The minted package: Highlighted source code in LaTeX

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

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

翻译 by
virhuiai@qq.com

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 是一个利用强大的 Pygments 库实现语法高亮的包。该包还提供了选项来自定义高亮的源代码输出。

License
许可证

LaTeX Project Public License (LPPL) version 1.3.
Additionally, the project may be distributed under the terms of the 3-Clause (“New”) BSD license: http://opensource.org/licenses/BSD-3-Clause.

LaTeX 项目公共许可证（LPPL）版本 1.3。
此外，该项目可以按照 3 条款（“新”）BSD 许可证的条款分发：http://opensource.org/licenses/BSD-3-Clause。

目录

1 Introduction 介绍

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

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}
```

will highlight a piece of code in a chosen language. The appearance can be customized with a number of options and color schemes.

将会高亮显示所选语言的一段代码。外观可以使用许多选项和颜色方案进行自定义。

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.

与其他一些包（特别是 listings）不同，minted 需要安装额外的软件 Pygments。这可能看起来像是一个劣势，但也有显著的优势。

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.

相比于传统的包，Pygments 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。另一方面，Pygments 可以完全定制以高亮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、不同类型的标识符和像 HTML 标记这样的奇特构造。

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

对于某些语言，这特别有用。考虑以下 Ruby 代码作为一个极端但同时也是典型的例子：

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

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.

这里有四种不同的标识符颜色（如果您关键字也算上则为五种），以及来自字符串内部的转义，它们都对 Pygments 没有问题。

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

此外，安装 Pygments 实际上非常简单（见下一节）。

2 Installation 安装

2.1 Prerequisites 前提条件

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:

Pygments 是用 Python 写的，所以请确保您的系统上安装了 Python 2.6 或更高版本[1]：

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

[1]译注：最新的使用 pip3 安装，2 版本的安装不了了！

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

如果您没有安装 Python，则可以从 Python 官网下载，或使用操作系统的软件包管理器。

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:

一些 Python 发行版包括 Pygments（请参阅 Python 站点下“替代实现”选项中的一些选项）。否则，您需要手动安装 Pygments。这可以通过安装 setuptools 完成，它可以促进 Python 应用程序的分发。然后，您可以使用以下命令安装 Pygments：

```
$ 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`:

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` 安装 Pygments：

```
$ 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.

如果您已经安装了 Pygments，请注意推荐使用最新版本（至少为 1.4 或更高版本）。一些功能，如 `escapeinside`，仅适用于 2.0+。minted 可能与早至 1.2 的版本一起使用，但没有保证。

2.2 Required packages
必要的依赖包

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

minted 要求您的系统上可用并且相对最新的以下包。所有这些都随最近的 TEX 发行版一起发布。

- `keyval`
- `kvoptions`
- `fancyvrb`
- `fvextra`
- `upquote`
- `float`
- `ifthen`
- `calc`
- `ifplatform`
- `pdftexcmds`
- `etoolbox`
- `xstring`
- `xcolor`
- `lineno`
- `framed`
- `shellesc (for luatex 0.87+)`
- `catchfile`

2.3 Installing minted
安装 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.

你可以使用你的 TEX 发行版的软件包管理器安装 minted。否则，如果你想要最新版本，你可以按照以下指示手动安装。

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.

你可以从项目主页下载`minted.sty`。我们需要安装该文件以便 TEX 能够找到它。请参考TEX 常见问题来做到这一点。如果你只想尝试最新版本，你可以

在 TEX 安装中找到当前的`minted.sty`并将其替换为最新版本。或者你也可以将最新的`minted.sty`放在与你要使用它的文件相同的目录中。

3 Basic usage 基本用法

3.1 Preliminary 预备知识

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:

由于 minted 需要调用外部工具（即 Pygments），你需要使用`-shell-escape`选项告诉 LATEX 排版引擎。否则它不会允许这样的调用。实际上，与其像这样调用排版引擎：

```
$ latex input
```

you need to call it like this:

你需要像这样调用它：

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

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

对于其他排版引擎（例如`pdflatex`或`xelatex`）也是同样的情况。

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.

你应该知道，使用`-shell-escape`允许 LATEX 在你的系统上运行潜在的任意命令。最好仅在需要时使用`-shell-escape`，并且仅与来自可信源的文档一起使用。

Working with OS X
在 OS X 上使用

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

如果你在某些版本/配置的 OS X 上使用 minted，并且使用具有大量代码块（> 256）的缓存，你可能会收到以下错误：

```
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 to increase the number of files that may be used:

这是由于操作系统处理文件的方式与缓存工作方式的结合引起的。为了解决这个问题，你可以使用 OS X 命令增加可使用的文件数量：

`launchctl limit maxfiles` or `ulimit -n`

3.2 A minimal complete example 一个最小的完整示例

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

下面的文件`minimal.tex`展示了 minted 的基本用法。

```
\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:

通过这样编译源文件:

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

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

运行上述命令后，会生成 `minimal.pdf` 文件，其中包含如下代码的输出：

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

3.3 Formatting source code 格式化源代码

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

使用 minted 包很简单。例如，若要高亮显示一些 Python 代码，可以使用以下代码片段（右侧为输出结果）:

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

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

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

此外，该环境还接受一些以 `key=value` 形式的选项，详见下文。

`\mint` For a single line of source code, you can alternatively use a shorthand notation:

对于单行源代码，您可以使用简写形式：

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

```
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.

这样可以使用命令而不是环境来排版一行代码，因此节省了一些打字量，但输出结果与 `minted` 环境的结果等效。

The code is delimited by a pair of identical characters, similar to how `\verb` works. The complete syntax is

代码由一对相同的字符分隔，类似于 `\verb` 的工作方式。完整的语法是

`\mint[`⟨*options*⟩`]{`⟨*language*⟩`}`⟨*delim*⟩⟨*code*⟩⟨*delim*⟩

where the code delimiter can be almost any punctuation character. The ⟨*code*⟩ may also be 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.

其中代码定界符可以是几乎任意标点符号。代码片段也可以用匹配的花括号 `{}` 进行定界，只要代码本身不包含未匹配的花括号。该命令也支持一些选项，详见下文。

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.

注意，`\mint` 命令**不能用于行内排版**。相反，它是当仅有一行代码时使用 `minted` 的一种快捷方式。对于行内排版，应使用 `\mintinline` 命令。

`\mintinline` Code can be typeset inline:

可以行内排版代码：

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

The syntax is

语法为

`\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, `{}`.

定界符可以是一对字符，如 `\mint`。它们也可以是匹配的花括号 `{}`。

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

该命令被精心制作，以便在大多数情况下，在其他命令内使用时它会正确地工作。[3]

`\inputminted` Finally, there's the `\inputminted` command to read and format whole files. Its syntax is

最后，有`\inputminted`命令来读取和格式化整个文件。它的语法为

`\inputminted[`⟨*options*⟩`]{`⟨*language*⟩`}{`⟨*filename*⟩`}`

3.4 Using different styles 使用不同的样式

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

你可以选择由 Pygments 提供的另一个样式表，而不是使用默认样式表。这可以通过以下方式实现：

```
\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 be set via `\setminted` and via the optional argument for each command and environment.[4]

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

[3]例如，`\mintinline`在脚注中也可以使用！主要例外是当代码包含百分号`%` 或井号 `#` 字符，或未匹配的花括号时。

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

完整的语法是`\usemintedstyle[`⟨*language*⟩`]{`⟨*style*⟩`}`。可以为整个文档设置样式（未指定语言），或仅为特定语言设置样式。请注意，样式也可以通过`\setminted` 和每个命令和环境的可选参数进行设置。[5]

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:

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

```
$ pygmentize -L styles
```

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

创建自己的样式也很容易。只需按照Pygments 网站提供的说明即可。

3.5 Supported languages 支持的语言

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

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

```
$ pygmentize -L lexers
```

4 Floating listings 浮的代码块

`listing` 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

[5]版本 2.0 添加了可选语言参数，并删除了必须在导言中使用该命令的限制。

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框中，其默认的放置方式为`tbp`，与图表相同。您还可以提供`\caption`和`\label`来定义此类代码清单（像使用`figure` 和 `table`环境一样）:

```
\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:

将产生/得到:

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

Listing 1: Example of a listing.

Listing 1 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,

默认的代码清单放置方式可以轻松修改。当使用包选项`newfloat=false`（默认值）时，minted 使用 float 包来创建`listing`环境。通过重新定义`\fps@listing`可以修改代码清单的放置方式。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`:

当使用选项`newfloat=true`时，将使用更强大的 newfloat 包来创建`listing`环境。在这种情况下，可使用 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`宏将插入文档中所有（浮动的）清单的列表：

```
\listoflistings
```

List of Listings

1 Example of a listing. 15

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，`listing` 环境是使用 float 宏包创建的。在这种情况下，可以使用下面描述的 `\listingscaption` 和 `\listoflistingscaption` 宏来自定义标题和列表。如果使用 `newfloat` 选项加载了 minted，则 `listing` 环境将使用更强大的 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 newfloat's `\SetupFloatingEnvironment` command. For example, the string "Listing" in the caption could be changed to "Program code" using

当使用 newfloat 创建 `listing` 环境时，应使用 `\SetupFloatingEnvironment` 命令进行自定义。例如，可以使用以下命令将标题中的字符串 "Listing" 更改为 "程序代码"：

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

And “List of Listings” could be changed to “List of Program Code” with

并且可以使用以下命令将 “List of Listings” 更改为 “程序代码列表”:

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

Refer to the newfloat and caption documentation for additional information.

有关详细信息，请参阅 newfloat 和 caption 文档。

`\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:

（仅适用于未使用包选项 `newfloat` 的情况。）可以更改列表标题中的字符串 “Listing”。要做到这一点，只需重新定义宏 `\listingscaption`，例如:

```
\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`:

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

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

5 Options 选项

5.1 Package options 宏包选项

chapter

可以在加载 minted 宏包时传递 section 或 chapter 选项，以控制 LaTeX 如何计数 listing 浮动体。例如，以下命令将导致列表按章进行计数：

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}
```

section

要控制 LaTeX 如何计数listing浮动对象,可以在加载 minted 包时传递section或chapter选项。

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

cache=⟨*boolean*⟩
(default: true)

minted 通过将代码保存到临时文件，通过 Pygments 对代码进行高亮处理并将输出保存到另一个临时文件，然后将输出输入到 LaTeX 文档中来工作。如果需要突出显示多个代码块，则这个过程可能会变得非常缓慢。为避免这种情况，该宏包提供了一个 cache 选项。默认情况下，此选项已启用。

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.

选项 cache 在文档的根目录下创建一个名为 The cache option creates a directory

_minted-⟨*jobname*⟩

的目录（可以使用选项 cachedir 自定义此目录名）。[6]高亮代码文件存储在此

目录中，以便将来无需再次高亮代码。在大多数情况下，缓存将显著加快文档的编译速度。

in the document's root directory (this may be customized with the `cachedir` option).[7]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.

不再使用的缓存文件将被自动删除。[8]

Cached files that are no longer in use are automatically deleted.[9]

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

这允许指定存储缓存文件的目录。路径应使用正斜杠，即使在 Windows 下也是如此。

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

必须转义特殊字符。例如，`cachedir=/mintedcache` 将无法工作，因为波浪号 ~ 将转换为非换行空格的 LaTeX 命令，而不是被直接处理。可以使用

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}`
或等效的解决方案。 or an equivalent solution.

路径可以包含空格，但仅在整个 ⟨*directory*⟩ 被大括号 `{}` 包裹并且空格被引用时。例如，

Paths may contain spaces, but only if the entire ⟨*directory*⟩ is wrapped in

[7]实际上，目录的名称使用"已消毒"的 ⟨*jobname*⟩ 副本命名，其中空格和星号已替换为下划线，双引号已被剥离。如果文件名包含空格，则 `\jobname` 将包含一个带引号的名称，除了使用星号替换空格的旧版 MiKTeX 外。使用"已消毒"的 ⟨*jobname*⟩ 比适应各种转义约定更简单。

[7]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.

[9]这取决于主辅助文件未被删除或损坏。如果发生这种情况，可以简单地删除缓存目录并重新开始。

[9]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.

curly braces {}, and only if the spaces are quoted. For example,

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

请注意，如果指定了 `outputdir`，则缓存目录相对于其而言。

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

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

在某些情况下，可能希望在不允许 `-shell-escape` 的环境中使用 minted。文档可能会被提交给出版商或预印本服务器，或者与不支持 `-shell-escape` 的在线服务一起使用。只要不需要修改 minted 的内容，就可以这样做。

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.

使用 `finalizecache` 选项进行编译会为在没有 `-shell-escape` 的环境中使用**准备缓存**。[10]完成后，可以将 `finalizecache` 选项替换为 `frozencache` 选项，以后可以在不需要 `-shell-escape` 的情况下使用冻结（静态）缓存。

Compiling with the `finalizecache` option prepares the cache for use in an environment without `-shell-escape`.[11]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`.

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

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

Use a frozen (static) cache created with the `finalizecache` option. When `frozencache` is on, `-shell-escape` is not needed, and Python and Pyg-

[11]通常，缓存文件使用突出显示设置和突出显示文本的 MD5 哈希命名。`finalizecache` 使用 `listing<number>.pygtex` 方案重命名缓存文件。这使得匹配文档内容和缓存文件更加简单，也是必要的，因为在 TeX Live 2016 之前，XeTeX 引擎缺乏 pdfTeX 和 LuaTeX 具有的内置 MD5 能力。

[11]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.

ments are not required. In addition, any external files accessed through `\inputminted` are no longer necessary.

必须谨慎使用此选项。文档在 minted 考虑的范围内必须处于最终形式，并且在打开 `frozencache` 之前必须使用 `finalizecache` 进行编译。打开此选项后，除非直接编辑缓存文件，否则无法修改 minted 内容。无法更改需要 Pygments 或 Python 的任何 minted 设置。如果在打开 `frozencache` 后不正确地修改了 minted 内容，则 minted 无法检测到修改。

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.

如果正在使用 `frozencache`，并且要验证 minted 设置或内容是否未被以无效的方式修改，则可以使用以下过程测试缓存。

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. 获取使用 `frozencache` 的缓存的副本。
   Obtain a copy of the cache used with `frozencache`.

2. 在支持 `-shell-escape` 的环境中使用 `finalizecache=true` 和 `frozencache=false` 编译文档。这本质上重新生成了冻结（静态）缓存。
   Compile the document in an environment that supports `-shell-escape`, with `finalizecache=true` and `frozencache=false`. This essentially regenerates the frozen (static) cache.

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`；在 Windows 下，可能需要使用 `fc`。如果 minted 内容和设置未以无效的方式进行修改，则所有文件都将相同（假设两个缓存都使用兼容的 Pygments 版本）。
   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 used for both caches).

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

设置用于排版代码的字体编码。

Set font encoding used for typesetting code.

例如，`fontencoding=T1`。 For example, `fontencoding=T1`.

`draft=⟨boolean⟩` (default: false)

这仅使用 fancyvrb 进行所有排版；不使用 Pygments。这是以失去语法高亮和一些其他 minted 功能为代价的，但编译速度更快。性能应与直接使用 fancyvrb 时基本相同；不使用外部临时文件。请注意，如果您在编译之间没有更改太多的代码，则缓存模式和草稿模式之间的性能差异可能很小。还请注意，草稿设置通常从文档类继承。

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.

草稿模式不支持`autogobble`。在草稿模式下，常规的`gobble`，`linenos`和大多数与语法高亮无关的选项仍将正常运行。

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

通常可以在草稿模式下编译文档而无需使用 shell escape。ifplatform 包可能会发出关于由于禁用了 shell escape 而导致功能有限的警告，但在几乎所有情况下都可以忽略此警告。（仅当您的系统配置不寻常，使得`\ifwindows`宏必须借助 shell escape 来确定系统时，才真正需要 shell escape。有关详细信息，请参见 ifplatform 文档。）

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.)

如果设置了`cache`选项，则在草稿模式下保留所有现有缓存文件。这允许在不需要每次完全重建缓存的情况下间歇性地使用缓存模式和草稿模式。一旦关闭草稿模式，未使用的缓存文件的自动清理将恢复。（这假设辅助文件在此期间未被删除；它包含缓存历史记录，并允许自动清理未使用的文件。）

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.)

`final=`⟨*boolean*⟩ `(default: true)`

这是`draft`的相反；它等效于`draft=false`。
同样，请注意，`draft`和`final`设置通常从文档类继承。

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.

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

这个选项使用 `kpsewhich` 来定位需要突出显示的文件。一些构建工具（如 `texi2pdf`）通过修改 `TEXINPUTS` 来实现功能；在某些情况下，用户也可以自定义 `TEXINPUTS`。`kpsewhich` 选项允许 minted 与这些配置一起工作。

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.

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

该选项不使得 minted 与 LaTeX 的 `-output-directory` 和 `-aux-directory` 命令行选项一起工作。对于这些选项，请参见 `outputdir` 包选项。

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.

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

minted 在后台使用 fancyvrb 宏包进行代码排版。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结束的地方继续。langlinenos 选项使得 firstnumber 可以对每个语言进行单独设置，对于所有 minted 和 \mint 的使用情况。例如，考虑以下代码和输出。

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.

```
\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}
```

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

```
def func
    puts "message"
end
```

```
def g(x):
    return 2*x
```

如果没有使用 `langlinenos` 选项，则第二个 Python 环境中的行号将不会从第一个 Python 环境结束的地方开始编号，而是从 Ruby 的行号开始。

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.

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

默认情况下，`listing` 环境使用 float 宏包创建。`newfloat` 选项使用 newfloat 来创建该环境，这可以更好地与 caption 宏包集成。

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.

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

对于 LaTeX 中的 `-output-directory` 和 `-aux-directory`（MiKTeX）命令行选项，对于 minted 会造成问题，因为 minted 的临时文件保存在 `<outputdir>` 中，但 minted 仍会在文档根目录中查找它们。无法访问命令行选项的值，以便 minted 可以自动查找正确的位置。但是，可以将输出目录手动指定为包选项。

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.

应使用绝对路径或相对于文档根目录的路径来指定输出目录。路径应使用正斜杠，即使在 Windows 下也是如此。必须转义特殊字符，而空格则需要用引号引起来，并需要用大括号`{}`将整个⟨*directory*⟩ 包装起来。有关转义和引用的示例，请参见上面的`cachedir`。

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.

5.2 Macro option usage
宏选项用法

所有 minted 高亮命令都接受相同的选项集。选项指定为逗号分隔的`key=value`对的列表。例如，我们可以指定行应编号：

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:

```
\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 }
```

选项值`true`也可以完全省略（包括 “=” ）。要进一步自定义行号的显示，请覆盖`\theFancyVerbLine`命令。有关详细信息，请参阅 fancyvrb 文档。

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.

`\mint`接受相同的选项：　　`\mint` accepts the same options:

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

nt[linenos]perl`$x=~/foo/`

这里是另一个示例：我们想在注释中使用 LaTeX 数学模式：

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

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

# Returns $\sum_{i=1}^{n} i$

```
def sum_from_one_to(n):
    r = range(1, n + 1)
    return sum(r)
```

为了使您的 LaTeX 代码更易于阅读，您可能希望在minted环境中缩进代码。选项gobble从输出中删除这些不必要的空白字符。还有一个autogobble选项，它会自动检测这个空格的长度。

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.

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

versus

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

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

versus

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

`\setminted` 您可能希望为整个文档或整个语言设置选项。这可以通过使用命令

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,...*⟩`}`

实现。特定于语言的选项会覆盖文档范围内的选项。个别命令和环境选项会覆盖特定于语言的选项。

Language-specific options override document-wide options. Individual command and environment options override language-specific options.

`\setmintedinline` 您可能希望为`\mintinline` 单独设置选项，无论是为整个文档还是为特定语言。这可以通过使用命令`\setmintedinline` 实现。语法如下：

`\setmintedinline[`⟨*language*⟩`]{`⟨*key=value,...*⟩`}`

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

特定于语言的选项会覆盖文档范围内的选项。个别命令选项会覆盖特定于语言的选项。所有使用`\setmintedinline` 指定的设置都会覆盖使用`\setminted` 指定的设置。也就是说，内联设置的优先级始终高于一般设置。

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.

5.3 Available options 可用选项

以下是可用选项的完整列表。有关更详细的选项描述，请参阅 fancyvrb 和 Pygments 文档。

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

`autogobble` (boolean) (default: `false`)

从代码中删除（gobble）所有常见的前导空格。基本上是自动确定要删除什么的 `gobble` 的版本。适用于最初未缩进但在粘贴到 LaTeX 文档后手动缩进的代码。

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.

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

```
...text.

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

`baselinestretch` (dimension) (default: ⟨*document default*⟩)

用于代码列表内部的行距的值。

Value to use as for baselinestretch inside the listing.

`beameroverlays` (boolean) (default: `false`)

在使用 `escapeinside` 和 `texcomments` 时，为 `<` 和 `>` 字符提供其正常的文本含义，以便形式为 `\only<1>{...}` 的 beamer 叠加效果正常工作。

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*⟩)

（当 `breaklines=true` 时）在指定的字符后换行，而不仅仅在空格处换行。不适用于 `\mintinline`。

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

例如，`breakafter=-/` 允许在连字符或斜杠后面断行。`breakafter` 中给定的特殊字符应进行反斜杠转义（通常为 `#`, `{`, `}`, `%`, `[`, `]`；通过 `\\` 可以获得反斜杠 `\`）。

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 `\\`).

另一种选择，请参见 `breakbefore`。当为相同字符使用 `breakbefore` 和 `breakafter` 时，`breakbeforegroup` 和 `breakaftergroup` 必须具有相同的设置。

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 = 'SomeTextThatGoesOnAndOnForSoLongThatItCould⌋
↪  NeverFitOnOneLine'
```

breakaftergroup (boolean) (default: true)

当使用 breakafter 时，将所有相邻的相同字符分组在一起，并仅在最后一个字符后允许断行。当为相同字符使用 breakbefore 和 breakafter 时，breakbeforegroup 和 breakaftergroup 必须具有相同的设置。

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.

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

在由 breakafter 插入的换行符前插入的符号。

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

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

在由 breakafter 插入的换行符后插入的符号。

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

breakanywhere (boolean) (default: false)

在 breaklines = true 时，不仅在空格处，而是在任何位置断行。不适用于 \mintinline。

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 = 'SomeTextThatGoesOnAndOnForSoLongThatItCouldN⌋
↪  everFitOnOneLine'
```

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

在由 `breakanywhere` 插入的换行符前插入的符号。

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

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

在由 `breakanywhere` 插入的换行符后插入的符号。

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

`breakautoindent` (boolean) (default: `true`)

自动缩进延续行，使其缩进等级与第一行相同。当同时使用 `breakautoindent` 和 `breakindent` 时，缩进会加和。此缩进与 `breaksymbolindentleft` 组合以给出总实际左缩进。不适用于 `\mintinline`。

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`.

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

当 `breaklines=true` 时，在指定字符之前而非仅在空格处断行。不适用于 `\mintinline`。

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

例如，`breakbefore=A` 允许在大写字母 A 之前断行。应对特殊字符应使用反斜杠转义 (通常为 `#`, `{`, `}`, `%`, `[`, `]`；反斜杠 `\` 可通过 `\\` 获得)。

For example, `breakbefore=A` would allow breaks before capital A's. Special characters given to `breakbefore` should be backslash-escaped (usually `#`, `{`, `}`, `%`, `[`, `]`; the backslash `\` may be obtained via `\\`).

另请参见 `breakafter`。当 `breakbefore` 和 `breakafter` 用于相同字符时，`breakbeforegroup` 和 `breakaftergroup` 必须具有相同的设置。

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

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

```
some_string = 'SomeTextThatGoesOn⌋
↪  AndOnForSoLongThatItCouldNeverFitOnOneLine'
```

`breakbeforegroup` (boolean) (default: `true`)
当使用 `breakbefore` 时，将所有相邻的相同字符分组在一起，并仅在第一个字符之前允许断行。当 `breakbefore` 和 `breakafter` 用于相同字符时，`breakbeforegroup` 和 `breakaftergroup` 必须具有相同的设置。

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.

`breakbeforesymbolpre` (string) (default: `\,\footnotesize\ensuremath{_\rfloor}`, ⌋)
用于由 `breakbefore` 插入断行的前置符号。

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

`breakbeforesymbolpost` (string) (default: ⟨*none*⟩)
用于由 `breakbefore` 插入断行的后置符号。

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

breakbytoken (boolean) (default: false)
仅在不在标记中的位置处断行;防止标记被行中断开。默认情况下,breaklines 在最靠近边缘的空格处断行。虽然这可以使必需的换行数最小化，但如果在字符串或类似标记的中间出现换行，则可能会不方便。

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.

这与 draft 模式不兼容。Pygments 标记的完整列表可在http://pygments.org/docs/tokens/上找到。如果 breakbytoken 提供的断点出现在意外的位置，则可能表明该语言的 Pygments 词法分析器存在缺陷或短板。

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.

breakbytokenanywhere (boolean) (default: false)
类似于 breakbytoken，但还允许在紧邻的标记之间，而不仅仅是在由空格分隔的标记之间换行。与 breakanywhere 一起使用 breakbytokenanywhere 是多余的。

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.

breakindent (dimension) (default: ⟨*breakindentnchars*⟩)
当断行时,将续行缩进这么多。当同时使用breakautoindent和breakindent时,缩进会相加。此缩进与breaksymbolindentleft结合使用，以给出总实际左缩进量。

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.

不适用于`\mintinline`。

Does not apply to `\mintinline`.

`breakindentnchars` (integer) (default: `0`)

这允许将`breakindent`指定为字符数的整数，而不是作为维度（假定为固定宽度字体）。

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

`breaklines` (boolean) (default: `false`)

在 `minted` 环境和`\mint` 命令中自动换行长行，并在`\mintinline` 中换行较长的行。

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

默认情况下，在空格字符处自动换行。使用 `breakanywhere` 启用任意位置的换行；使用 `breakbytoken`、`breakbytokenanywhere`、`breakbefore` 和 `breakafter` 进行更精细的换行。目前，仅 `breakbytoken` 和 `breakbytokenanywhere` 与`\mintinline`一起使用。使用 `escapeinside` 转义到 LaTeX 并插入手动换行也是一种选择。例如，使用 `escapeinside=`，然后在适当的位置插入`\\`。（请注意，`escapeinside` 不适用于字符串内部。）

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.)

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

...text.

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

`minted` 和`\mint` 中的断行可以通过多种方式自定义。要自定义断行行的缩

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。要自定义为断裂符号提供的额外缩进，请使用 breaksymbolindentleft 和 breaksymbolindentright。由于默认情况下仅使用左手符号，因此还可以使用别名选项 breaksymbol，breaksymbolsep 和 breaksymbolindent 进行修改。请注意，由于在行内上下文中不相关，因此这些选项均不适用于\mintinline。

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.

下面是一个使用这些选项自定义 minted 环境的示例。它使用 dingbat 包中的\carriagereturn 符号。

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

```
\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'
```

使用带有大量文本背景色的 Pygments 样式的自动换行受到限制。这种着色是通过\colorbox 实现的，它无法跨行换行。可能可以创建一个支持换行的\colorbox 替代方案，例如使用 TikZ，但作者不知道有什么令人满意的解决方案。唯一的当前替代方案是重新定义\colorbox，使其不起作用。例如，

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}}
```

使用 etoolbox 包在所有 minted 环境中重新定义\colorbox。

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

如果您使用 `showspaces=true`,则自动换行将不起作用,除非您使用 `breakanywhere` 或 `breakafter=\space`。

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

`breaksymbol` (string) (default: `breaksymbolleft`)
`breaksymbolleft` 的别名。

Alias for `breaksymbolleft`.

`breaksymbolleft` (string) (default: `\tiny\ensuremath{\hookrightarrow}`, $\hookrightarrow$)
在 `breaklines=true` 时,用于表示续行的符号。若要没有符号,则将 `breaksymbolleft` 设置为空字符串(“`=,`” 或 “`=`”)即可。当使用时，该符号被包装在花括号 中，因此不会出现格式命令(如`\tiny`)的 “逃逸”。

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.”

`\hookrightarrow` 和 `\hookleftarrow` 可通过 graphicx 提供的 `\rotatebox` 命令进行进一步定制。其他可能有用的箭头类型符号可在 dingbat(`\carriagereturn`)和 mnsymbol(钩和曲线箭头)等包中获得。

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.

不适用于 `\mintinline`。

Does not apply to `\mintinline`.

`breaksymbolright` (string) (default: ⟨*none*⟩)
在 `breaklines=true` 时，在断点(右侧)使用的符号。不会出现在断行的最

后一个片段的末尾。

The symbol used at breaks (right) when `breaklines=true`. Does not appear at the end of the very last segment of a broken line.

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

Alias for `breaksymbolindentleft`.

`breaksymbolindentnchars` (integer) (default: ⟨*breaksymbolindentleftnchars*⟩)
`breaksymbolindentleftnchars` 的别名。

Alias for `breaksymbolindentleftnchars`.

`breaksymbolindentleft` (dimension) (default: ⟨*breaksymbolindentleftnchars*⟩)
为给 `breaksymbolleft` 留出空间而提供的额外左缩进。仅当存在 `breaksymbolleft` 时才应用此缩进。

The extra left indentation that is provided to make room for `breaksymbolleft`. This indentation is only applied when there is a `breaksymbolleft`.

不适用于 `\mintinline`。

Does not apply to `\mintinline`.

`breaksymbolindentleftnchars` (integer) (default: 4)
这允许将 `breaksymbolindentleft` 指定为字符的整数数量，而不是维度（假定为等宽字体）。

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

`breaksymbolindentright` (dimension) (default: ⟨*breaksymbolindentrightnchars*⟩)
为给 `breaksymbolright` 留出空间而提供的额外右缩进。仅当存在 `breaksymbolright` 时才应用此缩进。

The extra right indentation that is provided to make room for `breaksymbolright`. This indentation is only applied when there is a `breaksymbolright`.

`breaksymbolindentrightnchars` (integer) (default: 4)

这允许将 breaksymbolindentright 指定为字符的整数数量，而不是维度（假定为等宽字体）。

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

breaksymbolsep (dimension) (default: ⟨*breaksymbolsepleftnchars*⟩)

breaksymbolsepleft 的别名。

Alias for breaksymbolsepleft.

breaksymbolsepnchars (integer) (default: ⟨*breaksymbolsepleftnchars*⟩)

breaksymbolsepleftnchars的别名。

Alias for breaksymbolsepleftnchars.

breaksymbolsepleft (dimension) (default: ⟨*breaksymbolsepleftnchars*⟩)

breaksymbolleft和相邻文本之间的间距。

The separation between the breaksymbolleft and the adjacent text.

breaksymbolsepleftnchars (integer) (default: 2)

允许将breaksymbolsepleft指定为字符数的整数，而不是作为尺寸（假定固定宽度字体）。

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

breaksymbolsepright (dimension) (default: ⟨*breaksymbolseprightnchars*⟩)

breaksymbolright和相邻文本之间的*最小*间距。这是breaksymbolright和相邻文本可以到达的最远范围之间的距离。实际上，\linewidth通常不会是字符宽度（假设固定宽度字体）的确切整数倍，因此breaksymbolright和相邻文本之间的实际间距通常会大于breaksymbolsepright。这确保了断行符与左右边缘的间距相同。如果希望获得相同的文本间距，请调整breaksymbolsepright。（有关实现详细信息，请参见 fvextra 中\FV@makeLineNumber的定义。）

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.)

`breaksymbolseprightnchars` (integer) (default: 2)
允许将`breaksymbolsepright`指定为字符数的整数，而不是作为尺寸（假定固定宽度字体）。

Allows `breaksymbolsepright` to be specified as an integer number of characters rather than as a dimension (assumes a fixed-width font).

`bgcolor` (string) (default: ⟨*none*⟩)
代码清单的背景颜色。请注意，此选项有几个限制（下面描述）；请参见下面的“框架替代方案”获取更强大的替代方案。

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

此选项的值*不能*是颜色命令。相反，它必须是已经定义过的颜色的颜色*名称*，给定为字符串：

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:

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

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

这个选项将 `minted` 环境和`\mint` 命令放置在 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`，这会阻止分页并导致周围文本的不良间距。) 请注意，如果在 `breaklines=true` 下使用 `bgcolor`，并且在分页之前发生换行，则某些情况下文本可能会延伸到

彩色背景下方。在这种情况下最好使用更高级的装框包；请参见下面的“其他装框包”。

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.

这个选项将`\mintinline`置于`\colorbox`中，**不允许换行**。如果您想使用`\setminted`设置背景颜色，并且只想对 `minted` 和`\mint` 使用背景颜色，则可以使用`\setmintedinlinebgcolor=` 关闭行内命令的着色。

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.

Framing alternatives
其他装框包

如果您想要更可靠和先进的背景颜色和装框选项，您应该考虑使用更高级的装框包，例如 mdframed 或 tcolorbox。使用 etoolbox 包，可以轻松地将框架添加到 minted 命令和环境中，该包会自动加载到 minted 中。例如，使用 mdframed：

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:

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

一些装框包还提供了专门用于此目的的内置命令。例如，mdframed 提供了一

个\surroundwithmdframed 命令，可用于为所有 minted 环境添加边框：

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 甚至提供了一个内置的装框环境，支持 minted。只需在导言区使用\tcbuselibraryminted，然后在 tcblisting 环境中放置代码：

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 提供了其他命令和环境，用于微调代码列表的外观和处理外部代码文件。

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

codetagify (list of strings) (default: highlight XXX, TODO, BUG, and NOTE)
在注释和文档字符串中高亮显示特殊的代码标签。

Highlight special code tags in comments and docstrings.

curlyquotes (boolean) (default: false)
默认情况下，反引号 ` 和打字机单引号 ' 总是以字面形式出现，而不会变成左右卷曲的单引号 ‘’. 该选项允许在需要时将这些字符替换为卷曲引号。

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.

`encoding` (string) (default: ⟨*system-specific*⟩)
设置 Pygments 期望的文件编码。也可参见 `outencoding`。

Sets the file encoding that Pygments expects. See also `outencoding`.

`escapeinside` (string) (default: ⟨*none*⟩)
在 `(string)` 中指定的两个字符之间转义到 LaTeX 中。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。转义字符不必相同。当将特殊的 LaTeX 字符用作转义字符时（例如，`escapeinside=\#\%`），必须转义这些字符。需要 Pygments 2.0+。

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+.

转义不适用于字符串和注释（对于注释，有 `texcomments`）。截至 Pygments 2.0.2，这意味着在某些词法分析器中，转义是“脆弱”的。由于 Pygments 实现了 `escapeinside` 的方式，任何类似于当前词法分析器中字符串或注释的“转义”LaTeX 代码都可能会破坏 `escapeinside`。针对此情况，有一个 Pygments 问题。有关更多细节和某些场景的有限解决方法，请访问 minted GitHub 站点。

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
```

请注意，当在转义符中使用数学内容时，除了在 verbatim 模式中通常活动的字符之外的任何活动字符都可能会导致问题。任何依赖于数学模式中特殊活动字符（例如 icomma）的软件包将会产生错误，如 `TeX capacity exceeded` 和`\leavevmode\kern\z@`。可以通过修改 `@noligs` 来解决这个问题，方法见`http://tex.stackexchange.com/questions/223876`。

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`.

`firstline` (integer) (default: `1`)
要显示的第一行。在该行之前的所有行将被忽略并不会出现在输出中。The first line to be shown. All lines before that line are ignored and do not appear in the output.

`firstnumber` (auto | last | integer) (default: `auto = 1`)
第一行的行号。Line number of the first line.

`fontfamily` (family name) (default: `tt`)
要使用的字体系列。`tt`、`courier`和`helvetica`是预定义的。The font family to use. `tt`, `courier` and `helvetica` are pre-defined.

`fontseries` (series name) (default: `auto` – the same as the current font)
要使用的字体系列。The font series to use.

`fontsize` (font size) (default: `auto` – the same as the current font)
要使用的字体大小，以字号命令的形式，例如`\footnotesize`。The size of the

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

`fontshape` (font shape) (default: `auto` – the same as the current font)
要使用的字体形状。The font shape to use.

`formatcom` (command) (default: ⟨*none*⟩)
在打印 verbatim 文本之前执行的格式化命令。A format to execute before printing verbatim text.

`frame` (none | leftline | topline | bottomline | lines | single) (default: `none`)
围绕源代码清单的框架类型。The type of frame to put around the source code listing.

`framerule` (dimension) (default: `0.4pt`)
框架的宽度。Width of the frame.

`framesep` (dimension) (default: `\fboxsep`)
框架和内容之间的距离。Distance between frame and content.

`funcnamehighlighting` (boolean) (default: `true`)
[For PHP only] 如果为`true`，则会突出显示内置函数名称。If `true`, highlights built-in function names.

`gobble` (integer) (default: `0`)
从每个输入行中删除前 n 个字符。Remove the first n characters from each input line.

`highlightcolor` (string) (default: `LightCyan`)
设置用于`highlightlines`的颜色，使用 color 或 xcolor 中的预定义颜色名称或通过`\definecolor`定义的颜色。

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

`highlightlines` (string) (default: ⟨*none*⟩)
基于行号突出显示单个行或一系列行。例如,`highlightlines={1, 3-4}`。行号是指如果`linenos=true`等,则会出现的行号。它们不是指在使用`firstnumber`进

行调整之前的原始或实际行号。

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`.

突出显示的颜色可以通过`highlightcolor`进行自定义。

The highlighting color can be customized with `highlightcolor`.

`keywordcase` (string) (default: `lower`)
更改关键字的大小写。取值为`lower`、`upper`或`capitalize`。

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

`label` (string) (default: *empty*)
在代码框的顶部、底部或两者都添加标签。有关更多信息和示例，请参见 fancyvrb 文档。*注意：*这不会在当前列表中添加 `\label`。要实现这一点，应使用浮动环境（第 4 节）。

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 4) instead.

`labelposition` (none | topline | bottomline | all) (default: `topline`, `all`, or `none`)
打印标签的位置（参见上文；默认值：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`）。有关更多信息，请参见 fancyvrb 文档。

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.

`lastline` (integer) (default: ⟨*last line of input*⟩)
要显示的最后一行。

The last line to be shown.

`linenos` (boolean) (default: `false`)

启用行号。为了自定义行号的显示样式，您需要重新定义 `\theFancyVerbLine` 宏：Enables line numbers. In order to customize the display style of line numbers, you need to redefine the `\theFancyVerbLine` macro:

```
\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
```

`numberfirstline` (boolean) (default: `false`)

始终对第一行进行编号， 而不考虑 `stepnumber`。

Always number the first line, regardless of `stepnumber`.

`numbers` (left | right | both | none) (default: `none`)

与 `linenos` 基本相同， 只不过可以指定行号显示在哪一侧。

Essentially the same as `linenos`, except the side on which the numbers appear may be specified.

`mathescape` (boolean) (default: `false`)

在注释中启用 LaTeX 数学模式。使用方式与 listings 宏包相同。关于数学模式和连字的注意事项， 请参见 `escapeinside` 下面的说明。

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

`numberblanklines` (boolean) (default: `true`)

启用或禁用空行的编号。

Enables or disables numbering of blank lines.

`numbersep` (dimension) (default: `12pt`)
行号与代码起始位置之间的间隔。

Gap between numbers and start of line.

`obeytabs` (boolean) (default: `false`)
将制表符视为制表符而不是将其转换为空格——也就是说，将其展开为由 `tabsize` 确定的制表符位置。**尽管这样可以正确展开缩进中的制表符，但通常无法正确展开除空格或制表符之外的任何内容之前的制表符。在这些情况下应该避免使用。**

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.**

`outencoding` (string) (default: ⟨*system-specific*⟩)
设置 Pygments 用于突出显示输出的文件编码。会覆盖之前通过 `encoding` 设置的任何编码。

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

`python3` (boolean) (default: `false`)
指定是否应用 Python 3 的语法高亮。

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

`resetmargins` (boolean) (default: `false`)
在其他环境中重置左边距。

Resets the left margin inside other environments.

`rulecolor` (color command) (default: `black`)
框架的颜色。

The color of the frame.

`samepage` (boolean) (default: `false`)
强制整个代码清单出现在同一页上，即使它不能适合一页。

Forces the whole listing to appear on the same page, even if it doesn't fit.

`showspaces` (boolean) (default: `false`)
启用可见空格：`visible␣spaces`。

Enables visible spaces: `visible␣spaces`.

`showtabs` (boolean) (default: `false`)
启用可见制表符——只能与 `obeytabs` 结合使用。

Enables visible tabs—only works in combination with `obeytabs`.

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

Redefine the visible space character. Note that this is only used if `showspaces=true`.

`spacecolor` (string) (default: `none`)
设置可见空格的颜色。默认情况下（`none`），它们采用其周围内容的颜色。

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

`startinline` (boolean) (default: `false`)
[For PHP only] 指定代码从 PHP 模式开始，即省略了前导 `<?php`。

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

`style` (string) (default: ⟨*default*⟩)
设置 Pygments 使用的样式表。

Sets the stylesheet used by Pygments.

`stepnumber` (integer) (default: `1`)
行号出现的间隔。

Interval at which line numbers appear.

`stepnumberfromfirst` (boolean) (default: `false`)

默认情况下，当使用行号并且 `stepnumber` $\neq 1$ 时，只包括 `stepnumber` 的倍数的行号。这会使行号从第一行偏移，因此第一行和它之后每隔 `stepnumber` 行的所有行都有行号。

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.

`stepnumberoffsetvalues` (boolean) (default: `false`)

默认情况下，当使用行号并且 `stepnumber` $\neq 1$ 时，只包括 `stepnumber` 的倍数的行号。使用 `firstnumber` 偏移行号将更改哪些行具有行号以及哪行有哪个号码，但不会更改出现的行号。此选项会导致忽略 `firstnumber` 来确定哪些行号是 `stepnumber` 的倍数。在计算实际显示的号码时仍然使用 `firstnumber`。因此，显示的行号将是 `stepnumber` 的倍数，再加上 `firstnumber` 减去 1。

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.

`stripall` (boolean) (default: `false`)

从输入中剥离所有前导和尾随空格。

Strip all leading and trailing whitespace from the input.

`stripnl` (boolean) (default: `false`)

从输入中剥离前导和尾随换行符。

Strip leading and trailing newlines from the input.

`tab` (macro) (default: fancyvrb's `\FancyVerbTab`, ⇥)

重新定义可见制表符字符。请注意，仅当 `showtabs=true` 时才会使用此选项。`\rightarrowfill`，$\longrightarrow$，可能是一个不错的替代方案。

Redefine the visible tab character. Note that this is only used if `showtabs=true`. `\rightarrowfill`, $\longrightarrow$, may be a nice alternative.

`tabcolor` (string) (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。对于缩进多行注释或字符串的制表符，通常不希望出现这种情况。

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.

`tabsize` (integer) (default: `8`)
制表符相当于的空格数。如果未激活 `obeytabs`，则制表符将转换为此数量的空格。如果激活了 `obeytabs`，则制表符停靠点之间将设置该数量的空格字符。

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.

`texcl` (boolean) (default: `false`)
启用注释中的 LaTeX 代码。使用方式类似于 listings 宏包。请参见 `escapeinside` 下关于数学和连字的注释。

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

`texcomments` (boolean) (default: `false`)
启用注释中的 LaTeX 代码。`texcl` 的较新名称。请参见 `escapeinside` 下关于数学和连字的注释。

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

关于 `texcomments`，从 Pygments 2.0.2 开始，它在处理多行 C/C++ 预处理器指令时失败，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标记为`Comment.Preproc`，因此 `texcomments` 会将预处理器指令视为文字 LaTeX 代码。在 Pygments 网站上已经有一个问题；关于minted GitHub 网

站还有其他详细信息。

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.

`xleftmargin` (dimension) (default: `0`)
列表之前要添加的缩进。

Indentation to add before the listing.

`xrightmargin` (dimension) (default: `0`)
列表之后要添加的缩进。

Indentation to add after the listing.

6 Defining shortcuts 定义快捷方式

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.

对于有大量源码列表的大型文档，尽管使用相同的源语言和相同的选项集，但仍然需要为大多数源码列表指定所有选项，这是冗余的，需要输入很多内容，并且使更改变得困难。

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

一种选项是使用`\setminted`，但即使如此，您仍然必须每次指定语言。

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

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

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

`\newminted`定义了一个`minted`环境的新别名：

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

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

```cpp
template <typename T>
T id(T value) {
    return value;
}
```

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

如果您想要在操作过程中提供额外的选项，或覆盖默认的选项设置，您也可以这样做：

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

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

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

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.

请注意环境名称后面的星号"`*`"，由于 fancyvrb 处理选项的限制，需要提供一个*单独的*接受选项的环境，而这些选项在星号版本的环境中*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:

环境的默认名称为⟨*language*⟩`code`。如果此名称与另一个环境冲突，或者出于其他原因想要选择自己的名称，可以将其指定为第一个参数：

`\newminted[`⟨*environment name*⟩`]{`⟨*language*⟩`}{`⟨*options*⟩`}`。

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 环境在内部使用 fancyvrb，因此基于它们的任何环境都必须包括 fancyvrb 命令`\VerbatimEnvironment`。这使得 fancyvrb 能够确定正在定义的环境的名称，并正确找到其结尾。最好在定义的开头包含此命令。例如，

```
\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.

上面的宏仅为`minted`环境定义了快捷方式。主要原因是`mint`的短命令形式通常需要不同的选项——至少，它通常不使用`gobble`选项。使用`\newmint[`⟨*宏名*⟩`]{`⟨*语言*⟩`}{`⟨*选项*⟩`}`定义了一个`\mint`的快捷方式。参数和用法与`\newminted`相同。如果没有指定⟨*macro name*⟩，则使用⟨*language*⟩。

```
\newmint{perl}{showspaces}
                                        my␣$foo␣=␣$bar;
\perl/my $foo = $bar;/
```

`\newmintinline` This creates custom versions of `\mintinline`. The syntax is the same as that for `\newmint`:

这将创建`\mintinline` 的自定义版本。其语法与`\newmint` 的语法相同：

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

If a ⟨*macro name*⟩ is not specified, then the created macro is called `\`⟨*language*⟩`inline`.

如果未指定⟨*macro name*⟩，则创建的宏将被称为`\`⟨*language*⟩`inline`。

```
\newmintinline{perl}{showspaces}
                                        Xmy␣$foo␣=␣$bar;X
X\perlinline/my $foo = $bar;/X
```

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

这将创建自定义版本的`\inputminted`。语法为

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

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

如果没有给出⟨*macro name*⟩，则该宏被称为⟨*language*⟩`file`。

7 FAQ and Troubleshooting 常见问题与故障排除

在某些情况下，minted 可能由于其他文档设置而无法达到所期望的效果，这些设置是它无法控制的。下面列出了常见问题及解决方法或解决方案。如果您正在使用 minted 进行非典型上下文的工作，您也可以搜索 tex.stackexchange.com 或在那里提问。

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.

- **有时会出现“无法写入文件”的错误。**这可能是由于在与 Dropbox 或类似的文件同步程序同步的目录中使用 minted 引起的。这些程序可能在它

仍需要修改它们的时候尝试同步 minted 的临时文件。解决方案是关闭文件同步或使用未同步的目录。

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.

- **我收到 "Font Warning: Some font shapes were not available" 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 默认替换的字体形状是完全足够的，可以忽略警告。在其他情况下，字体替换可能不清楚地指示粗体或斜体文本，您需要切换到不同的字体。请参见 LaTeX 字体目录中关于打字机字体的部分，了解替代方案。如果您喜欢默认的 LaTeX 字体，则 lmodern 包是一个很好的起点。beramono 和 courier 包也是不错的选择。

  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.

- **在使用缓存时，在 OS X 下出现 "Too many open files" 错误。**请参见第 3.1 节中关于 OS X 的说明。

  I receive a "Too many open files" error under OS X when using caching. See the note on OS X under Section 3.1.

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

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

- **当我使用 fancybox 宏包时，会出现一些奇怪的事情。**fancybox 与 minted 内部使用的 fancyvrb 冲突。在使用 fancybox 时，请确保在加载 minted

（或在未由 minted 加载 fancyvrb 的情况下，在加载 fancyvrb 之前）之前加载。

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

- **当我在使用 KOMA-Script 文档类时,使用 minted 会提示关于`\float@addtolists`的警告。** minted 使用 float 宏包生成浮动列表，但这与 KOMA-Script 生成浮动对象的方式冲突。解决方法是加载 scrhack 宏包或使用 minted 的`newfloat`选项。

  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 字符（~）被抬高,几乎像上标。**这是字体的问题。你需要使用不同的字体编码,可能是使用不同的字体。尝试使用`\usepackage[T1]{fontenc}`,也许搭配`\usepackage{lmodern}`或类似的宏包。

  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.

- **使用数学公式时出现错误,比如出现 `TeX capacity exceeded` 和`\leavevmode\kern\z@`。**这是由于公式内容中禁用了连字号造成的。请查看`escapeinside`下的注意事项。

  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`.

- **在使用 breqn 等特殊数学宏包时，使用 `mathescape` 会导致文档无法完成编译，或出现其他意外结果。**一些数学宏包（如 breqn）会给逗号等字符赋予数学模式下的特殊含义，这可能与 minted 发生冲突。在 breqn 和逗号的情况下，可以通过在`minted`环境中重新定义逗号来解决问题：

  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}
```

其他宏包或特殊字符可能需要自己的修改。

Other packages/special characters may need their own modifications.

- **在使用 Beamer 时出现错误。**由于 Beamer 的处理方式,你可能需要在包含 minted 命令和环境的幻灯片上使用`fragile`或`fragile=singleslide`选项。`fragile=singleslide`效果最好，但它禁用了叠加。`fragile`通过将每个幻灯片的内容保存到临时文件中再重复使用，这种方法允许使用叠加，但如果你在一行的开头有字符串`\end{frame}`（例如在`minted`环境中），它将会出错。为解决这个问题，可以将环境的内容缩进（以便`\end{frame}`前面有一个或多个空格）,然后使用`gobble`或`autogobble`选项来删除缩进。
  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.

- **Beamer 会吞掉制表符。**这是由于Beamer 在处理抄录内容时存在错误。请升级 Beamer 或使用链接的补丁。否则，如果您不需要覆盖图层，请尝试使用`fragile=singleslide`，或考虑使用`\inputminted`，或将制表符转换为空格。
  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 need overlays, or consider using `\inputminted` or converting the tabs into spaces.

- **我正在尝试创建几个新的 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。但它失败了。**这是由于 keyval 的工作方式（minted 使用它来管理选项）所致。参数不会被扩展。有关更多信息，请参见此处和此处。仍然可以实现您想要的功能；您只需在传递给创建新命令/环境的命令之前扩展选项宏即可。下面是一个示例。`\expandafter`是至关重要的部分。
  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}
```

- **我想在通常不允许使用抄录内容的上下文中使用`\mintinline`。**`\mintinline`命令已经可以在许多不允许使用普通抄录命令（如`\verb`）的地方使用，因此请先尝试。如果不起作用，则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，
  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}}
```

- **在 minted 命令和环境内无法使用扩展字符，即使使用了 inputenc 宏包也不行。**版本 2.0 增加了对 pdfTeX 引擎下扩展字符的支持。但是，如果需要使用 inputenc 不支持的字符，则应使用 XeTeX 或 LuaTeX 引擎。
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.

- **polyglossia 宏包对代码进行了一些不必要的更改。（例如，在法语中在冒号周围添加额外的空格。）**您可能需要将代码放在`\begin{english}...\end{english}`中。这可以在导言区使用 etoolbox 为所有`minted`环境完成：
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 `\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}}
```

- **制表符被转换为字符序列 `^^I`**。这发生在使用 XeLaTeX 时。您需要使用`-8bit`命令行选项，以便制表符可以正确地写入临时文件。有关 XeLaTeX 处理制表符的更多信息，请参见`http://tex.stackexchange.com/questions/58732/how-to-output-a-tabulation-into-a-file`。
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-fil` for more on XeLaTeX's handling of tab characters.

- **caption 宏包在与 minted 一起使用`\captionof` 和其他命令时会产生错误。**使用选项`compatibility=false`加载 caption 宏包。或者更好的方法

是使用 minted 的`newfloat`包选项，这提供了更好的 caption 兼容性。

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.

- **我需要一个支持分页的列表环境。**内置的列表环境是标准浮动体，不支持分页。您可能需要为长浮动体定义一个新环境。例如，

  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}}{}
  ```

  使用 caption 宏包时，最好使用 minted 的`newfloat`包选项。有关带有分页的`listing`环境的更多信息，请参见http://tex.stackexchange.com/a/53540/10742。

  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.

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

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

  ```
  \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的位置。

  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 automatically.

- **我想在帧标签中使用扩展字符,但是出现了错误。**这可能是由于 minted<2.0 和 Python 2.7 中的Pygments 终端编码问题导致的。它应该可以与任何版本的 Python 与 minted 2.0+ 一起使用，因为它会在内部处理标签并将其发送到 Python。

  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` 环境在 `tabular` 内部具有额外的垂直空间。**可以创建自定义环境来消除额外的空间。但是，在存在相邻文本的情况下表现符合预期的一般解决方案仍有待发现。

  `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.

- **我收到来自 `lineno.sty` 的警告，提示 "Command `@parboxrestore` has changed."** 当在 csquotes 之后加载 minted 时，可能会发生这种情况。尝试先加载 minted。如果您在不使用 csquotes 时收到此消息，则可能需要尝试加载包的顺序，并可能还要打开问题。

  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.

- **我正在使用 texi2pdf，并且从 tar 收到"Cannot stat"错误：**这是由于 texi2pdf 处理临时文件的方式。minted 会自动清理其临时文件，但是 texi2pdf 会认为在运行结束时任何曾经创建的临时文件仍然存在，因此尝试访问 minted 已删除的文件。可以通过在`\usepackage{minted}`之后添加`\renewcommand{\DeleteFile}[2][]{}`来禁用 minted 的临时文件清理。

  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}`.

Acknowledgements
致谢

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

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.
- 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.

Version History

v2.6 (2021/12/24)

- `autogobble` automatically uses `python` or `python3` executables, depending on availability, instead of requiring `python`. A custom executable can be specified by redefining `\MintedPython` (#277, #287).
- Fixed `autogobble` compatibility with `fancyvrb` 4.0+ (#315, #316).

- Pygments style names may now contain arbitrary non-whitespace characters. Previously, style names containing digits and some punctuation characters were incompatible (#210, #294, #299, #317). Pygments macros are now only defined just before use locally within `minted` commands and environments, rather than globally. Pygments macros now always use a `\PYG` prefix regardless of style, rather than a prefix of the form `\PYG<style>` (for example, what was previously `\PYGdefault` is now simply `\PYG`).
- Removed Python-based MD5 hashing for XeTeX, which was necessary before XeTeX added `\mdfivesum` in 2017.
- The default for `stripnl` is now `false`, so that original code is preserved exactly by default (#198).
- Added support for `fontencoding` option from `fvextra` (#208).
- Added note to FAQ about getting `texi2pdf` to work with `minted` given `texi2pdf`'s assumptions about temp files (#186).
- Reimplemented `bgcolor` option to be compatible with `color` package.

v2.5 (2017/07/19)

- The default placement for the `listing` float is now `tbp` instead of `h`, to parallel `figure` and `table` and also avoid warnings caused by `h` (#165). The documentation now contains information about changing default placement. The `float` package is no longer loaded when the `newfloat` package option is used.
- Added support for `*nchars` options from `fvextra` v1.3 that allow setting `breaklines`-related indentation in terms of a number of characters, rather than as a fixed dimension.
- Fixed incompatibility with `babel magyar` (#158).
- Added support for `beamer` overlays with `beameroverlays` option (#155).
- Comments in the Pygments LaTeX style files no longer appear as literal text when `minted` is used in `.dtx` files (#161).
- `autogobble` now works with package option `kpsewhich` (#151). Under Windows, the `kpsewhich` option no longer requires PowerShell.

- Fixed a bug that prevented `finalizecache` from working with `outputdir` (#149).
- Fixed a bug with `firstline` and `lastline` that prevented them from working with the `minted` environment (#145).
- Added note on `breqn` conflicts to FAQ (#163).

v2.4.1 (2016/10/31)

- Single quotation marks in `\jobname` are now replaced with underscores in `\minted@jobname` to prevent quoting errors (#137).
- The `autogobble` option now takes `firstline` and `lastline` into account (#130).
- Fixed `numberblanklines`, which had been lost in the transition to v2.0+ (#135).

v2.4 (2016/07/20)

- Line breaking and all associated options are now completely delegated to `fvextra`.
- Fixed a bug from v2.2 that could cause the first command or environment to vanish when `cache=false` (related to work on `\MintedPygmentize`).

v2.3 (2016/07/14)

- The `fvextra` package is now required. `fvextra` extends and patches `fancyvrb`, and includes improved versions of `fancyvrb` extensions that were formerly in `minted`.
- As part of `fvextra`, the `upquote` package is always loaded. `fvextra` brings the new option `curlyquotes`, which allows curly single quotation marks instead of the literal backtick and typewriter single quotation mark produced by `upquote`. This allows the default `upquote` behavior to be disabled when desired.
- Thanks to `fvextra`, the options `breakbefore`, `breakafter`, and `breakanywhere` are now compatible with non-ASCII characters under pdfTeX (#123).
- Thanks to `fvextra`, `obeytabs` no longer causes lines in multi-line comments or strings to vanish (#88), and is now compatible with

`breaklines` (#99). `obeytabs` will now always give correct results with tabs used for indentation. However, tab stops are not guaranteed to be correct for tabs in the midst of text.

- `fvextra` brings the new options `space`, `spacecolor`, `tab`, and `tabcolor` that allow these characters and their colors to be redefined (#98). The tab may now be redefined to a flexible-width character such as `\rightarrowfill`. The visible tab will now always be black by default, instead of changing colors depending on whether it is part of indentation for a multiline string or comment.
- `fvextra` brings the new options `highlightcolor` and `highlightlines`, which allow single lines or ranges of lines to be highlighted based on line number (#124).
- `fvextra` brings the new options `numberfirstline`, `stepnumberfromfirst`, and `stepnumberoffsetvalues` that provide better control over line numbering when `stepnumber` is not 1.
- Fixed a bug from v2.2.2 that prevented `upquote` from working.

v2.2.2 (2016/06/21)

- Fixed a bug introduced in v2.2 that prevented setting the Pygments style in the preamble. Style definitions are now more compatible with using `\MintedPygmentize` to call a custom `pygmentize`.

v2.2.1 (2016/06/15)

- The `shellesc` package is loaded before `ifplatform` and other packages that might invoke `\write18` (#112).
- When caching is enabled, XeTeX uses the new `\mdfivesum` macro from TeX Live 2016 to hash cache content, rather than using `\ShellEscape` with Python to perform hashing.

v2.2 (2016/06/08)

- All uses of `\ShellEscape` (`\write18`) no longer wrap file names and paths with double quotes. This allows a cache directory to be specified relative to a user's home directory, for example, `~/minted_cache`. `cachedir` and `outputdir` paths containing spaces will now require explicit quoting of the parts of the paths that contain spaces, since `minted` no longer supplies quoting. See the updated documentation for examples (#89).

- Added `breakbefore`, `breakbeforegroup`, `breakbeforesymbolpre`, and `breakbeforesymbolpost`. These parallel `breakafter*`. It is possible to use `breakbefore` and `breakafter` for the same character, so long as `breakbeforegroup` and `breakaftergroup` have the same setting (#117).
- Added package options `finalizecache` and `frozencache`. These allow the cache to be prepared for (`finalizecache`) and then used (`frozencache`) in an environment in which `-shell-escape`, Python, and/or Pygments are not available. Note that this only works if `minted` content does not need to be modified, and if no settings that depend on Pygments or Python need to be changed (#113).
- Style names containing hyphens and underscores (`paraiso-light`, `paraiso-dark`, `algol_nu`) now work (#111).
- The `shellesc` package is now loaded, when available, for compatibility with LuaTeX 0.87+ (TeX Live 2016+, etc.). `\ShellEscape` is now used everywhere instead of `\immediate\write18`. If `shellesc` is not available, then a `\ShellEscape` macro is created. When `shellesc` is loaded, there is a check for versions before v0.01c to patch a bug in v0.01b (present in TeX Live 2015) (#112).
- The `bgcolor` option now uses the `snugshade*` environment from the `framed` package, so `bgcolor` is now compatible with page breaks. When `bgcolor` is in use, immediately preceding text will no longer push the `minted` environment into the margin, and there is now adequate spacing from surrounding text (#121).
- Added missing support for `fancyvrb`'s `labelposition` (#102).
- Improved fix for TikZ externalization, thanks to Patrick Vogt (#73).
- Fixed `breakautoindent`; it was disabled in version 2.1 due to a bug in `breakanywhere`.
- Properly fixed handling of `\MintedPygmentize` (#62).
- Added note on incompatibility of `breaklines` and `obeytabs` options. Trying to use these together will now result in a package

error (#99). Added note on issues with `obeytabs` and multiline comments (#88). Due to the various `obeytabs` issues, the docs now discourage using `obeytabs`.

- Added note to FAQ on `fancybox` and `fancyvrb` conflict (#87).
- Added note to docs on the need for `\VerbatimEnvironment` in environment definitions based on `minted` environments.

v2.1 (2015/09/09)

- Changing the highlighting style now no longer involves re-highlighing code. Style may be changed with almost no overhead.
- Improved control of automatic line breaks. New option `breakanywhere` allows line breaks anywhere when `breaklines=true`. The pre-break and post-break symbols for these types of breaks may be set with `breakanywheresymbolpre` and `breakanywheresymbolpost` (#79). New option `breakafter` allows specifying characters after which line breaks are allowed. Breaks between adjacent, identical characters may be controlled with `breakaftergroup`. The pre-break and post-break symbols for these types of breaks may be set with `breakaftersymbolpre` and `breakaftersymbolpost`.
- `breakbytoken` now only breaks lines between tokens that are separated by spaces, matching the documentation. The new option `breakbytokenanywhere` allows for breaking between tokens that are immediately adjacent. Fixed a bug in `\mintinline` that produced a following linebreak when `\mintinline` was the first thing in a paragraph and `breakbytoken` was true (#77).
- Fixed a bug in draft mode option handling for `\inputminted` (#75).
- Fixed a bug with `\MintedPygmentize` when a custom `pygmentize` was specified and there was no `pygmentize` on the default path (#62).
- Added note to docs on caching large numbers of code blocks under OS X (#78).
- Added discussion of current limitations of `texcomments` (#66) and `escapeinside` (#70).
- PGF/Ti*k*Z externalization is automatically detected and supported (#73).

- The package is now compatible with LaTeX files whose names contain spaces (#85).

v2.0 (2015/01/31)

- Added the compatibility package `minted1`, which provides the `minted` 1.7 code. This may be loaded when 1.7 compatibility is required. This package works with other packages that `\RequirePackage{minted}`, so long as it is loaded first.
- Moved all old `\changes` into `changelog`.

Development releases for 2.0 (2014–January 2015)

- Caching is now on by default.
- Fixed a bug that prevented compiling under Windows when file names contained commas.
- Added `breaksymbolleft`, `breaksymbolsepleft`, `breaksymbolindentleft`, `breaksymbolright`, `breaksymbolsepright`, and `breaksymbolindentright` options. `breaksymbol`, `breaksymbolsep`, and `breaksymbolindent` are now aliases for the correspondent `*left` options.
- Added `kpsewhich` package option. This uses `kpsewhich` to locate the files that are to be highlighted. This provides compatibility with build tools like `texi2pdf` that function by modifying `TEXINPUTS` (#25).
- Fixed a bug that prevented `\inputminted` from working with `outputdir`.
- Added informative error messages when Pygments output is missing.
- Added `final` package option (opposite of `draft`).
- Renamed the default cache directory to `_minted-<jobname>` (replaced leading period with underscore). The leading period caused the cache directory to be hidden on many systems, which was a potential source of confusion.
- `breaklines` and `breakbytoken` now work with `\mintinline` (#31).
- `bgcolor` may now be set through `\setminted` and `\setmintedinline`.

- When math is enabled via `texcomments`, `mathescape`, or `escapeinside`, space characters now behave as in normal math by vanishing, instead of appearing as literal spaces. Math need no longer be specially formatted to avoid undesired spaces.
- In default value of `\listoflistingscaption`, capitalized "Listings" so that capitalization is consistent with default values for other lists (figures, tables, algorithms, etc.).
- Added `newfloat` package option that creates the `listing` environment using `newfloat` rather than `float`, thus providing better compatibility with the `caption` package (#12).
- Added support for Pygments option `stripall`.
- Added `breakbytoken` option that prevents `breaklines` from breaking lines within Pygments tokens.
- `\mintinline` uses a `\colorbox` when `bgcolor` is set, to give more reasonable behavior (#57).
- For PHP, `\mintinline` automatically begins with `startinline=true` (#23).
- Fixed a bug that threw off line numbering in `minted` when `langlinenos=false` and `firstnumber=last`. Fixed a bug in `\mintinline` that threw off subsequent line numbering when `langlinenos=false` and `firstnumber=last`.
- Improved behavior of `\mint` and `\mintinline` in `draft` mode.
- The `\mint` command now has the additional capability to take code delimited by paired curly braces `{}`.
- It is now possible to set options only for `\mintinline` using the new `\setmintedinline` command. Inline options override options specified via `\setminted`.
- Completely rewrote option handling. fancyvrb options are now handled on the LaTeX side directly, rather than being passed to Pygments and then returned. This makes caching more efficient, since code is no longer rehighlighted just because fancyvrb options changed.
- Fixed buffer size error caused by using `cache` with a very large number of files (#61).

- Fixed `autogobble` bug that caused failure under some operating systems.
- Added support for `escapeinside` (requires Pygments 2.0+; #38).
- Fixed issues with XeTeX and caching (#40).
- The `upquote` package now works correctly with single quotes when using Pygments 1.6+ (#34).
- Fixed caching incompatibility with Linux and OS X under xelatex (#18 and #42).
- Fixed `autogobble` incompatibility with Linux and OS X.
- `\mintinline` and derived commands are now robust, via `\newrobustcmd` from `etoolbox`.
- Unused styles are now cleaned up when caching.
- Fixed a bug that could interfere with caching (#24).
- Added `draft` package option (#39). This typesets all code using `fancyvrb`; Pygments is not used. This trades syntax highlighting for maximum speed in compiling.
- Added automatic line breaking with `breaklines` and related options (#1).
- Fixed a bug with boolean options that needed a False argument to cooperate with `\setminted` (#48).

v2.0-alpha3 (2013/12/21)

- Added `autogobble` option. This sends code through Python's `textwrap.dedent()` to remove common leading whitespace.
- Added package option `cachedir`. This allows the directory in which cached content is saved to be specified.
- Added package option `outputdir`. This allows an output directory for temporary files to be specified, so that the package can work with LaTeX's `-output-directory` command-line option.
- The `kvoptions` package is now required. It is needed to process key-value package options, such as the new `cachedir` option.
- Many small improvements, including better handling of paths under Windows and improved key system.

v2.0-alpha2 (2013/08/21)

- `\DeleteFile` now only deletes files if they do indeed exist. This eliminates warning messages due to missing files.
- Fixed a bug in the definition of `\DeleteFile` for non-Windows systems.
- Added support for Pygments option `stripnl`.
- Settings macros that were previously defined globally are now defined locally, so that `\setminted` may be confined by `\begingroup...\endgroup` as expected.
- Macro definitions for a given style are now loaded only once per document, rather than once per command/environment. This works even without caching.
- A custom script/executable may now be substituted for `pygmentize` by redefining `\MintedPygmentize`.

v2.0alpha (2013/07/30)

- Added the package option `cache`. This significantly increases compilation speed by caching old output. For example, compiling the documentation is around 5x faster.
- New inline command `\mintinline`. Custom versions can be created via `\newmintinline`. The command works inside other commands (for example, footnotes) in most situations, so long as the percent and hash characters are avoided.
- The new `\setminted` command allows options to be specified at the document and language levels.
- All extended characters (Unicode, etc.) supported by `inputenc` now work under the pdfTeX engine. This involved using `\detokenize` on everything prior to saving.
- New package option `langlinenos` allows line numbering to pick up where it left off for a given language when `firstnumber=last`.
- New options, including `style`, `encoding`, `outencoding`, `codetagify`, `keywordcase`, `texcomments` (same as `texcl`), `python3` (for the `PythonConsoleLexer`), and `numbers`.

- `\usemintedstyle` now takes an optional argument to specify the style for a particular language, and works anywhere in the document.
- `xcolor` is only loaded if `color` isn't, preventing potential package clashes.

1.7 (2011/09/17)

- Options for float placement added [2011/09/12]
- Fixed `tabsize` option [2011/08/30]
- More robust detection of the `-shell-escape` option [2011/01/21]
- Added the `label` option [2011/01/04]
- Installation instructions added [2010/03/16]
- Minimal working example added [2010/03/16]
- Added PHP-specific options [2010/03/14]
- Removed unportable flag from Unix shell command [2010/02/16]

1.6 (2010/01/31)

- Added font-related options [2010/01/27]
- Windows support added [2010/01/27]
- Added command shortcuts [2010/01/22]
- Simpler versioning scheme [2010/01/22]

0.1.5 (2010/01/13)

- Added `fillcolor` option [2010/01/10]
- Added float support [2010/01/10]
- Fixed `firstnumber` option [2010/01/10]
- Removed `caption` option [2010/01/10]

0.0.4 (2010/01/08)

- Initial version [2010/01/08]