LaTeX/Source Code Listings

Contents

Using the *listings* package

Using the package `listings` you can add non-formatted text as you would do with `\begin{verbatim}` but its main aim is to include the source code of any programming language within your document. If you wish to include pseudocode or algorithms, you may find Algorithms and Pseudocode useful also.

To use the package, you need:

```
\usepackage{listings}
```

The `listings` package supports highlighting of all the most common languages and it is highly customizable. If you just want to write code within your document the package provides the `lstlisting` environment:

```
\begin{lstlisting}
```

```
Put your code here.
\end{lstlisting}
```

Another possibility, that is very useful if you created a program on several files and you are still editing it, is to import the code from the source itself. This way, if you modify the source, you just have to recompile the LaTeX code and your document will be updated. The command is:

```
\lstinputlisting{source_filename.py}
```

in the example there is a Python source, but it doesn't matter: you can include any file but you have to write the full file name. It will be considered plain text and it will be highlighted according to your settings, that means it doesn't recognize the programming language by itself. You can specify the language while including the file with the following command:

```
\lstinputlisting[language=Python]{source_filename.py}
```

You can also specify a scope for the file.

```
\lstinputlisting[language=Python, firstline=37, lastline=45]{source_filename.py}
```

This comes in handy if you are sure that the file will not change (at least before the specified lines). You may also omit the `firstline` or `lastline` parameter: it means *everything up to or starting from this point*.

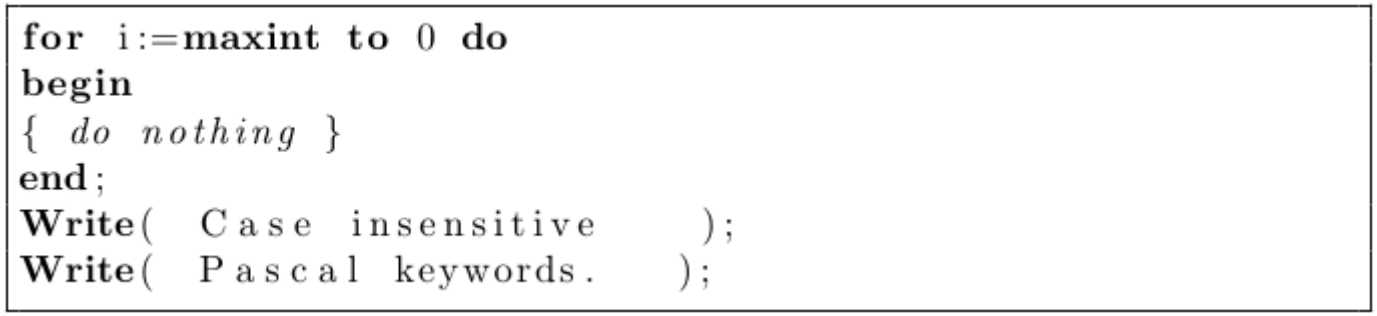
This is a basic example for some Pascal code:

```
\documentclass{article}
\usepackage{listings}             % Include the listings-package
\begin{document}
\lstset{language=Pascal}          % Set your language (you can change the language for each code-block optionally)

\begin{lstlisting}[frame=single]  % Start your code-block
```

```
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write('Case insensitive ');
Write('Pascal keywords.');
\end{lstlisting}

\end{document}
```



Supported languages

It supports the following programming languages:

ABAP[2,4], ACSL, Ada[4], Algol[4], Ant, Assembler[2,4], Awk[4], bash, Basic[2,4], C#[5], C++[4], C[4], Caml[4], Clean, Cobol[4], Comal, csh, Delphi, Eiffel, Elan, erlang, Euphoria, Fortran[4], GCL, Gnuplot, Haskell, HTML, IDL[4], inform, Java[4], JVMIS, ksh, Lisp[4], Logo, Lua[2], make[4], Mathematica[1,4], Matlab, Mercury, MetaPost, Miranda, Mizar, ML, Modelica[3], Modula-2, MuPAD, NASTRAN, Oberon-2, Objective C[5] , OCL[4], Octave, Oz, Pascal[4], Perl, PHP, PL/I, Plasm, POV, Prolog, Promela, Python, R, Reduce, Rexx, RSL, Ruby, S[4], SAS, Scilab, sh, SHELXL, Simula[4], SQL, tcl[4], TeX[4], VBScript, Verilog, VHDL[4], VRML[4], XML, XSLT.

For some of them, several dialects are supported. For more information, refer to the documentation that comes with the package, it should be within your distribution under the name `listings-*.dvi`.

Notes

1. It supports Mathematica code only if you are typing in plain text format. You can't include *.NB files `\lstinputlisting{...}` as you could with any other programming language, but Mathematica can export in a pretty-formatted LaTeX source.
2. Specification of the dialect is mandatory for these languages (e.g. `language={[x86masm]Assembler}`).
3. Modelica is supported via the dtsyntax package available here (https://code.google.com/p/dtsyntax/).
4. For these languages, multiple dialects are supported. C, for example, has ANSI, Handel, Objective and Sharp. See p. 12 of the listings manual (http://mirrors.ctan.org/macros/latex/contrib/listings/listings.pdf) for an overview.

5. Defined as a dialect of another language

Settings

You can modify several parameters that will affect how the code is shown. You can put the following code anywhere in the document (it doesn't matter whether before or after `\begin{document}`), change it according to your needs. The meaning is explained next to any line.

```
\usepackage{listings}
\usepackage{color}

\definecolor{mygreen}{rgb}{0,0.6,0}
\definecolor{mygray}{rgb}{0.5,0.5,0.5}
\definecolor{mymauve}{rgb}{0.58,0,0.82}

\lstset{ %
  backgroundcolor=\color{white},   % choose the background color; you must add \usepackage{color} or \usepackage{xcolor}; should come as last argument
  basicstyle=\footnotesize,        % the size of the fonts that are used for the code
  breakatwhitespace=false,         % sets if automatic breaks should only happen at whitespace
  breaklines=true,                 % sets automatic line breaking
  captionpos=b,                    % sets the caption-position to bottom
  commentstyle=\color{mygreen},    % comment style
  deletekeywords={...},            % if you want to delete keywords from the given language
  escapeinside={\%*}{*)},          % if you want to add LaTeX within your code
  extendedchars=true,              % lets you use non-ASCII characters; for 8-bits encodings only, does not work with UTF-8
  frame=single,                    % adds a frame around the code
  keepspaces=true,                 % keeps spaces in text, useful for keeping indentation of code (possibly needs columns=flexible)
  keywordstyle=\color{blue},       % keyword style
  language=Octave,                 % the language of the code
  morekeywords={*,...},            % if you want to add more keywords to the set
  numbers=left,                    % where to put the line-numbers; possible values are (none, left, right)
  numbersep=5pt,                   % how far the line-numbers are from the code
  numberstyle=\tiny\color{mygray}, % the style that is used for the line-numbers
  rulecolor=\color{black},         % if not set, the frame-color may be changed on line-breaks within not-black text (e.g. comments (green here))
  showspaces=false,                % show spaces everywhere adding particular underscores; it overrides 'showstringspaces'
  showstringspaces=false,          % underline spaces within strings only
  showtabs=false,                  % show tabs within strings adding particular underscores
  stepnumber=2,                    % the step between two line-numbers. If it's 1, each line will be numbered
  stringstyle=\color{mymauve},     % string literal style
  tabsize=2,                       % sets default tabsize to 2 spaces
  title=\lstname                   % show the filename of files included with \lstinputlisting; also try caption instead of title
}
```

escapeinside

The `escapeinside` line needs an explanation. The option `escapeinside={A}{B}` will define delimiters for escaping into LaTeX code, *i.e.* all the code between the string "A" and "B" will be parsed as LaTeX over the current *listings* style. In the example above, the comments for *Octave* start with `%`, and they are going to be printed in the document unless they start with `%*`, in which case they are read as LaTeX (with all LaTeX commands fulfilled) until they're closed with another `*)`.

If you add the above paragraph, the following can be used to alter the settings within the code:

```
\lstset{language=C,caption={Descriptive Caption Text},label=DescriptiveLabel}
```

There are many more options, check the official documentation.

Style definition

The package lets you define styles, *i.e.* profiles specifying a set of settings.

Example

```
\lstdefinestyle{customc}{
  belowcaptionskip=1\baselineskip,
  breaklines=true,
  frame=L,
  xleftmargin=\parindent,
  language=C,
  showstringspaces=false,
  basicstyle=\footnotesize\ttfamily,
  keywordstyle=\bfseries\color{green!40!black},
  commentstyle=\itshape\color{purple!40!black},
  identifierstyle=\color{blue},
  stringstyle=\color{orange},
}

\lstdefinestyle{customasm}{
  belowcaptionskip=1\baselineskip,
  frame=L,
  xleftmargin=\parindent,
  language=[x86masm]Assembler,
  basicstyle=\footnotesize\ttfamily,
  commentstyle=\itshape\color{purple!40!black},
}

\lstset{escapechar=@,style=customc}
```

In our example, we only set two options globally: the default style and the escape character. Usage:

```
\begin{lstlisting}
#include <stdio.h>
```

```
#define N 10
/* Block
 * comment */

int main()
{
    int i;

    // Line comment.
    puts("Hello world!");

    for (i = 0; i < N; i++)
    {
        puts("LaTeX is also great for programmers!");
    }

    return 0;
}
\end{lstlisting}

\lstinputlisting[caption=Scheduler, style=customc]{hello.c}
```

The C part will print as

```
#include <stdio.h>
#define N 10
/* Block
 * comment */

int main()
{
    int i;

    // Line comment.
    puts("Hello world!");

    for (i = 0; i < N; i++)
    {
        puts("LaTeX is also great for programmers!");
    }

    return 0;
}
```

Automating file inclusion

If you have a bunch of source files you want to include, you may find yourself doing the same thing over and over again. This is where macros show their real power.

```
\newcommand{\includecode}[2][c]{\lstinputlisting[caption=#2, escapechar=, style=custom#1]{#2}<!---->}
% ...

\includecode{sched.c}
\includecode[asm]{sched.s}
% ...

\lstlistoflistings
```

In this example, we create one command to ease source code inclusion. We set the default style to be *customc*. All listings will have their name as caption: we do not have to write the file name twice thanks to the macro. Finally we list all listings with this command from the `listings` package.

See Macros for more details.

Encoding issue

By default, `listings` does not support multi-byte encoding for source code. The `extendedchar` option only works for 8-bits encodings such as latin1.

To handle UTF-8, you should tell listings how to interpret the special characters by defining them like so

```
\lstset{literate=
  {á}{{\'a}}1 {é}{{\'e}}1 {í}{{\'i}}1 {ó}{{\'o}}1 {ú}{{\'u}}1
  {Á}{{\'A}}1 {É}{{\'E}}1 {Í}{{\'I}}1 {Ó}{{\'O}}1 {Ú}{{\'U}}1
  {à}{{\`a}}1 {è}{{\`e}}1 {ì}{{\`i}}1 {ò}{{\`o}}1 {ù}{{\`u}}1
  {À}{{\`A}}1 {È}{{\'E}}1 {Ì}{{\`I}}1 {Ò}{{\`O}}1 {Ù}{{\`U}}1
  {ä}{{\"a}}1 {ë}{{\"e}}1 {i}{{\"i}}1 {ö}{{\"o}}1 {ü}{{\"u}}1
  {Ä}{{\"A}}1 {Ë}{{\"E}}1 {Ï}{{\"I}}1 {Ö}{{\"O}}1 {Ü}{{\"U}}1
  {â}{{\^a}}1 {ê}{{\^e}}1 {î}{{\^i}}1 {ô}{{\^o}}1 {û}{{\^u}}1
  {Â}{{\^A}}1 {Ê}{{\^E}}1 {Î}{{\^I}}1 {Ô}{{\^O}}1 {Û}{{\^U}}1
  {œ}{{\oe}}1 {Œ}{{\OE}}1 {æ}{{\ae}}1 {Æ}{{\AE}}1 {ß}{{\ss}}1
  {ű}{{\H{u}}}1 {Ű}{{\H{U}}}1 {ő}{{\H{o}}}1 {Ő}{{\H{O}}}1
  {ç}{{\c c}}1 {Ç}{{\c C}}1 {ø}{{\o}}1 {å}{{\r a}}1 {Å}{{\r A}}1
  {€}{{\euro}}1 {£}{{\pounds}}1 {«}{{\guillemotleft}}1
  {»}{{\guillemotright}}1 {ñ}{{\~n}}1 {Ñ}{{\~N}}1 {¿}{{?`}}1
}
```

The above table will cover most characters in latin languages. For a more detailed explanation of the usage of the `literate` option check section 6.4 in the Listings Documentation (http://mirrors.ctan.org/macros/latex/contrib/listings/listings.pdf).

Another possibility is to replace `\usepackage{listings}` (in the preamble) with `\usepackage{listingsutf8}`, but this will only work for `\lstinputlisting{...}`.

Customizing captions

You can have fancy captions (or titles) for your listings using the `caption` package. Here is an example for `listings`.

```
\usepackage{caption}
\usepackage{listings}

\DeclareCaptionFont{white}{ \color{white} }
\DeclareCaptionFormat{listing}{
  \colorbox[cmyk]{0.43, 0.35, 0.35,0.01 }{
    \parbox{\textwidth}{\hspace{15pt}#1#2#3}
  }
}
\captionsetup[lstlisting]{ format=listing, labelfont=white, textfont=white, singlelinecheck=false, margin=0pt, font={bf,footnotesize} }

% ...

\lstinputlisting[caption=My caption]{sourcefile.lang}
```

The *minted* package

`minted` is an alternative to `listings` which has become popular. It uses the external Python library Pygments (http://pygments.org/) for code highlighting, which as of Nov 2014 boasts over 300 supported languages and text formats.

As the package relies on external Python code, the setup require a few more steps than a usual LaTeX package, so please have a look at their GitHub repo (https://github.com/gpoore/minted) and their manual (https://github.com/gpoore/minted/blob/master/source/minted.pdf).

References

A lot more detailed information can be found in a PDF by Carsten Heinz and Brooks Moses (http://mirror.hmc.edu/ctan/macros/latex/contrib/listings/listings.pdf).

Details and documentation about the Listings package can be found at its CTAN website (http://www.ctan.org/tex-archive/macros/latex/contrib/listings/).