩)

Set font encoding used for typesetting code.

For example, `fontencoding=T1`.

设置用于排版代码的字体编码。

例如，`fontencoding=T1`。

`fontencoding=`⟨*encoding*⟩ (default: ⟨*doc encoding*⟩)

`frozencache=`⟨*boolean*⟩ (默认值: false)

Use a frozen (static) cache created with the `finalizecache` option. When `frozencache` is on, `-shell-escape` is not needed, and Python and Pygments are not required. In addition, any external files accessed through `\inputminted` are no longer necessary.

This option must be used with care. A document *must* be in final form, as far as minted is concerned, *before* `frozencache` is turned on, and the document *must* have been compiled with `finalizecache`. When this option is on, minted content cannot be modified, except by editing the cache files directly. Changing any minted settings that require Pygments or Python is not possible. If minted content is incorrectly modified after `frozencache` is turned on, minted *cannot* detect the modification.

If you are using `frozencache`, and want to verify that minted settings or content have not been modified in an invalid fashion, you can test the cache using the following procedure.

1. Obtain a copy of the cache used with `frozencache`.
2. Compile the document in an environment that supports `-shell-escape`, with `finalizecache=true` and `frozencache=false`. This essentially regenerates the frozen (static) cache.
3. Compare the original cache with the newly generated cache. Under Linux and OS X, you could use `diff`; under Windows, you probably want `fc`. If minted content and settings have not been modified in an invalid fashion, all files will be identical (assuming that compatible versions of Pygments are

使用使用`finalizecache`选项创建的冻结（静态）缓存。当开启`frozencache`时，无需`-shell-escape`，也不需要 Python 和 Pygments。此外，通过`\inputminted`访问的任何外部文件也不再需要。

`frozencache=`⟨*boolean*⟩ (default: false)

请谨慎使用此选项。在启用`frozencache`之前，文档在 minted 看来必须是最终形式，而且必须使用`finalizecache`编译文档。开启此选项后，除非直接编辑缓存文件，否则无法修改 minted 内容。不可能更改任何需要 Pygments 或 Python 的 minted 设置。如果在开启`frozencache`后错误地修改了 minted 内容，minted 将无法检测到这些修改。

如果使用`frozencache`，并且希望验证 minted 的设置或内容是否以无效的方式进行了修改，可以使用以下步骤测试缓存。

1. 获取使用`frozencache`的缓存的副本。
2. 在支持`-shell-escape`的环境中使用`finalizecache=true`和`frozencache=false`编译文档。这实际上重新生成了冻结（静态）缓存。
3. 将原始缓存与新生成的缓存进行比较。在 Linux 和 OS X 下，可以使用`diff`命令；在 Windows 下，可能需要使用`fc`命令。

[6] Ordinarily, cache files are named using an MD5 hash of highlighting settings and highlighted text. `finalizecache` renames cache files using a `listing<number>.pygtex` scheme. This makes it simpler to match up document content and cache files, and is also necessary for the XeTeX engine since prior to TeX Live 2016 it lacked the built-in MD5 capabilities that pdfTeX and LuaTeX have.

[6] 通常，缓存文件的命名是使用高亮设置和高亮文本的 MD5 哈希值。使用 `finalizecache` 选项，会使用 `listing<number>.pygtex` 方案重命名缓存文件。这样可以更容易地匹配文档内容和缓存文件，并且对于没有内置 MD5 功能的 XeTeX 引擎（在 TeX Live 2016 之前），这是必需的。

used for both caches).

draft=⟨*boolean*⟩ (默认值: false)

This uses fancyvrb alone for all typesetting; Pygments is not used. This trades syntax highlighting and some other minted features for faster compiling. Performance should be essentially the same as using fancyvrb directly; no external temporary files are used. Note that if you are not changing much code between compiles, the difference in performance between caching and draft mode may be minimal. Also note that `draft` settings are typically inherited from the document class.

这个选项只使用 fancyvrb 进行排版，而不使用 Pygments 进行语法高亮和其他一些 minted 特性，以加快编译速度。性能应该与直接使用 fancyvrb 相同，不使用外部临时文件。请注意，如果在编译之间没有改变太多的代码，则缓存和草稿模式之间的性能差异可能很小。还要注意，草稿模式的设置通常从文档类继承而来。

draft=⟨*boolean*⟩ (default: false)

Draft mode does not support `autogobble`. Regular `gobble`, `linenos`, and most other options not related to syntax highlighting will still function in draft mode.

在草稿模式下，不支持`autogobble`选项。普通的`gobble`、`linenos`和大多数与语法高亮无关的其他选项仍然可以在草稿模式下使用。

Documents can usually be compiled without shell escape in draft mode. The ifplatform package may issue a warning about limited functionality due to shell escape being disabled, but this may be ignored in almost all cases. (Shell escape is only really required if you have an unusual system configuration such that the `\ifwindows` macro must fall back to using shell escape to determine the system. See the ifplatform documentation for more details: http://www.ctan.org/pkg/ifplatform.)

在草稿模式下，通常可以不使用 shell escape 编译文档。但是，ifplatform 宏包可能会发出有关功能受限的警告，因为 shell escape 被禁用了，但在几乎所有情况下都可以忽略这个警告。(只有在您的系统配置非常特殊，以至于`\ifwindows`宏必须回退到使用 shell escape 来确定系统时，才真正需要 shell escape。有关详细信息，请参阅 ifplatform 的文档：http://www.ctan.org/pkg/ifplatform。)

If the `cache` option is set, then all existing cache files will be kept while draft mode is on. This allows caching to be used intermitently with draft mode without requiring that the cache be completely recreated each time. Automatic cleanup of cached files will resume as soon as draft mode is turned off. (This assumes that the auxiliary file has not been deleted in the meantime; it contains the cache history and allows automatic cleanup of unused files.)

如果设置了`cache`选项，那么在草稿模式下会保留所有现有的缓存文件。这样可以在草稿模式和缓存模式之间交替使用缓存，而无需每次都完全重建缓存。一旦关闭草稿模式，未使用的缓存文件的自动清理将重新开始。(前提是辅助文件在此期间没有被删除；它包含缓存历史记录，并允许自动清理未使用的文件。)

final=⟨*boolean*⟩ (默认值: true)

This is the opposite of `draft`; it is equivalent to `draft=false`. Again, note that `draft` and `final` settings are typically inherited from the document class.

这是`draft`的相反，相当于`draft=false`。同样，注意`draft`和`final`的设置通常从文档类继承而来。

final=⟨*boolean*⟩ (default: true)

kpsewhich=⟨*boolean*⟩ (默认值: false)

This option uses `kpsewhich` to locate files that are to be highlighted. Some build tools such as `texi2pdf` function by modifying `TEXINPUTS`; in some cases, users may customize `TEXINPUTS` as well. The `kpsewhich` option allows minted to work with such configurations.

此选项使用`kpsewhich`来定位需要进行语法高亮的文件。一些构建工具(如`texi2pdf`)通过修改`TEXINPUTS`来进行操作；在某些情况下，用户也可以自定义`TEXINPUTS`。`kpsewhich`选项允许 minted 与此类配置一起工作。

kpsewhich=⟨*boolean*⟩ (default: false)

This option may add a noticeable amount of overhead on some systems, or with some system configurations.

此选项可能会在某些系统或某些系统配置下增加明显的开销。

This option does *not* make minted work with the `-output-directory` and `-aux-directory` command-line options for LaTeX. For those, see the `outputdir` package option.

此选项*不会*使 minted 与 LaTeX 的`-output-directory`和`-aux-directory`命令行选项兼容。对于这些选项，请参阅`outputdir`包选项。

使用 kpsewhich 可以快速查找 TeX 系统中的文件，例如：

- 查找一个宏包的路径：kpsewhich <package.sty>
- 查找一个字体文件的路径：kpsewhich <font.ttf>
- 查找一个文档类的路径：kpsewhich <class.cls>
- 于编写脚本或 Makefile，以自动查找和引用 TeX 相关文件。

`langlinenos=`⟨*boolean*⟩（默认值：`false`）

minted uses the fancyvrb package behind the scenes for the code typesetting. fancyvrb provides an option `firstnumber` that allows the starting line number of an environment to be specified. For convenience, there is an option `firstnumber=last` that allows line numbering to pick up where it left off. The `langlinenos` option makes `firstnumber` work for each language individually with all `minted` and `\mint` usages. For example, consider the code and output below.

minted 在代码排版时使用 fancyvrb 宏包。fancyvrb 提供了一个选项`firstnumber`，允许指定环境的起始行号。为了方便起见，还有一个选项`firstnumber=last`，允许行号从上一个环境的位置继续编号。`langlinenos`选项使得`firstnumber`在每种语言的所有`minted`和`\mint`使用中都起作用。例如，考虑以下代码和输出。

`langlinenos=`⟨*boolean*⟩ (default: `false`)

```
\begin{minted}[linenos]{python}
def f(x):
    return x**2
\end{minted}

\begin{minted}[linenos]{ruby}
def func
    puts "message"
end
\end{minted}

\begin{minted}[linenos, firstnumber=last]{python}
def g(x):
    return 2*x
\end{minted}
```

```
1 def f(x):
2     return x**2
1 def func
2     puts "message"
3 end
3 def g(x):
4     return 2*x
```

Without the `langlinenos` option, the line numbering in the second Python environment would not pick up where the first Python environment left off. Rather, it would pick up with the Ruby line numbering.

如果没有使用`langlinenos`选项，第二个 Python 环境中的行号不会从第一个 Python 环境的行号继续，而是从 Ruby 行号开始编号。

`newfloat=`⟨*boolean*⟩（默认值：`false`）

By default, the `listing` environment is created using the float package. The `newfloat` option creates the environment using newfloat instead. This provides better integration with the caption package.

默认情况下，使用 float 宏包创建`listing`环境。`newfloat`选项改用 newfloat 创建该环境，以更好地与 caption 宏包集成。

`newfloat=`⟨*boolean*⟩ (default: `false`)

outputdir=⟨*directory*⟩ (默认值: ⟨*none*⟩)

The `-output-directory` and `-aux-directory` (MiKTeX) command-line options for LaTeX cause problems for minted, because the minted temporary files are saved in `<outputdir>`, but minted still looks for them in the document root directory. There is no way to access the value of the command-line option so that minted can automatically look in the right place. But it is possible to allow the output directory to be specified manually as a package option.

The output directory should be specified using an absolute path or a path relative to the document root directory. Paths should use forward slashes, even under Windows. Special characters must be escaped, while spaces require quoting and need the entire ⟨*directory*⟩ to be wrapped in curly braces `{}`. See `cachedir` above for examples of escaping and quoting.

section

To control how LaTeX counts the `listing` floats, you can pass either the `section` or `chapter` option when loading the minted package.

outputdir=⟨*directory*⟩ (default: ⟨*none*⟩)

LaTeX 的`-output-directory`和`-aux-directory`（MiKTeX）命令行选项会给 minted 带来问题，因为 minted 临时文件保存在`<outputdir>`中，但 minted 仍然在文档根目录中查找它们。无法访问命令行选项的值，以使 minted 能自动在正确的位置查找。但可以允许手动指定输出目录作为包选项。

输出目录应使用绝对路径或相对于文档根目录的路径来指定。路径应该使用斜杠，即使在 Windows 下也是如此。特殊字符必须转义，而空格需要用引号引起来，并且需要将整个目录⟨*directory*⟩用花括号 `{}` 括起来。参见上面的 `cachedir` 的示例来了解转义和引用的用法。

section

要控制 LaTeX 对 `listing` 浮动体的计数方式，可以在加载 minted 宏包时传递 `section` 或 `chapter` 选项。

5.2 Macro option usage

5.2 宏选项用法

All minted highlighting 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:

所有的 minted 高亮命令都接受相同的选项集合。选项以逗号分隔的 `key=value` 对的形式指定。例如，我们可以指定行数的显示：

```
\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.

也可以完全省略选项值为 `true` 的部分（包括 "=" 符号）。要进一步自定义行数的显示方式，请覆盖 `\theFancyVerbLine` 命令。有关详细信息，请参阅 fancyvrb 文档。

`\mint` accepts the same options:

`\mint` 命令也接受相同的选项：

```
\mint[linenos]{perl}|$x=~/foo/|
```

```
1  $x=~/foo/
```

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

下面是另一个例子：我们希望在注释中使用 LaTeX 的数学模式：

```
\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. There is also an `autogobble` option that detects the length of this whitespace automatically.

为了使你的 LaTeX 代码更具可读性，你可能希望在 `minted` 环境中对代码进行缩进。选项 `gobble` 可以从输出中删除这些不必要的空白字符。还有一个 `autogobble` 选项，可以自动检测这些空白字符的长度。

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

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

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

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

`\setminted` You may wish to set options for the document as a whole, or for an entire language. This is possible via `\setminted[`⟨*language*⟩`]{`⟨*key=value,...*⟩`}`. Language-specific options override document-wide options. Individual command and environment options override language-specific options.

`\setminted` 您可能希望为整个文档或整个语言设置选项。这可以通过 `\setminted[`⟨*language*⟩`]{`⟨*key=value,...*⟩`}` 实现。语言特定的选项会覆盖文档范围的选项。单独的命令和环境选项会覆盖语言特定的选项。

`\setmintedinline` You may wish to set separate options for `\mintinline`, either for the document as a whole or for a specific language. This is possible via `\setmintedinline`. The syntax is `\setmintedinline[`⟨*language*⟩`]{`⟨*key=value,...*⟩`}`. Language-specific options override document-wide options. Individual command options override language-specific options. All settings specified with `\setmintedinline` override those set with `\setminted`. That is, inline settings always have a higher precedence than general settings.

`\setmintedinline` 您可能希望为 `\mintinline` 设置单独的选项，无论是为整个文档还是为特定的语言。这可以通过 `\setmintedinline` 来实现。其语法为 `\setmintedinline[`⟨*language*⟩`]{`⟨*key=value,...*⟩`}`。语言特定的选项会覆盖文档范围的选项。单独的命令选项会覆盖语言特定的选项。所有使用 `\setmintedinline` 设置的选项都会覆盖使用 `\setminted` 设置的选项。也就是说，内联设置始所有的 LaTeX 代码都会比一般设置具有更高的优先级。

5.3 Available options

Following is a full list of available options. For more detailed option descriptions please refer to the fancyvrb and Pygments documentation.

baselinestretch (dimension) (default: ⟨*document default*⟩)
Value to use as for baselinestretch inside the listing.

beameroverlays (boolean) (default: `false`)
Give the `<` and `>` characters their normal text meanings when used with `escapeinside` and `texcomments`, so that beamer overlays of the form `\only<1>{...}` will work.

breakafter (string) (default: ⟨*none*⟩)
Break lines after specified characters, not just at spaces, when `breaklines=true`. Does not apply to `\mintinline`.

For example, `breakafter=-/` would allow breaks after any hyphens or slashes. Special characters given to `breakafter` should be backslash-escaped (usually `#`, `{`, `}`, `%`, `[`, `]`; the backslash `\` may be obtained via `\\`).

For an alternative, see `breakbefore`. When `breakbefore` and `breakafter` are used for the same character, `breakbeforegroup` and `breakaftergroup` must both have the same setting.

```
\begin{minted}[breaklines, breakafter=d]{python}
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
\end{minted}
```

```
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
```

5.3 可用选项

以下是所有可用选项的完整列表。有关更详细的选项描述，请参考 fancyvrb 和 Pygments 的文档。

baselinestretch (尺寸) (default: ⟨ *文档默认值*⟩)
在列表中使用的基线伸展值。

beameroverlays (布尔值) (default: `false`)
当与 `escapeinside` 和 `texcomments` 一起使用时，赋予 `<` 和 `>` 字符其正常的文本含义，以便 beamer 的形式为 `\only<1>{...}` 的叠加效果可以正常工作。

breakafter (字符串) (default: ⟨*none*⟩)
在指定字符之后换行，而不仅仅是在空格处换行，当 `breaklines=true` 时。不适用于 `\mintinline`。

例如，`breakafter=-/` 允许在任何连字符或斜杠之后换行。给 `breakafter` 提供的特殊字符应该进行反斜杠转义（通常是 `#`、`{`、`}`、`%`、`[`、`]`；反斜杠 `\` 可以通过 `\\` 获得）。

对于一个替代方案，请参见 `breakbefore`。当 `breakbefore` 和 `breakafter` 同时用于同一个字符时，`breakbeforegroup` 和 `breakaftergroup` 必须具有相同的设置。

```
\begin{minted}[breaklines, breakafter=d]{python}
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
\end{minted}
```

```
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCould⌟
↪ NeverFitOnOneLine'
```

breakaftergroup (boolean) (default: true)

When breakafter is used, group all adjacent identical characters together, and only allow a break after the last character. When breakbefore and breakafter are used for the same character, breakbeforegroup and breakaftergroup must both have the same setting.

breakaftergroup (布尔值) (default: true)

当使用 breakafter 时，将所有相邻的相同字符分组在一起，并且只允许在最后一个字符之后换行。当 breakbefore 和 breakafter 同时用于同一个字符时，breakbeforegroup 和 breakaftergroup 必须具有相同的设置。

breakaftersymbolpre (string) (default: \,\footnotesize\ensuremath{_\rfloor}, 」)

The symbol inserted pre-break for breaks inserted by breakafter.

breakaftersymbolpre (字符串) (default: \,\footnotesize\ensuremath{_\rfloor}, 」)

使用 breakafter 插入的换行符前的符号。

breakaftersymbolpost (string) (default: ⟨*none*⟩)

The symbol inserted post-break for breaks inserted by breakafter.

breakaftersymbolpost (字符串) (default: ⟨*none*⟩)

使用 breakafter 插入的换行符后的符号。

breakanywhere (boolean) (default: false)

Break lines anywhere, not just at spaces, when breaklines=true. Does not apply to \mintinline.

```
\begin{minted}[breaklines, breakanywhere]{python}
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
\end{minted}
```

```
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
```

breakanywhere (布尔值) (default: false)

在 breaklines=true 时，在任意位置而不仅仅是在空格处换行。不适用于\mintinline。

```
\begin{minted}[breaklines, breakanywhere]{python}
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
\end{minted}
```

```
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeve」
↪ rFitOnOneLine'
```

breakanywheresymbolpre (string) (default: \,\footnotesize\ensuremath{_\rfloor}, 」)

The symbol inserted pre-break for breaks inserted by breakanywhere.

breakanywheresymbolpre (字符串) (default: \,\footnotesize\ensuremath{_\rfloor}, 」)

使用 breakanywhere 插入的换行符前的符号。

breakanywheresymbolpost (string) (default: ⟨*none*⟩)

The symbol inserted post-break for breaks inserted by breakanywhere.

breakanywheresymbolpost (字符串) (default: ⟨*none*⟩)

使用 breakanywhere 插入的换行符后的符号。

breakbefore (string) (default: ⟨*none*⟩)

Break lines before specified characters, not just at spaces, when breaklines=true. Does not apply to \mintinline.

For example, breakbefore=A would allow breaks before capital A's. Special characters given to

breakbefore (字符串) (default: ⟨*none*⟩)

在指定的字符之前换行，而不仅仅是在空格处换行，当 breaklines=true 时。不适用于\mintinline。

例如，breakbefore=A 允许在大写 A 之前断行。breakbefore 所给出的特殊字符

`breakbefore` should be backslash-escaped (usually `#`, `{`, `}`, `%`, `[`, `]`; the backslash `\` may be obtained via `\\`).

应该以反斜杠转义（通常为 `#`, `{`, `}`, `%`, `[`, `]`; 反斜杠 `\` 可以通过 `\\` 获得。

For an alternative, see `breakafter`. When `breakbefore` and `breakafter` are used for the same character, `breakbeforegroup` and `breakaftergroup` must both have the same setting.

对于另一种选择，请参见 `breakafter`。当 `breakbefore` 和 `breakafter` 同时用于同一个字符时，`breakbeforegroup` 和 `breakaftergroup` 必须具有相同的设置。

```
\begin{minted}[breaklines, breakbefore=A]{python}
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
\end{minted}
```

```
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
```

```
\begin{minted}[breaklines, breakbefore=A]{python}
some_string = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldNeverFitOnOneLine'
\end{minted}
```

```
some_string = 'SomeTextThatGoesOn」
 ↪  AndOnForSoLongThatItCouldNeverFitOnOneLine'
```

`breakbeforegroup` (boolean) (default: `true`)

When `breakbefore` is used, group all adjacent identical characters together, and only allow a break before the first character. When `breakbefore` and `breakafter` are used for the same character, `breakbeforegroup` and `breakaftergroup` must both have the same setting.

`breakbeforegroup` **(布尔值)** (default: `true`)

当使用 `breakbefore` 时，将所有相邻的相同字符分组在一起，并且只允许在第一个字符之前断行。当 `breakbefore` 和 `breakafter` 同时用于同一个字符时，`breakbeforegroup` 和 `breakaftergroup` 必须具有相同的设置。

`breakbeforesymbolpre` (string) (default: `\,\footnotesize\ensuremath{_\rfloor}, 」`)

The symbol inserted pre-break for breaks inserted by `breakbefore`.

`breakbeforesymbolpre` **(字符串)** (default: `\,\footnotesize\ensuremath{_\rfloor}, 」`)

在 `breakbefore` 插入的断行符号之前插入的符号。

`breakbeforesymbolpost` (string) (default: ⟨*none*⟩)

The symbol inserted post-break for breaks inserted by `breakbefore`.

`breakbeforesymbolpost` **(字符串)** (default: ⟨*none*⟩)

在 `breakbefore` 插入的断行符号之后插入的符号。

`breakbytoken` (boolean) (default: `false`)

Only break lines at locations that are not within tokens; prevent tokens from being split by line breaks. By default, `breaklines` causes line breaking at the space nearest the margin. While this minimizes the number of line breaks that are necessary, it can be inconvenient if a break occurs in the middle of a string or similar token.

This is not compatible with `draft` mode. A complete list of Pygments tokens is available at http://pygments.org/docs/tokens/. If the breaks provided by `breakbytoken` occur in unexpected locations, it may indicate a bug or shortcoming in the Pygments lexer for the language.

`breakbytoken` **(布尔值)** (default: `false`)

仅在不在标记内部的位置断行，防止标记被分割成多行。默认情况下，`breaklines` 会在最靠近边缘的空格处断行。虽然这最小化了所需的断行数量，但如果断行发生在字符串或类似标记的中间，可能会不方便。

此选项与 `draft` 模式不兼容。可以在http://pygments.org/docs/tokens/上找到完整的 Pygments 标记列表。如果由 `breakbytoken` 提供的断行发生在意外的位置，可能表示语言的 Pygments 词法分析器存在错误或不足。

breakbytokenanywhere (boolean) (default: false)

Like breakbytoken, but also allows line breaks between immediately adjacent tokens, not just between tokens that are separated by spaces. Using breakbytokenanywhere with breakanywhere is redundant.

breakbytokenanywhere (布尔值) (default: false)

类似于 breakbytoken，但也允许在相邻标记之间进行断行，而不仅仅是在由空格分隔的标记之间。在使用 breakanywhere 时，使用 breakbytokenanywhere 是多余的。

breakautoindent (boolean) (default: true)

When a line is broken, automatically indent the continuation lines to the indentation level of the first line. When breakautoindent and breakindent are used together, the indentations add. This indentation is combined with breaksymbolindentleft to give the total actual left indentation. Does not apply to \mintinline.

breakautoindent (布尔值) (default: true)

当换行时，自动将续行缩进到第一行的缩进级别。当 breakautoindent 和 breakindent 一起使用时，缩进级别会相加。此缩进与 breaksymbolindentleft 结合在一起，形成实际的左缩进总量。不适用于\mintinline。

breakindent (dimension) (default: ⟨*breakindentnchars*⟩)

When a line is broken, indent the continuation lines by this amount. When breakautoindent and breakindent are used together, the indentations add. This indentation is combined with breaksymbolindentleft to give the total actual left indentation.

Does not apply to \mintinline.

breakindent (尺寸) (default: ⟨*breakindentnchars*⟩)

当一行被断行时，将连续行缩进这个量。当breakautoindent和breakindent一起使用时，缩进会相加。此缩进与breaksymbolindentleft相结合，给出总的实际左缩进。

不适用于\mintinline。

breakindentnchars (integer) (default: 0)

This allows breakindent to be specified as an integer number of characters rather than as a dimension (assumes a fixed-width font).

breakindentnchars (整数) (default: 0)

这允许将breakindent指定为整数个字符而不是作为尺寸（假设使用等宽字体）。

breaklines (boolean) (default: false)

Automatically break long lines in minted environments and \mint commands, and wrap longer lines in \mintinline.

By default, automatic breaks occur at space characters. Use breakanywhere to enable breaking anywhere; use breakbytoken, breakbytokenanywhere, breakbefore, and breakafter for more fine-tuned breaking. Currently, only breakbytoken and breakbytokenanywhere work with \mintinline. Using escapeinside to escape to LaTeX and then insert a manual break is also an option. For example, use escapeinside=, and then insert \\ at the appropriate point. (Note that escapeinside does not work within strings.)

breaklines (布尔值) (default: false)

在 minted 环境和\mint 命令中自动断行长行，并在\mintinline 中换行。

默认情况下，自动换行发生在空格字符处。使用 breakanywhere 可以在任何位置进行换行;使用 breakbytoken、breakbytokenanywhere、breakbefore 和 breakafter 可以进行更精细的换行控制。目前，只有 breakbytoken 和 breakbytokenanywhere 能够与\mintinline一起使用。还可以使用 escapeinside 来转义到 LaTeX 并插入手动换行。例如，使用 escapeinside=，然后在适当的位置插入\\。（注意，escapeinside 在字符串内部不起作用。）

```
...text.
\begin{minted}[breaklines]{python}
def f(x):
    return 'Some text ' + str(x)
\end{minted}
```

```
...text.
 def f(x):
     return 'Some text ' + str(x)
```

```
...text.
\begin{minted}[breaklines]{python}
def f(x):
    return 'Some text ' + str(x)
\end{minted}
```

```
...text.
 def f(x):
     return 'Some text ' +
     ↪  str(x)
```

Breaking in `minted` and `\mint` may be customized in several ways. To customize the indentation of broken lines, see `breakindent` and `breakautoindent`. To customize the line continuation symbols, use `breaksymbolleft` and `breaksymbolright`. To customize the separation between the continuation symbols and the code, use `breaksymbolsepleft` and `breaksymbolsepright`. To customize the extra indentation that is supplied to make room for the break symbols, use `breaksymbolindentleft` and `breaksymbolindentright`. Since only the left-hand symbol is used by default, it may also be modified using the alias options `breaksymbol`, `breaksymbolsep`, and `breaksymbolindent`. Note than none of these options applies to `\mintinline`, since they are not relevant in the inline context.

可以通过多种方式自定义 `minted` 和`\mint` 中的换行。要自定义换行的缩进，请参见 `breakindent` 和 `breakautoindent`。要自定义行延续符号，请使用 `breaksymbolleft` 和 `breaksymbolright`。要自定义延续符号与代码之间的间隔，请使用 `breaksymbolsepleft` 和 `breaksymbolsepright`。要自定义为换行符号腾出空间的额外缩进，请使用 `breaksymbolindentleft` 和 `breaksymbolindentright`。由于默认情况下仅使用左侧符号，因此还可以使用别名选项 `breaksymbol`、`breaksymbolsep` 和 `breaksymbolindent` 来修改左侧符号。请注意，这些选项均不适用于 `\mintinline`，因为在内联上下文中它们无关紧要。

An example using these options to customize the `minted` environment is shown below. This uses the `\carriagereturn` symbol from the dingbat package.

以下示例演示了使用这些选项自定义 `minted` 环境的方法。它使用了 dingbat 宏包的`\carriagereturn` 符号。

```
\begin{minted}[breaklines,
               breakautoindent=false,
               breaksymbolleft=\raisebox{0.8ex}{
               \small\reflectbox{\carriagereturn}},
               breaksymbolindentleft=0pt,
               breaksymbolsepleft=0pt,
               breaksymbolright=\small\carriagereturn,
               breaksymbolindentright=0pt,
               breaksymbolsepright=0pt]{python}
def f(x):
    return 'Some text ' + str(x) + ' some more text ' + str(x) + ' even
    ↪  more text that goes on for a while'
\end{minted}
```

```
def f(x):
    return 'Some text ' + str(x) + ' some more text ' + str(x) + ' even
↪more text that goes on for a while'
```

Automatic line breaks are limited with Pygments styles that use a colored background behind large chunks of text. This coloring is accomplished with `\colorbox`, which cannot break across lines. It may be possible to create an alternative to `\colorbox` that supports line breaks, perhaps with TikZ, but the author is unaware of a satisfactory solution. The only current alternative is to redefine `\colorbox` so that it does nothing. For example,

```
\AtBeginEnvironment{minted}{\renewcommand{\colorbox}[3][]{#3}}
```

uses the etoolbox package to redefine `\colorbox` within all `minted` environments.

Automatic line breaks will not work with `showspaces=true` unless you use `breakanywhere` or `breakafter=\space`.

具有大块文本背景色的 Pygments 样式对自动换行有限制。这种着色是通过`\colorbox`实现的，而`\colorbox` 不能跨行换行。可能可以创建一个支持换行的`\colorbox` 替代方案，比如使用 TikZ，但作者对此没有满意的解决方案。目前唯一的替代方法是重新定义`\colorbox` 使其不起作用。例如，以下代码使用 etoolbox 宏包在所有 `minted` 环境中重新定义`\colorbox`。

```
\AtBeginEnvironment{minted}{\renewcommand{\colorbox}[3][]{#3}}
```

使用 etoolbox 宏包来重新定义所有 `minted` 环境中的`\colorbox` 命令。

自动换行在 `showspaces=true` 时无法正常工作，除非使用 `breakanywhere` 或 `breakafter=\space` 选项。

```
\begin{minted}[breaklines,
               breakautoindent=false,
               breaksymbolleft=\raisebox{0.8ex}{
               \small\reflectbox{\carriagereturn}},
               breaksymbolindentleft=0pt,
               breaksymbolsepleft=0pt,
               breaksymbolright=\small\carriagereturn,
               breaksymbolindentright=0pt,
               breaksymbolsepright=0pt]{python}
def f(x):
    return 'Some text ' + str(x) + ' some more text ' +
    ↪  str(x) + ' even more text that goes on for a
    ↪  while'
\end{minted}
```

```
def f(x):
    return 'Some text ' + str(x) + ' some more text ' +
↪str(x) + ' even more text that goes on for a while'
```

`breaksymbol` (string) (default: `breaksymbolleft`)

Alias for `breaksymbolleft`.

`breaksymbol` (string) (default: `breaksymbolleft`)

是 `breaksymbolleft` 的别名。

breaksymbolleft (string) (default: \tiny\ensuremath{\hookrightarrow}, ↪)
The symbol used at the beginning (left) of continuation lines when breaklines=true. To have no symbol, simply set breaksymbolleft to an empty string ("=," or "={}"). The symbol is wrapped within curly braces {} when used, so there is no danger of formatting commands such as \tiny "escaping."

The \hookrightarrow and \hookleftarrow may be further customized by the use of the \rotatebox command provided by graphicx. Additional arrow-type symbols that may be useful are available in the dingbat (\carriagereturn) and mnsymbol (hook and curve arrows) packages, among others.

Does not apply to \mintinline.

breaksymbolleft (string) (default: \tiny\ensuremath{\hookrightarrow}, ↪)
在 breaklines=true 时,用于表示连续行开头的符号。要没有符号,只需将 breaksymbolleft 设置为空字符串 ("=," 或 "={}") 。当使用时,符号将被包裹在花括号 {} 中,因此不会出现格式命令 (如 \tiny) "逃逸" 的危险。

\hookrightarrow 和\hookleftarrow 可以通过使用 graphicx 提供的\rotatebox 命令进行进一步自定义。其他可能有用的箭头类型符号可以在 dingbat (\carriagereturn) 和 mnsymbol (hook 和 curve arrows) 等包中找到,还有其他一些包也提供了这样的符号。

不适用于 \mintinline。

breaksymbolright (string) (default: ⟨*none*⟩)
The symbol used at breaks (right) when breaklines=true. Does not appear at the end of the very last segment of a broken line.

breaksymbolright (string) (default: ⟨*none*⟩)
在 breaklines=true 时,用于表示断行 (右边) 的符号。不会出现在最后一行的最后一个片段的末尾。

breaksymbolindent (dimension) (default: ⟨*breaksymbolindentleftnchars*⟩)
Alias for breaksymbolindentleft.

breaksymbolindent (dimension) (default: ⟨*breaksymbolindentleftnchars*⟩)
别名为 breaksymbolindentleft。

breaksymbolindentnchars (integer) (default: ⟨*breaksymbolindentleftnchars*⟩)
Alias for breaksymbolindentleftnchars.

breaksymbolindentnchars (整数) (default: ⟨*breaksymbolindentleftnchars*⟩)
别名为 breaksymbolindentleftnchars。

breaksymbolindentleft (dimension) (default: ⟨*breaksymbolindentleftnchars*⟩)
The extra left indentation that is provided to make room for breaksymbolleft. This indentation is only applied when there is a breaksymbolleft.

Does not apply to \mintinline.

breaksymbolindentleft (尺寸) (default: ⟨*breaksymbolindentleftnchars*⟩)
提供额外的左缩进以为 breaksymbolleft 腾出空间。只有当存在 breaksymbolleft 时才应用该缩进。

不适用于 \mintinline。

breaksymbolindentleftnchars (integer) (default: 4)
This allows breaksymbolindentleft to be specified as an integer number of characters rather than as a dimension (assumes a fixed-width font).

breaksymbolindentleftnchars (整数) (default: 4)
允许将 breaksymbolindentleft 指定为字符的整数数量,而不是作为尺寸 (假设使用等宽字体)。

breaksymbolindentright (dimension) (default: ⟨*breaksymbolindentrightnchars*⟩)
The extra right indentation that is provided to make room for `breaksymbolright`. This indentation is only applied when there is a `breaksymbolright`.

(尺寸) (default: ⟨*breaksymbolindentrightnchars*⟩) breaksymbolindentright
提供额外的右缩进以为 `breaksymbolright` 腾出空间。只有当存在 `breaksymbolright` 时才应用该缩进。

breaksymbolindentrightnchars (integer) (default: 4)
This allows `breaksymbolindentright` to be specified as an integer number of characters rather than as a dimension (assumes a fixed-width font).

(整数) (default: 4) breaksymbolindentrightnchars
允许将 `breaksymbolindentright` 指定为字符的整数数量，而不是作为尺寸（假设使用等宽字体）。

breaksymbolsep (dimension) (default: ⟨*breaksymbolsepleftnchars*⟩)
Alias for `breaksymbolsepleft`.

(尺寸) (default: ⟨*breaksymbolsepleftnchars*⟩) breaksymbolsep
别名为 `breaksymbolsepleft`。

breaksymbolsepnchars (integer) (default: ⟨*breaksymbolsepleftnchars*⟩)
Alias for `breaksymbolsepleftnchars`.

(整数) (default: ⟨*breaksymbolsepleftnchars*⟩) breaksymbolsepnchars
别名为 `breaksymbolsepleftnchars`。

breaksymbolsepleft (dimension) (default: ⟨*breaksymbolsepleftnchars*⟩)
The separation between the `breaksymbolleft` and the adjacent text.

(尺寸) (default: ⟨*breaksymbolsepleftnchars*⟩) breaksymbolsepleft
`breaksymbolleft` 与相邻文本之间的间距。

breaksymbolsepleftnchars (integer) (default: 2)
Allows `breaksymbolsepleft` to be specified as an integer number of characters rather than as a dimension (assumes a fixed-width font).

(整数) (default: 2) breaksymbolsepleftnchars
允许将 `breaksymbolsepleft` 指定为字符的整数数量，而不是作为尺寸（假设使用等宽字体）。

breaksymbolsepright (dimension) (default: ⟨*breaksymbolseprightnchars*⟩)
The *minimum* separation between the `breaksymbolright` and the adjacent text. This is the separation between `breaksymbolright` and the furthest extent to which adjacent text could reach. In practice, `\linewidth` will typically not be an exact integer multiple of the character width (assuming a fixed-width font), so the actual separation between the `breaksymbolright` and adjacent text will generally be larger than `breaksymbolsepright`. This ensures that break symbols have the same spacing from the margins on both left and right. If the same spacing from text is desired instead, `breaksymbolsepright` may be adjusted. (See the definition of `\FV@makeLineNumber` in fvextra for implementation details.)

(尺寸) (default: ⟨*breaksymbolseprightnchars*⟩) breaksymbolsepright
`breaksymbolright` 与相邻文本之间的最小间距。这是 `breaksymbolright` 与相邻文本可能达到的最远范围之间的间距。在实践中，`\linewidth` 通常不是字符宽度的精确整数倍（假设使用等宽字体），因此 `breaksymbolright` 与相邻文本的实际间距通常大于 `breaksymbolsepright`。这确保了断行符号与左右两侧边距的间距相同。如果希望与文本具有相同的间距，可以调整 `breaksymbolsepright`。（有关实现细节，请参见 fvextra 中的 `\FV@makeLineNumber` 的定义。）

breaksymbolseprightnchars (integer) (default: 2)
Allows `breaksymbolsepright` to be specified as an integer number of characters rather than as a

(整数) (default: 2) breaksymbolseprightnchars
允许将 `breaksymbolsepright` 指定为字符的整数数量，而不是作为尺寸（假设使

dimension (assumes a fixed-width font).

用等宽字体)。

bgcolor (string) (default: ⟨*none*⟩)

Background color of the listing. Be aware that this option has several limitations (described below); see "Framing alternatives" below for more powerful alternatives.

bgcolor (string) (default: ⟨ 无⟩)

列表的背景颜色。请注意，此选项有一些限制 (下面描述)；有关更强大的替代方法，请参见下面的 "其他框架" 部分。

The value of this option must *not* be a color command. Instead, it must be a color *name*, given as a string, of a previously-defined color:

此选项的值不能是颜色命令。而是必须是先前定义的颜色的颜色名称的字符串：

```
\definecolor{bg}{rgb}{0.95,0.95,0.95}
\begin{minted}[bgcolor=bg]{php}
<?php
    echo "Hello, $x";
?>
\end{minted}
```

```
<?php
    echo "Hello, $x";
?>
```

```
\definecolor{bg}{rgb}{0.9,0.8,0.7}
\begin{minted}[bgcolor=bg]{php}
<?php
    echo "Hello, $x";
?>
\end{minted}
```

```
<?php
    echo "Hello, $x";
?>
```

This option puts `minted` environments and `\mint` commands in a `snugshade*` environment from the framed package, which supports breaks across pages. (Prior to minted 2.2, a `minipage` was used, which prevented page breaks and gave undesirable spacing from surrounding text.) Be aware that if `bgcolor` is used with `breaklines=true`, and a line break occurs just before a page break, then text may extend below the colored background in some instances. It is best to use a more advanced framing package in those cases; see "Framing alternatives" below.

此选项将 `minted` 环境和`\mint` 命令放在 `framed` 包的 `snugshade*` 环境中，该环境支持跨页断行。(在 `minted` 2.2 之前，使用的是 `minipage`，它会阻止分页，并且给周围文本带来不必要的间距。) 请注意，如果 `bgcolor` 与 `breaklines=true` 一起使用，并且在分页之前发生换行，则在某些情况下，文本可能会延伸到有色背景下方。在这些情况下，最好使用更高级的装裱包；请参见下面的 "其他框架" 部分。

This option puts `\mintinline` inside a `\colorbox`, which **does not allow line breaks**. If you want to use `\setminted` to set background colors, and only want background colors on `minted` and `\mint`, you may use `\setmintedinline{bgcolor={}}` to turn off the coloring for inline commands.

此选项将`\mintinline` 放在`\colorbox` 内部，**不允许换行**。如果要使用`\setminted` 设置背景颜色，并且只想在 `minted` 和`\mint` 上使用背景颜色，则可以关闭内联命令的着色: 使用`\setmintedinline{bgcolor={}}`。

Framing alternatives

其他框架

If you want more reliable and advanced options for background colors and framing, you should consider a more advanced framing package such as mdframed or tcolorbox. It is easy to add framing to minted commands and environments using the etoolbox package, which is automatically loaded by minted. For example, using mdframed:

如果您希望为背景颜色和装饰提供更可靠和先进的选择，应考虑使用更高级的装裱包，例如 `mdframed` 或 `tcolorbox`。使用 `minted` 自动加载的 `etoolbox` 包可以轻松地为 `minted` 命令和环境添加装饰。例如，使用 `mdframed:`

```
\BeforeBeginEnvironment{minted}{\begin{mdframed}}
\AfterEndEnvironment{minted}{\end{mdframed}}
```

Some framing packages also provide built-in commands for such purposes. For example, mdframed provides a `\surroundwithmdframed` command, which could be used to add a frame to all `minted` environments:

```
\surroundwithmdframed{minted}
```

tcolorbox even provides a built-in framing environment with minted support. Simply use `\tcbuselibrary{minted}` in the preamble, and then put code within a `tcblisting` environment:

```
\begin{tcblisting}{<tcb options>,
                   minted language=<language>,
                   minted style=<style>,
                   minted options={<option list>} }
<code>
\end{tcblisting}
```

tcolorbox provides other commands and environments for fine-tuning listing appearance and for working with external code files.

```
\BeforeBeginEnvironment{minted}{\begin{mdframed}}
\AfterEndEnvironment{minted}{\end{mdframed}}
```

某些装裱包还提供了用于此类目的的内置命令。例如，`mdframed` 提供了一个可用于为所有的 `minted` 环境添加框架的 `\surroundwithmdframed` 命令：

```
\surroundwithmdframed{minted}
```

`tcolorbox` 甚至提供了一个带有 `minted` 支持的内置装裱环境。只需在导言区使用`\tcbuselibrary{minted}`，然后将代码放在 `tcblisting` 环境中：

```
\begin{tcblisting}{<tcb options>,
                   minted language=<language>,
                   minted style=<style>,
                   minted options={<option list>} }
<code>
\end{tcblisting}
```

`tcolorbox` 还提供其他用于调整列表外观和处理外部代码文件的命令和环境。

codetagify (list of strings) (default: highlight `XXX`, `TODO`, `BUG`, and `NOTE`)
Highlight special code tags in comments and docstrings.

codetagify (字符串列表) (default: highlight `XXX`、`TODO`、`BUG` 和 `NOTE`)
高亮代码中的特殊代码标记

curlyquotes (boolean) (default: `false`)
By default, the backtick ` and typewriter single quotation mark ' always appear literally, instead of becoming the left and right curly single quotation marks ‘’. This option allows these characters to be replaced by the curly quotation marks when that is desirable.

curlyquotes (boolean) (default: `false`)
默认情况下，反引号（`）和打字机样式的单引号（'）将直接显示，而不会变成左右花括号样式的单引号（‘’）。当需要时，可以使用该选项将这些字符替换为花括号样式的单引号。

encoding (string) (default: ⟨*system-specific*⟩)
Sets the file encoding that Pygments expects. See also `outencoding`.

encoding (字符串) (default: ⟨ *系统特定*⟩)
设置 Pygments 预期的文件编码。参见`outencoding`。

escapeinside (string) (default: ⟨*none*⟩)

Escape to LaTeX between the two characters specified in `(string)`. All code between the two characters will be interpreted as LaTeX and typeset accordingly. This allows for additional formatting. The escape characters need not be identical. Special LaTeX characters must be escaped when they are used as the escape characters (for example, `escapeinside=\#\%`). Requires Pygments 2.0+.

Escaping does not work inside strings and comments (for comments, there is `texcomments`). As of Pygments 2.0.2, this means that escaping is "fragile" with some lexers. Due to the way that Pygments implements `escapeinside`, any "escaped" LaTeX code that resembles a string or comment for the current lexer may break `escapeinside`. There is a Pygments issue for this case. Additional details and a limited workaround for some scenarios are available on the minted GitHub site.

```
\begin{minted}[escapeinside=||]{py}
def f(x):
    y = x|\colorbox{green}{**}|2
    return y
\end{minted}
```

```
def f(x):
    y = x ** 2
    return y
```

Note that when math is used inside escapes, any active characters beyond those that are normally active in verbatim can cause problems. Any package that relies on special active characters in math mode (for example, icomma) will produce errors along the lines of `TeX capacity exceeded` and `\leavevmode\kern\z@`. This may be fixed by modifying `\@noligs`, as described at http://tex.stackexchange.com/questions/223876.

escapeinside (字符串) (default: ⟨ 无⟩)

在指定的两个字符之间转义为 LaTeX。两个字符之间的所有代码将被解释为 LaTeX 并相应地排版。这允许进行额外的格式设置。逃避字符无需相同。当使用它们作为转义字符时，必须对特殊的 LaTeX 字符进行转义（例如，`escapeinside=\#\%`）。需要 Pygments 2.0+。

转义在字符串和注释中不起作用（对于注释，有 `texcomments`）。截至 Pygments 2.0.2，这意味着转义在某些词法分析器中是"脆弱"的。由于 Pygments 实现了 `escapeinside`，所以类似字符串或注释的"转义" LaTeX 代码可能会破坏 `escapeinside`。有一个关于此情况的Pygments 问题。关于一些场景的更多详细信息和有限的解决方案也可在minted GitHub 站点上找到。

```
\begin{minted}[escapeinside=||]{py}
def f(x):
    y = x|\colorbox{red}{**}|2
    return y
\end{minted}
```

```
def f(x):
    y = x ** 2
    return y
```

请注意，当在转义中使用数学时，除了在通常在抄录中活动的字符外，任何其他活动字符都可能导致问题。任何依赖于数学模式中特殊活动字符的包（例如 icomma）都会产生错误，例如 `TeX capacity exceeded` 和`\leavevmode\kern\z@`。通过修改`\@noligs`来解决此问题，如http://tex.stackexchange.com/questions/223876所述。

fontfamily (family name) (default: `tt`)

The font family to use. `tt`, `courier` and `helvetica` are pre-defined.

fontfamily (字体系列名称) (default: `tt`)

要使用的字体系列。`tt`、`courier`和`helvetica`是预定义的。

fontseries (series name) (default: `auto` – the same as the current font)

The font series to use.

fontseries (字体系列名称) (default: `auto` – 与当前字体相同)

要使用的字体系列。

fontsize (font size) (default: `auto` – the same as the current font)

The size of the font to use, as a size command, e.g. `\footnotesize`.

fontsize (字体大小) (default: `auto` – 与当前字体相同)

要使用的字体大小，作为大小命令，例如`\footnotesize`。

`fontshape` (font shape) (default: `auto` – the same as the current font)
The font shape to use.

(**字体形状**) (default: `auto` – 与当前字体相同) `fontshape`
要使用的字体形状。

`formatcom` (command) (default: ⟨*none*⟩)
A format to execute before printing verbatim text.

(**命令**) (default: ⟨ *无*⟩) `formatcom`
在打印等宽文本之前执行的格式化命令。

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

(**整数**) (default: 1) `firstline`
要显示的第一行。在此行之前的所有行将被忽略，不会出现在输出中。

`lastline` (integer) (default: ⟨*last line of input*⟩)
The last line to be shown.

(**整数**) (default: ⟨ *输入的最后一行*⟩) `lastline`
要显示的最后一行。

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

(auto | last | **整数**) (default: `auto` = 1) `firstnumber`
第一行的行号。

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

(**整数**) (default: 1) `stepnumber`
出现行号的间隔。

`stepnumberfromfirst` (boolean) (default: `false`)
By default, when line numbering is used with `stepnumber` $\neq 1$, only line numbers that are a multiple of `stepnumber` are included. This offsets the line numbering from the first line, so that the first line, and all lines separated from it by a multiple of `stepnumber`, are numbered.

(**布尔值**) (default: `false`) `stepnumberfromfirst`
默认情况下，当使用`stepnumber` $\neq 1$ 进行行编号时，只包括`stepnumber`的倍数的行号。这会使行编号从第一行偏移，以便第一行和其后以`stepnumber`的倍数分隔的所有行都被编号。

`stepnumberoffsetvalues` (boolean) (default: `false`)
By default, when line numbering is used with `stepnumber` $\neq 1$, only line numbers that are a multiple of `stepnumber` are included. Using `firstnumber` to offset the numbering will change which lines are numbered and which line gets which number, but will not change which *numbers* appear. This option causes `firstnumber` to be ignored in determining which line numbers are a multiple of `stepnumber`. `firstnumber` is still used in calculating the actual numbers that appear. As a result, the line numbers that appear will be a multiple of `stepnumber`, plus `firstnumber` minus 1.

(**布尔值**) (default: `false`) `stepnumberoffsetvalues`
默认情况下，当使用`stepnumber` $\neq 1$ 进行行编号时，只包括`stepnumber`的倍数的行号。使用`firstnumber`进行偏移将改变编号的行和行获得的编号，但不会改变出现的编号。此选项导致在确定哪些行号是`stepnumber`的倍数时忽略`firstnumber`。仍然使用`firstnumber`来计算实际出现的数字。结果是，出现的行号将是`stepnumber`的倍数，加上`firstnumber`减 1。

numberfirstline (boolean) (default: false)
Always number the first line, regardless of stepnumber.

numberfirstline (布尔值) (default: false)
总是对第一行进行编号，而不管stepnumber如何。

linenos (boolean) (default: false)
Enables line numbers. In order to customize the display style of line numbers, you need to redefine the \theFancyVerbLine macro:

linenos (布尔值) (default: false)
启用行号。要自定义行号的显示样式，需要重定义\theFancyVerbLine宏：

```
\renewcommand{\theFancyVerbLine}{\sffamily
    \textcolor[rgb]{0.5,0.5,1.0}{\scriptsize
    \oldstylenums{\arabic{FancyVerbLine}}}}

\begin{minted}[linenos,
    firstnumber=11]{python}
def all(iterable):
    for i in iterable:
        if not i:
            return False
    return True
\end{minted}
```

```
11  def all(iterable):
12      for i in iterable:
13          if not i:
14              return False
15      return True
```

numbers (left | right | both | none) (default: none)
Essentially the same as linenos, except the side on which the numbers appear may be specified.

numbers (left | right | both | none) (default: none)
与linenos基本相同，只是可以指定数字显示的位置。

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

numberblanklines (布尔值) (default: true)
启用或禁用空行的编号。

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

numbersep (尺寸) (default: 12pt)
数字和行的起始位置之间的间距。

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

frame (none | leftline | topline | bottomline | lines | single) (default: none)
围绕源代码列表放置的边框的类型。

framerule (dimension) (default: 0.4pt)
Width of the frame.
(尺寸) (default: 0.4pt) framerule
边框的宽度。

framesep (dimension) (default: \fboxsep)
Distance between frame and content.
(尺寸) (default: \fboxsep) framesep
边框和内容之间的距离。

rulecolor (color command) (default: black)
The color of the frame.
(颜色命令) (default: black) rulecolor
边框的颜色。

resetmargins (boolean) (default: false)
Resets the left margin inside other environments.
(布尔值) (default: false) resetmargins
在其他环境中重置左边距。

xleftmargin (dimension) (default: 0)
Indentation to add before the listing.
(尺寸) (default: 0) xleftmargin
列表之前要添加的缩进。

xrightmargin (dimension) (default: 0)
Indentation to add after the listing.
(尺寸) (default: 0) xrightmargin
列表之后要添加的缩进。

obeytabs (boolean) (default: false)
Treat tabs as tabs instead of converting them to spaces—that is, expand them to tab stops determined by tabsize. **While this will correctly expand tabs within leading indentation, usually it will not correctly expand tabs that are preceded by anything other than spaces or other tabs. It should be avoided in those case.**
(布尔值) (default: false) obeytabs
将制表符视为制表符，而不是将其转换为空格-即，将其扩展为由tabsize确定的制表位。**虽然这样可以正确扩展前导缩进中的制表符，但通常情况下，它不能正确扩展除空格或其他制表符之外的任何内容之前的制表符。在这些情况下，应避免使用。**

showspaces (boolean) (default: false)
Enables visible spaces: visible␣spaces.
(布尔值) (default: false) showspaces
启用可见空格：可见空格。

showtabs (boolean) (default: false)
Enables visible tabs—only works in combination with obeytabs.
(布尔值) (default: false) showtabs
启用可见制表符-仅在与obeytabs组合使用时有效。

space (macro) (default: \textvisiblespace, ␣)
Redefine the visible space character. Note that this is only used if showspaces=true.

space (宏) (default: \textvisiblespace, ␣)
重新定义可见空格字符。请注意，只有在showspaces=true时才会使用它。

spacecolor (string) (default: none)
Set the color of visible spaces. By default (none), they take the color of their surroundings.

spacecolor (字符串) (default: none)
设置可见空格的颜色。默认情况下（ none ），它们采用其周围的颜色。

tab (宏) (default: fancyvrb 的 \FancyVerbTab, ↦)
重新定义可见制表符字符。请注意，只有在showtabs=true时才会使用它。\rightarrowfill，⟶，可能是一个不错的选择。

tab (macro) (default: fancyvrb's \FancyVerbTab, ↦)
Redefine the visible tab character. Note that this is only used if showtabs=true. \rightarrowfill, ⟶, may be a nice alternative.

tabcolor (string) (default: black)
Set the color of visible tabs. If tabcolor=none, tabs take the color of their surroundings. This is typically undesirable for tabs that indent multiline comments or strings.

tabcolor (字符串) (default: black)
设置可见制表符的颜色。如果tabcolor=none，制表符将采用其周围的颜色。这通常对于缩进多行注释或字符串的制表符是不理想的。

tabsize (integer) (default: 8)
The number of spaces a tab is equivalent to. If obeytabs is *not* active, tabs will be converted into this number of spaces. If obeytabs is active, tab stops will be set this number of space characters apart.

tabsize (整数) (default: 8)
制表符相当于的空格数。如果未启用obeytabs，则将制表符转换为此数量的空格。如果启用了obeytabs，则制表位将与此数量的空格字符之间设置制表位。

stripall (boolean) (default: false)
Strip all leading and trailing whitespace from the input.

stripall (布尔值) (default: false)
从输入的每行中删除所有前导和尾随空格。

stripnl (boolean) (default: false)
Strip leading and trailing newlines from the input.

stripnl (布尔值) (default: false)
从输入的每行中删除前导和尾随换行符。

autogobble (boolean) (default: false)
Remove (gobble) all common leading whitespace from code. Essentially a version of gobble that automatically determines what should be removed. Good for code that originally is not indented, but is manually indented after being pasted into a LaTeX document.

autogobble (布尔值) (default: false)
从代码中删除 (gobble) 所有通用的前导空白字符。实际上是自动确定哪些空白字符应该被删除的版本。适用于原始没有缩进的代码，但在粘贴到 LaTeX 文档中后手动缩进的情况。

```
...text.
\begin{minted}[autogobble]{python}
    def f(x):
        return x**2
\end{minted}
```

```
...text.
def f(x):
    return x**2
```

```
...text.
\begin{minted}[autogobble]{python}
    def f(x):
        return x**2
\end{minted}
```

```
...text.
def f(x):
    return x**2
```

gobble (integer) (default: 0)

Remove the first n characters from each input line.

gobble (整数) (default: 0)

从每行中删除前 n 个字符。

mathescape (boolean) (default: `false`)

Enable LaTeX math mode inside comments. Usage as in package listings. See the note under `escapeinside` regarding math and ligatures.

mathescape (布尔值) (default: `false`)

在注释中启用 LaTeX 数学模式。与 listings 包中的用法相同。请参阅有关数学和连字符的注释的说明。

texcl (boolean) (default: `false`)

Enables LaTeX code inside comments. Usage as in package listings. See the note under `escapeinside` regarding math and ligatures.

texcl (布尔值) (default: `false`)

在注释中启用 LaTeX 代码。与 listings 包中的用法相同。请参阅有关数学和连字符的注释的说明。

texcomments (boolean) (default: `false`)

Enables LaTeX code inside comments. The newer name for `texcl`. See the note under `escapeinside` regarding math and ligatures.

As of Pygments 2.0.2, `texcomments` fails with multiline C/C++ preprocessor directives, and may fail in some other circumstances. This is because preprocessor directives are tokenized as `Comment.Preproc`, so `texcomments` causes preprocessor directives to be treated as literal LaTeX code. An issue has been opened at the Pygments site; additional details are also available on the minted GitHub site.

texcomments (布尔值) (default: `false`)

在注释中启用 LaTeX 代码。这是`texcl`的新名称。有关数学和连字符的注释的注意事项，请参见`escapeinside`。

从 Pygments 2.0.2 开始，`texcomments` 在多行 C/C++ 预处理器指令中失败，并且在某些其他情况下可能会失败。这是因为预处理器指令被令牌化为 `Comment.Preproc`，因此 `texcomments` 会导致将预处理器指令视为字面 LaTeX 代码。在 Pygments 站点上已经打开了一个问题；关于此问题还可以在minted GitHub 站点上找到其他详细信息。

funcnamehighlighting (boolean) (default: `true`)

[For PHP only] If `true`, highlights built-in function names.

funcnamehighlighting (布尔值) (default: `true`)

[仅适用于 PHP] 如果为 true，则突出显示内置函数名称。

startinline (boolean) (default: `false`)

startinline (布尔值) (default: `false`)

[For PHP only] Specifies that the code starts in PHP mode, i.e., leading `<?php` is omitted.

[仅适用于 PHP] 指定代码以 PHP 模式开始，即省略前导的`<?php`。

highlightcolor

(string) (default: `LightCyan`)

Set the color used for `highlightlines`, using a predefined color name from color or xcolor, or a color defined via `\definecolor`.

(字符串) (default: `LightCyan`)

设置用于`highlightlines`的颜色，使用 color 或 xcolor 中预定义的颜色名称，或通过`\definecolor`定义的颜色。

highlightlines

(string) (default: ⟨*none*⟩)

This highlights a single line or a range of lines based on line numbers. For example, `highlightlines={1, 3-4}`. The line numbers refer to the line numbers that would appear if `linenos=true`, etc. They do not refer to original or actual line numbers before adjustment by `firstnumber`.

The highlighting color can be customized with `highlightcolor`.

(字符串) (default: ⟨ *无*⟩)

基于行号高亮单个行或行范围。例如，`highlightlines={1, 3-4}`。行号指的是如果`linenos=true`，等等，将出现的行号。它们不是指调整之前的原始或实际行号。

可以使用`highlightcolor`自定义高亮颜色。

keywordcase

(string) (default: `lower`)

Changes capitalization of keywords. Takes `lower`, `upper`, or `capitalize`.

(字符串) (default: `lower`)

更改关键字的大小写。接受`lower`、`upper`或`capitalize`。

label

(string) (default: *empty*)

Add a label to the top, the bottom or both of the frames around the code. See the fancyvrb documentation for more information and examples. *Note:* This does *not* add a `\label` to the current listing. To achieve that, use a floating environment (section **??**) instead.

(字符串) (default: 空)

在代码周围的框的顶部、底部或两者之间添加标签。有关更多信息和示例，请参见 fancyvrb 文档。请注意，这不会向当前列表添加`\label`。要实现这一点，请使用浮动环境（第**??**节）。

labelposition

(none | topline | bottomline | all) (default: `topline`, `all`, or `none`)

Position where to print the label (see above; default: `topline` if one label is defined, `all` if two are defined, `none` else). See the fancyvrb documentation for more information.

(none | topline | bottomline | all) (default: `topline`, `all`, or `none`)

打印标签的位置（见上文；默认为如果定义了一个标签，则为`topline`，如果定义了两个标签，则为`all`，否则为`none`）。有关详细信息，请参见 fancyvrb 文档。

outencoding

(字符串) (default: ⟨ *系统特定*⟩)

设置 Pygments 用于高亮输出的文件编码。覆盖之前通过`encoding`设置的任何编码。

(string) (default: ⟨*system-specific*⟩)

Sets the file encoding that Pygments uses for highlighted output. Overrides any encoding previously set via `encoding`.

python3

(boolean) (default: `false`)

[For PythonConsoleLexer only] Specifies whether Python 3 highlighting is applied.

(布尔值) (default: `false`)

[For PythonConsoleLexer only] 指定是否应用 Python 3 的语法高亮。

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

(布尔值) (default: `false`) samepage
强制整个列表出现在同一页上，即使它不适合。

style (string) (default: ⟨*default*⟩)
Sets the stylesheet used by Pygments.

(字符串) (default: ⟨ 默认⟩) style
设置 Pygments 使用的样式表。

6 Defining shortcuts

Large documents with a lot of listings will nonetheless use the same source language and the same set of options for most listings. Always specifying all options is redundant, a lot to type and makes performing changes hard.

One option is to use `\setminted`, but even then you must still specify the language each time.

minted therefore defines a set of commands that lets you define shortcuts for the highlighting commands. Each shortcut is specific for one programming language.

\newminted `\newminted` defines a new alias for the `minted` environment:

```
\newminted{cpp}{gobble=2,linenos}

\begin{cppcode}
    template <typename T>
    T id(T value) {
        return value;
    }
\end{cppcode}
```

```
1  template <typename T>
2  T id(T value) {
3      return value;
4  }
```

If you want to provide extra options on the fly, or override existing default options, you can do that, too:

```
\newminted{cpp}{gobble=2,linenos}

\begin{cppcode*}{linenos=false,
                 frame=single}
    int const answer = 42;
\end{cppcode*}
```

```
int const answer = 42;
```

6 定义快捷方式

大型文档中有许多源码清单，但大多数源码清单使用相同的源语言和相同的选项。始终指定所有选项是多余的，输入的内容很多且更改困难。

一个选项是使用`\setminted`，但是您仍然需要每次都指定语言。

minted 因此定义了一组命令，可让您为高亮命令定义快捷方式。每个快捷方式特定于一种编程语言。

\newminted `\newminted`定义了一个新的别名，用于`minted`环境：

如果您要在使用中即时提供额外的选项，或覆盖现有的默认选项，也可以这样做：

Notice the star "*" behind the environment name—due to restrictions in fancyvrb's handling of options, it is necessary to provide a *separate* environment that accepts options, and the options are *not* optional on the starred version of the environment.

注意环境名后的星号 "*" ——由于 fancyvrb 对选项处理的限制，有必要提供一个接受选项的*单独*环境，而且选项在环境的星号版本上*不是*可选的。

The default name of the environment is ⟨*language*⟩code. If this name clashes with another environment or if you want to choose an own name for another reason, you may do so by specifying it as the first argument: \newminted[⟨*environment name*⟩]{⟨*language*⟩}{⟨*options*⟩}.

环境的默认名称是⟨*language*⟩code。如果此名称与其他环境冲突，或者出于其他原因希望选择自己的名称，可以将其作为第一个参数指定：\newminted[⟨ *环境名称*⟩]{⟨ *语言*⟩}{⟨ *选项*⟩}。

Like normal minted environments, environments created with \newminted may be used within other environment definitions. Since the minted environments use fancyvrb internally, any environment based on them must include the fancyvrb command \VerbatimEnvironment. This allows fancyvrb to determine the name of the environment that is being defined, and correctly find its end. It is best to include this command at the beginning of the definition. For example,

与普通的 minted 环境一样，使用\newminted创建的环境可以在其他环境定义中使用。由于 minted 使用 fancyvrb 内部实现的环境，因此基于它们的任何环境都必须包含 fancyvrb 命令\VerbatimEnvironment。这允许 fancyvrb 确定正在定义的环境的名称，并正确找到其结束。最好在定义的开始处包含此命令。例如，

```
\newminted{cpp}{gobble=2,linenos}
\newenvironment{env}{\VerbatimEnvironment\begin{cppcode}}{\end{cppcode}}
```

```
\newminted{cpp}{gobble=2,linenos}
\newenvironment{env}%
{\VerbatimEnvironment\begin{cppcode}}{\end{cppcode}}
```

\newmint The above macro only defines shortcuts for the minted environment. The main reason is that the short command form \mint often needs different options—at the very least, it will generally not use the gobble option. A shortcut for \mint is defined using \newmint[⟨*macro name*⟩]{⟨*language*⟩}{⟨*options*⟩}. The arguments and usage are identical to \newminted. If no ⟨*macro name*⟩ is specified, ⟨*language*⟩ is used.

上述宏仅为minted环境定义了快捷方式。主要原因是简短的命令形式\mint通常需要不同的选项-至少通常不会使用gobble选项。使用\newmint[⟨ *宏名称*⟩]{⟨ *语言*⟩}{⟨ *选项*⟩} 定义\mint的快捷方式。参数和用法与\newminted相同。如果未指定⟨ *宏名称*⟩，则使用⟨ *语言*⟩。 \newmint

```
\newmint{perl}{showspaces}

\perl/my $foo = $bar;/
```

```
my␣$foo␣=␣$bar;
```

\newmintinline This creates custom versions of \mintinline. The syntax is the same as that for \newmint: \newmintinline[⟨*macro name*⟩]{⟨*language*⟩}{⟨*options*⟩}. If a ⟨*macro name*⟩ is not specified, then the created macro is called \⟨*language*⟩inline.

这将创建 \mintinline 的自定义版本。语法与\newmint 相同:\newmintinline[⟨ *宏名称*⟩]{⟨ *语言*⟩}{⟨ *选项*⟩}。如果未指定⟨ *宏名称*⟩，则创建的宏将被称为\⟨ *语言*⟩inline。 \newmintinline

```
\newmintinline{perl}{showspaces}

X\perlinline/my $foo = $bar;/X
```

```
Xmy␣$foo␣=␣$bar;X
```

\newmintedfile This creates custom versions of \inputminted. The syntax is

这将创建 \inputminted 的自定义版本。语法为 \newmintedfile

`\newmintedfile[`⟨*macro name*⟩`]{`⟨*language*⟩`}{`⟨*options*⟩`}`

If no ⟨*macro name*⟩ is given, then the macro is called `\`⟨*language*⟩`file`.

7 FAQ and Troubleshooting

In some cases, minted may not give the desired result due to other document settings that it cannot control. Common issues are described below, with workarounds or solutions. You may also wish to search tex.stackexchange.com or ask a question there, if you are working with minted in a non-typical context.

- **There are intermittent "I can't write on file" errors.** This can be caused by using minted in a directory that is synchronized with Dropbox or a similar file syncing program. These programs can try to sync minted's temporary files while it still needs to be able to modify them. The solution is to turn off file syncing or use a directory that is not synced.

- **I receive a "Font Warning: Some font shapes were not available" message, or bold or italic seem to be missing.** This is due to a limitation in the font that is currently in use for typesetting code. In some cases, the default font shapes that LaTeX substitutes are perfectly adequate, and the warning may be ignored. In other cases, the font substitutions may not clearly indicate bold or italic text, and you will want to switch to a different font. See The LaTeX Font Catalogue's section on Typewriter Fonts for alternatives. If you like the default LaTeX fonts, the lmodern package is a good place to start. The beramono and courier packages may also be good options.

- **I receive a "Too many open files" error under OS X when using caching.** See the note on OS X under Section **??**.

- **TeXShop can't find `pygmentize`.** You may need to create a symlink. See `https://tex.stackexchange.com/questions/279214`.

- **Weird things happen when I use the fancybox package.** fancybox conflicts with fancyvrb, which minted uses internally. When using fancybox, make sure that it is loaded before minted

`\newmintedfile[`⟨*宏名称*⟩`]{`⟨*语言*⟩`}{`⟨*选项*⟩`}`

如果未指定⟨*宏名称*⟩，则将使用`\`⟨*语言*⟩`file`作为宏名称。

7 常见问题和故障排除

在某些情况下，由于其他文档设置的原因，minted 可能无法给出所需的结果。下面描述了一些常见的问题，并提供了解决方法或解决方案。如果您在非典型环境中使用 minted，您可能希望搜索tex.stackexchange.com或在那里提问。

- **出现间歇性的"无法写入文件"错误。**这可能是因为在与 Dropbox 或类似的文件同步程序同步的目录中使用 minted。这些程序可能在 minted 仍需要修改临时文件时尝试同步 minted 的临时文件。解决方法是关闭文件同步或使用一个不与之同步的目录。

- **我收到"字体警告：某些字体形状不可用"的消息，或者粗体或斜体似乎丢失了。**这是由于当前用于排版代码的字体的限制。在某些情况下，LaTeX 默认的字体形状替代是完全合适的，可以忽略警告。在其他情况下，字体替代可能无法明确指示粗体或斜体文本，您可能需要切换到其他字体。有关备选方案，请参考 LaTeX 字体目录中的Typewriter Fonts部分。如果您喜欢默认的 LaTeX 字体，可以尝试使用 lmodern 宏包。此外，beramono 和 courier 宏包也是不错的选择。

- **我遇到了"Too many open files"的错误。**这是由于使用缓存时的问题。解决方法是在 XeLaTeX 命令行中添加`-output-driver="xdvipdfmx -8bit"`选项，这会将文件保存到临时文件夹中。

- **TexShop 无法找到 `pygmentize`。**您可能需要创建一个符号链接。请参见`https://tex.stackexchange.com/questions/279214`。

- **当我使用 minted 和 fancybox 宏包时会出现奇怪的问题。**这是因为 fancybox 和 minted 之间存在冲突，而 minted 在内部使用 fancyvrb。使用 minted 之前

(or before fancyvrb, if fancyvrb is not loaded by minted).

- **When I use minted with KOMA-Script document classes, I get warnings about `\float@addtolists`.** minted uses the float package to produce floated listings, but this conflicts with the way KOMA-Script does floats. Load the package scrhack to resolve the conflict. Or use minted's `newfloat` package option.

- **Tilde characters ~ are raised, almost like superscripts.** This is a font issue. You need a different font encoding, possibly with a different font. Try `\usepackage[T1]{fontenc}`, perhaps with `\usepackage{lmodern}`, or something similar.

- **I'm getting errors with math, something like `TeX capacity exceeded` and `\leavevmode\kern\z@`.** This is due to ligatures being disabled within verbatim content. See the note under `escapeinside`.

- **With `mathescape` and the breqn package (or another special math package), the document never finishes compiling or there are other unexpected results.** Some math packages like breqn give certain characters like the comma special meanings in math mode. These can conflict with minted. In the breqn and comma case, this can be fixed by redefining the comma within `minted` environments:

  ```
  \AtBeginEnvironment{minted}{\catcode`\,=12\mathcode`\,="613B}
  ```

  Other packages/special characters may need their own modifications.

- **I'm getting errors with Beamer.** Due to how Beamer treats verbatim content, you may need to use either the `fragile` or `fragile=singleslide` options for frames that contain minted commands and environments. `fragile=singleslide` works best, but it disables overlays. `fragile` works by saving the contents of each frame to a temp file and then reusing them. This approach allows overlays, but will break if you have the string `\end{frame}` at the beginning of a line (for example, in a `minted` environment). To work around that, you can indent the content of the environment (so that the `\end{frame}` is preceded by one or more spaces) and then use the `gobble` or `autogobble` options to remove the indentation.

- **Tabs are eaten by Beamer.** This is due to a bug in Beamer's treatment of verbatim content. Upgrade Beamer or use the linked patch. Otherwise, try `fragile=singleslide` if you don't

加载 fancybox 宏包可以解决此问题，或者将 fancyvrb 在 minted 之前加载。

- **当我在 KOMA-Script 文档类中使用 minted 时，我收到关于`\float@addtolists`的警告。**这是因为 minted 使用 float 宏包来创建浮动的`listing`环境，而这与 KOMA-Script 处理浮动的方式冲突。解决此问题的方法是在导言区加载 scrhack 宏包，以解决冲突。或者，可以使用 minted 的`newfloat`宏包选项。

- **波浪字符 ~ 被抬高，几乎像上标一样。**这是一个字体问题。你需要使用不同的字体编码，可能需要使用不同的字体。尝试使用 `\usepackage[T1]{fontenc}`，或者配合使用 `\usepackage{lmodern}` 或类似的选项。

- **我在数学公式中遇到了类似于 `TeX capacity exceeded` 和 `\leavevmode\kern\z@` 的错误。**这是因为在抄录内容中禁用了连字。请参考 `escapeinside` 下的注释。

- **使用 breqn 宏包（或其他特殊的数学宏包）和 `mathescape` 时，文档无法编译完成或出现其他意外结果。**一些数学宏包（如 breqn）在数学模式中赋予逗号等字符特殊含义。这可能与 minted 发生冲突。对于 breqn 和逗号的情况，可以通过在 `minted` 环境中重新定义逗号来解决问题：

  ```
  \AtBeginEnvironment{minted}{\catcode`\,=12\mathcode`\,="613B}
  ```

  其他宏包/特殊字符可能需要进行相应的修改。

- **我在使用 Beamer 时遇到了错误。**由于 Beamer 对抄录内容的处理方式，你可能需要在包含 minted 命令和环境的幻灯片中使用 `fragile` 或 `fragile=singleslide` 选项。`fragile=singleslide` 是最佳选择，但它会禁用覆盖效果。`fragile` 通过将每个幻灯片的内容保存到临时文件中并重新使用它们来工作。这种方法允许使用覆盖效果，但如果你在一行的开头有字符串 `\end{frame}`（例如，在 `minted` 环境中），它会出错。为了解决这个问题，你可以缩进环境的内容（使得 `\end{frame}` 的前面有一个或多个空格），然后使用 `gobble` 或 `autogobble` 选项来去除缩进。

- **Beamer 会删除制表符。**这是因为 Beamer 在处理抄录内容时存在一个 错误。如果你不需要覆盖效果，可以尝试使用 `fragile=singleslide`；否则，

need overlays, or consider using `\inputminted` or converting the tabs into spaces.

可以考虑使用 `\inputminted` 或将制表符转换为空格。

- **I'm trying to create several new minted commands/environments, and want them all to have the same settings. I'm saving the settings in a macro and then using the macro when defining the commands/environments. But it's failing.** This is due to the way that keyval works (minted uses it to manage options). Arguments are not expanded. See this and this for more information. It is still possible to do what you want; you just need to expand the options macro before passing it to the commands that create the new commands/environments. An example is shown below. The `\expandafter` is the vital part.

```
\def\args{linenos,frame=single,fontsize=\footnotesize,style=bw}

\newcommand{\makenewmintedfiles}[1]{%
\newmintedfile[inputlatex]{latex}{#1}%
\newmintedfile[inputc]{c}{#1}%
}

\expandafter\makenewmintedfiles\expandafter{\args}
```

- **我想创建几个新的 minted 命令/环境，并希望它们都具有相同的设置。我将设置保存在一个宏中，然后在定义命令/环境时使用该宏，但失败了。**这是因为 keyval 的工作方式（minted 使用它来管理选项）中的一个问题。参数没有被展开。请参考 这个 和 这个 获取更多信息。你仍然可以实现你想要的效果；你只需要在将选项宏传递给创建新命令/环境的命令之前展开它。以下是一个示例，其中 `\expandafter` 是关键部分。

```
\def\args{linenos,frame=single,fontsize=\footnotesize,style=bw}

\newcommand{\makenewmintedfiles}[1]{%
\newmintedfile[inputlatex]{latex}{#1}%
\newmintedfile[inputc]{c}{#1}%
}

\expandafter\makenewmintedfiles\expandafter{\args}
```

- **I want to use `\mintinline` in a context that normally doesn't allow verbatim content.** The `\mintinline` command will already work in many places that do not allow normal verbatim commands like `\verb`, so make sure to try it first. If it doesn't work, one of the simplest alternatives is to save your code in a box, and then use it later. For example,

```
\newsavebox\mybox
\begin{lrbox}{\mybox}
\mintinline{cpp}{std::cout}
\end{lrbox}

\commandthatdoesnotlikeverbatim{Text \usebox{\mybox}}
```

- **我想在通常不允许抄录内容的上下文中使用 `\mintinline` 命令。**`\mintinline` 命令在许多不允许普通抄录命令（如 `\verb`）的地方已经可以正常工作，请先尝试它。如果它不起作用，最简单的替代方法之一是将代码保存在一个盒子中，然后稍后使用它。例如：

```
\newsavebox\mybox
\begin{lrbox}{\mybox}
\mintinline{cpp}{std::cout}
\end{lrbox}

\commandthatdoesnotlikeverbatim{Text \usebox{\mybox}}
```

- **Extended characters do not work inside minted commands and environments, even when the inputenc package is used.** Version 2.0 adds support for extended characters under the pdfTeX engine. But if you need characters that are not supported by inputenc, you should use the XeTeX or LuaTeX engines instead.

- **即使使用了 inputenc 宏包，minted 命令和环境中仍无法使用扩展字符。**2.0 版本在 pdfTeX 引擎下添加了对扩展字符的支持。但是，如果你需要使用 inputenc 不支持的字符，应该改用 XeTeX 或 LuaTeX 引擎。

- **The polyglossia package is doing undesirable things to code. (For example, adding extra space around colons in French.)** You may need to put your code within

- **polyglossia 宏包在代码中产生了不希望的效果（例如，在法语中在冒号周围添加额外的空格）。**你可能需要将代码放在 `\begin{english}...\end{english}`

\begin{english}...\end{english}. This may done for all minted environments using etoolbox in the preamble:

```
\usepackage{etoolbox}
\BeforeBeginEnvironment{minted}{\begin{english}}
\AfterEndEnvironment{minted}{\end{english}}
```

Tabs are being turned into the character sequence ^^I. This happens when you use XeLaTeX. You need to use the -8bit command-line option so that tabs may be written correctly to temporary files. See http://tex.stackexchange.com/questions/58732/how-to-output-a-tabulation-into-a-file for more on XeLaTeX's handling of tab characters.

- **The caption package produces an error when \captionof and other commands are used in combination with minted.** Load the caption package with the option compatibility=false. Or better yet, use minted's newfloat package option, which provides better caption compatibility.

- **I need a listing environment that supports page breaks.** The built-in listing environment is a standard float; it doesn't support page breaks. You will probably want to define a new environment for long floats. For example,

```
\usepackage{caption}
\newenvironment{longlisting}{\captionsetup{type=listing}}{}
```

With the caption package, it is best to use minted's newfloat package option. See http://tex.stackexchange.com/a/53540/10742 for more on listing environments with page breaks.

- **I want to use a custom script/executable to access Pygments, rather than pygmentize.** Redefine \MintedPygmentize:

```
\renewcommand{\MintedPygmentize}{...}
```

- **I want to use the command-line option -output-directory, or MiKTeX's -aux-directory, but am getting errors.** Use the package option outputdir to specify the location of the output directory. Unfortunately, there is no way for minted to detect the output directory

中。可以使用 etoolbox 宏包在导言区为所有 minted 环境进行如下设置：

```
\usepackage{etoolbox}
\BeforeBeginEnvironment{minted}{\begin{english}}
\AfterEndEnvironment{minted}{\end{english}}
```

- **制表符被转换为字符序列 ^^I。**这是由于使用 XeLaTeX 引擎导致的。你需要使用 -8bit 命令行选项，以便正确将制表符写入临时文件。请参考 http://tex.stackexchange.com/questions/58732/how-to-output-a-tabulation-into-a-file 了解 XeLaTeX 处理制表符的更多信息。

- **caption 宏包在与 \captionof 和其他命令结合使用时会产生错误。**使用选项 compatibility=false 加载 caption 宏包。或者更好地，使用 minted 的 newfloat 宏包选项，它提供了更好的 caption 兼容性。

- **我需要一个支持分页的代码环境。**内置的 listing 环境是一个标准的浮动环境，不支持分页。你可能需要为长浮动对象定义一个新的环境。例如：

```
\usepackage{caption}
\newenvironment{longlisting}{\captionsetup{type=listing}}{}
```

使用 caption 宏包时，最好使用 minted 的 newfloat 宏包选项。请参考 http://tex.stackexchange.com/a/53540/10742 了解有关具有分页功能的 listing 环境的更多信息。

- **我想使用自定义脚本/可执行文件来访问 Pygments，而不是使用 pygmentize。**重新定义 \MintedPygmentize 命令：

```
\renewcommand{\MintedPygmentize}{...}
```

- **我想使用命令行选项 -output-directory 或 MiKTeX 的 -aux-directory，但是出现了错误。**使用宏包选项 outputdir 来指定输出目录的位置。不幸的是，minted 无法自动检测输出目录。

automatically.

- **I want extended characters in frame labels, but am getting errors.** This can happen with minted <2.0 and Python 2.7, due to a terminal encoding issue with Pygments. It should work with any version of Python with minted 2.0+, which processes labels internally and does not send them to Python.

- **`minted` environments have extra vertical space inside `tabular`.** It is possible to create a custom environment that eliminates the extra space. However, a general solution that behaves as expected in the presence of adjacent text remains to be found.

- **I'm receiving a warning from `lineno.sty` that "Command `\@parboxrestore` has changed."** This can happen when minted is loaded after csquotes. Try loading minted first. If you receive this message when you are not using csquotes, you may want to experiment with the order of loading packages and might also open an issue.

- **I'm using texi2pdf, and getting "Cannot stat" errors from tar**: This is due to the way that texi2pdf handles temporary files. minted automatically cleans up its temporary files, but texi2pdf assumes that any temporary file that is ever created will still exist at the end of the run, so it tries to access the files that minted has deleted. It's possible to disable minted's temp file cleanup by adding `\renewcommand{\DeleteFile}[2][]{}` after the `\usepackage{minted}`.

- **我想在帧标签中使用扩展字符，但是出现了错误。**这可能是因为 minted 版本小于 2.0 并且使用了 Python 2.7，这是由于 Pygments 存在的一个 终端编码问题。在任何带有 minted 2.0+ 版本的 Python 版本中，它都应该可以工作，因为 minted 内部处理标签并且不将其发送给 Python。

- **`minted` 环境在 `tabular` 环境中有额外的垂直空间。**可以 创建一个自定义环境 来消除额外的空间。然而，在存在相邻文本的情况下，尚未找到一个行为符合预期的通用解决方案。

- **我收到 `lineno.sty` 的警告信息，"Command `\@parboxrestore` has changed."** 这可能是因为在加载 csquotes 之后加载了 minted。尝试先加载 minted。如果在不使用 csquotes 的情况下收到此消息，你可以尝试调整包的加载顺序，并可能提交一个问题报告。

- **我正在使用 texi2pdf，并且从 tar 收到"Cannot stat"的错误消息：**这是因为 texi2pdf 处理临时文件的方式。minted 会自动清理其临时文件，但是 texi2pdf 假定任何创建过的临时文件在运行结束时仍然存在，因此尝试访问 minted 已删除的文件。你可以在 `\usepackage{minted}` 之后添加 `\renewcommand{\DeleteFile}[2][]{}` 来禁用 minted 的临时文件清理。

Acknowledgements

Konrad Rudolph: Special thanks to Philipp Stephani and the rest of the guys from `comp.text.tex` and `tex.stackexchange.com`.

Geoffrey Poore:

- Thanks to Marco Daniel for the code on `tex.stackexchange.com` that inspired automatic line breaking.

- Thanks to Patrick Vogt for improving TikZ externalization compatibility.

致谢

Konrad Rudolph: 特别感谢 Philipp Stephani 和其他来自 `comp.text.tex` 和 `tex.stackexchange.com` 的人们。

Geoffrey Poore:

- 感谢 Marco Daniel 在 `tex.stackexchange.com` 上提供的代码，启发了自动换行功能。

- 感谢 Patrick Vogt 改进了与 TikZ 的外部化兼容性。

- Thanks to @muzimuzhi for assistance with GitHub issues.

- Thanks to @jfbu for suggestions and discussion regarding support for arbitrary Pygments style names (#210, #294, #299, #317), and for debugging assistance.

- 感谢 @muzimuzhi 在 GitHub 问题中提供的帮助。

- 感谢 @jfbu 对于支持任意 Pygments 样式名称（#210、#294、#299、#317）的建议和讨论，以及提供的调试帮助。