

Verbatim Text and Code Listings in L^AT_EX

Raphael Frey, <webmaster@alpenwasser.net>

March 23, 2017

Abstract

There are many ways to integrate verbatim text (text which is presented in the resulting document as it is in the source code) and listings (which may include pretty printing, syntax highlighting and other fancy things) into a L^AT_EX document, as can easily be seen when looking at the corresponding topic pages on CTAN [10, 11]. This document presents examples for a few of them. The idea is not to provide in-depth documentation, but to give a brief overview of *a few* possibilities to enable the reader to get an idea of what's out there and what might, potentially, be useful to them. Consulting the respective user manual is always recommended for more detailed information.

For the *very* impatient: For most code listings, I use the `minted` package these days. While it is slow (at least on first compilation), it has served me well most of the time. But there are alternatives, and personal preferences matter, too. One's mileage may vary.

Contents

1	The verbatim Environment	2
1.1	Stock L ^A T _E X	2
1.2	Re-Implementation [1]	2
2	The verbatimbox Package [2]	3
3	The fancyvrb Package [3]	5
3.1	Footnotes	5
3.2	Inline Verbatim Content	5
3.3	Verbatim Environments	6
4	The listing Package [4]	8
5	The listings Package [5]	8
6	The listingsutf8 Package [6]	8
7	The mlppretty Package [7]	8
8	The minted Package [8]	8
9	The verbmnts Package [9]	8
10	References	8

1 The verbatim Environment

L^AT_EX provides a `verbatim` Environment which, although it has quite a few limitations, is simple to use and often gets the job done well enough. For fancier things, there exists a re-implementation with a few added niceties.

1.1 Stock L^AT_EX

The stock L^AT_EX `verbatim` environment is used like this:

Code	L ^A T _E X Output
<pre>\begin{verbatim} #include <stdio.h> main() { print("Hello, world!\n"); } \end{verbatim}</pre>	<pre>#include <stdio.h> main() { print("Hello, world!\n"); }</pre>

1.2 Re-Implementation [1]

The re-implementation alleviates some limitations of the stock `verbatim` environment. For more details and to evaluate whether these limitations may be of relevance to your use case, consult the documentation at [1]. Basic usage is identical to the stock `verbatim` environment.

Code	L ^A T _E X Output
<pre>\begin{verbatim} #include <stdio.h> main() { print("Hello, world!\n"); } \end{verbatim}</pre>	<pre>#include <stdio.h> main() { print("Hello, world!\n"); }</pre>

2 The verbatimbox Package [2]

A box is defined via `\begin{verbatimbox}[options] content \end{verbatimbox}`. This, however, does not yet display the box. Instead, it is stored for later use via the `\theverbatimbox` command. This can be easily wrapped inside an `\fbox`, optionally adding some vertical space with the `\addvbuffer` command.

Code	L ^A T _E X Output
<pre>\begin{verbatimbox}[\arabic{VerbbboxLineNo}:\hspace{1ex}] #include <stdio.h> main() { print("Hello, world!\n"); } \end{verbatimbox} \fbox{\addvbuffer[10pt 5pt]\theverbatimbox} % different spacing above and below \fbox{\addvbuffer[10pt]\theverbatimbox} % same spacing above and below</pre>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <pre>1: #include <stdio.h> 2: 3: main() { 4: print("Hello, world!\n"); 5: }</pre> </div> <div style="border: 1px solid black; padding: 5px;"> <pre>1: #include <stdio.h> 2: 3: main() { 4: print("Hello, world!\n"); 5: }</pre> </div>

Note the removal of empty lines in the result on the right. As above, check the documentation [2] for more options and possibilities. Also, the optional argument `[\arabic{VerbbboxLineNo}:\hspace{1ex}]` to `\begin{verbatimbox}` must be on the same line and only on that line.

For cases where the limitation of having only a single named box to which one can refer, `\theverbatimbox`, must be overcome, there exists the `\myverbatimbox` command, by which named boxes can be created for later use. One use case for this is the use of verbatim environments in tabular environments, where the use of `verbatim` is not allowed (example from the `verbatimbox` manual [2]).

Code	L ^A T _E X Output				
<pre>\begin{myverbatimbox}{\vtheta}\theta\end{myverbatimbox} \begin{myverbatimbox}{\valpha}\alpha\end{myverbatimbox} \begin{tabular}{ c c } \hline \valpha & \$\alpha\$ \\ \hline \vtheta & \$\theta\$ \\ \hline \end{tabular}</pre>	<table border="1" style="margin: auto; border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 5px;"><code>\alpha</code></td> <td style="padding: 5px;">α</td> </tr> <tr> <td style="padding: 5px;"><code>\theta</code></td> <td style="padding: 5px;">θ</td> </tr> </table>	<code>\alpha</code>	α	<code>\theta</code>	θ
<code>\alpha</code>	α				
<code>\theta</code>	θ				

`verbatimbox` also allows including the contents of an entire file:

Code	L ^A T _E X Output
<pre>\verbfilebox{code/hw.c} \theverbbox</pre>	<pre>#include <stdio.h> main() { print("Hello, world!\n"); }</pre>

Because `verbatimbox` uses boxes to save its contents, they cannot be broken across pages. For breaking verbatims across pages, there exist the `verbnobox` environment and the `\verbfilenobox` command. These do not allow recalling the verbatim contents at a later point though; their output is generated directly where they are placed in the source code.

Code	L ^A T _E X Output
<pre>\begin{verbnobox}[\arabic{VerbboxLineNo}:\hspace{1ex}] #include <stdio.h> main() { print("Hello, world!\n"); } \end{verbnobox}</pre>	
<pre>1: #include <stdio.h> 2: 3: main() { 4: print("Hello, world!\n"); 5: }</pre>	

Code	L ^A T _E X Output
<pre>\verbfilenobox{code/hw.c}</pre>	<pre>#include <stdio.h> main() { print("Hello, world!\n"); }</pre>

3 The fancyvrb Package [3]

`fancyvrb` offers several advancements over the standard \LaTeX verbatim environment. Some of them are:

- Footnotes can contain verbatim content.
- Several different verbatim environments are provided, and it is possible to create new environments.
- Verbatim content can be stored and recalled.
- Files can be read and written in verbatim mode.

Putting verbatim content in footnotes is accomplished by invoking the `\VerbatimFootnotes` command after the preamble at some point, after which verbatim content can be put inside footnotes:

3.1 Footnotes

After invoking `VerbatimFootnotes`, verbatim can be put into footnotes¹. Note that the verbatim content in the footnote must be on a single line (though the footnote text itself can break across multiple lines of source code.)

Code	\LaTeX Output
<pre>\VerbatimFootnotes And then we have some text\footnotemark with a footnote, and that footnote shall contain verbatim content. \footnotetext{\verb+_And here we are, down in the footnotes!!_+ This is outside the verbatim content in the footnote.}</pre>	

3.2 Inline Verbatim Content

`fancyvrb` allows convenient inlining of verbatim content via the `\DefineShortVerb`, the `\UnDefineShortVerb` and the `\Verb` commands²:

Code	\LaTeX Output
<pre>We can write \Verb+_inline verbatim_+ by defining delimiting characters ad hoc, or via: \DefineShortVerb{\ } And now we can use _this_ _delimiter_ around inline verbatim content, until its special meaning is revoked via \UnDefineShortVerb{\ } And now the delimiter no longer has any effect. A new one could also be defined.</pre>	<p>We can write <code>_inline verbatim_</code> by defining delimiting characters ad hoc, or via: And now we can use <code>_this_ _delimiter_</code> around inline verbatim content, until its special meaning is revoked via And now the <code> delimiter </code> no longer has any effect. A new one could also be defined.</p>

¹`_And here we are, down in the footnotes!!_` This is outside the verbatim content in the footnote.

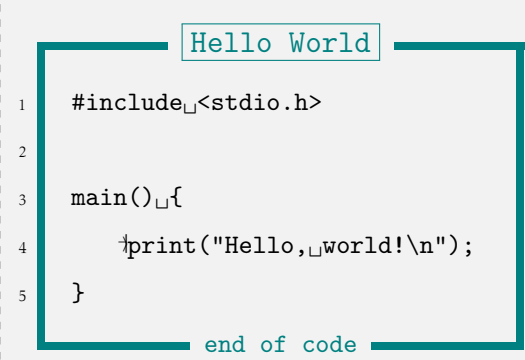
²Note that the underscores in this example are not necessary, but merely serve to emphasize that we are in verbatim mode, because in regular text mode it is a special character which would need to be escaped like so: `_.`

3.3 Verbatim Environments

`fancyvrb` provides several verbatim environments. The most straightforward replacement for the stock \LaTeX `verbatim` environment is the `Verbatim` environment³:

Code	\LaTeX Output
<pre>\begin{Verbatim} #include <stdio.h> main() { print("Hello, world!\n"); } \end{Verbatim}</pre>	<pre>#include <stdio.h> main() { print("Hello, world!\n"); }</pre>

An example with a lot of options (not all of which might be sensible):

Code	\LaTeX Output
<pre>\begin{Verbatim}[% gobble=4, numbers=left, numbersep=4pt, showspaces=true, showtabs=true, baselinestretch=1.5, tabsize=4, frame=single, framerule=1mm, framesep=1em, rulecolor=\color{teal}, label={% [\fbox{\Large{Hello World}}]}% end of code}%] #include <stdio.h> main() { print("Hello, world!\n"); } \end{Verbatim}</pre>	 <pre>1 #include <stdio.h> 2 3 main() { 4 print("Hello, world!\n"); 5 } end of code</pre>

We can also define custom environments:

³For a complete list of options, consult the user manual at `fancyvrb`.

Code	L ^A T _E X Output
<pre> \DefineVerbatimEnvironment% {CustomVerbatim}{Verbatim}{% gobble=4, numbers=left, numbersep=4pt, baselinestretch=1.1, frame=single, framerule=1mm, framesep=1em, rulecolor=\color{teal}} \begin{CustomVerbatim}[% label={% [\fbox{\Large{Hello World In C}}}% end of code}%] #include <stdio.h> main() { print("Hello, world!\n"); } \end{CustomVerbatim} \begin{CustomVerbatim}[% label={% [\fbox{\Large{Hello World in Perl}}}% end of code]} #!/usr/bin/env perl use strict; use warnings; use v5.10; say("Hello, world!"); \end{CustomVerbatim} </pre>	<pre> 1 #include <stdio.h> 2 3 main() { 4 print("Hello, world!\n"); 5 } 6 7 end of code </pre> <pre> 1 #!/usr/bin/env perl 2 use strict; 3 use warnings; 4 use v5.10; 5 6 say("Hello, world!"); 7 8 end of code </pre>

There is also the possibility to store and recall verbatim content, which, as in the case of `verbatimbox` package, allows to use verbatim content where it is otherwise not allowed, and to re-use it repeatedly in a convenient way. Additionally, content can be written to and read from external files. Consult the user manual for information on how to achieve this.

4 The listing Package [4]

5 The listings Package [5]

6 The listingsutf8 Package [6]

7 The mlppretty Package [7]

8 The minted Package [8]

9 The verbmments Package [9]

10 References

- [1] Rainer Schöpf and The L^AT_EX Team. “*verbatim* – Reimplementation of and extensions to L^AT_EX *verbatim*”, Version 1.5q, 2001-MAR-12. [Online], <http://ctan.org/pkg/verbatim>, [Accessed: 2017-MAR-22].
- [2] Steven B. Segletes. “*verbatimbox* – Deposit verbatim text in a box”, Version 3.13, 2014-MAR-12. [Online], <http://ctan.org/pkg/verbatimbox>, [Accessed: 2017-MAR-22].
- [3] Timothy Van Zandt and Herbert Voß and Denis Girou. “*fancyvrb* – Sophisticated verbatim text”, Version 2.8, 2010-MAY-15. [Online], <http://ctan.org/pkg/fancyvrb>, [Accessed: 2017-MAR-22].
- [4] Volker Kuhlmann and Matthew Hebley. “*listing* – Produce formatted program listings”, Version 1.2, 1999-MAY-25. [Online], <http://ctan.org/pkg/listing>, [Accessed: 2017-MAR-22].
- [5] Brooks Moses and Carsten Heinz and Jobst Hoffman. “*listings* – Typeset source code listings using L^AT_EX”, Version 1.6, 2015-JUN-04. [Online], <http://ctan.org/pkg/listings>, [Accessed: 2017-MAR-22].
- [6] Heiko Oberdiek. “*listingsutf8* – Allow UTF-8 in listings input”, Version 1.3, 2016-MAY-16. [Online], <http://ctan.org/pkg/listingsutf8>, [Accessed: 2017-MAR-22].
- [7] Julien Cretel. “*matlab-prettifier* – Pretty-print Matlab source code”, Version 0.3, 2014-JUN-19. [Online], <http://ctan.org/pkg/matlab-prettifier>, [Accessed: 2017-MAR-22].
- [8] Konrad Rudolph and Geoffrey Poore. “*minted* – Highlighted source code for L^AT_EX”, Version 2.4.1, 2016-OCT-31. [Online], <http://ctan.org/pkg/minted>, [Accessed: 2017-MAR-22].
- [9] Dejan Živkovic. “*verbments* – Syntax highlighting of source code in L^AT_EX documents”, Version 1.2, 2011-AUG-20. [Online], <http://ctan.org/pkg/verbments>, [Accessed: 2017-MAR-22].
- [10] Comprehensive T_EX Archive Network. “*Topic listing*”. [Online], <http://ctan.org/topic/listing>, [Accessed: 2017-MAR-22].

- [11] Comprehensive T_EX Archive Network. “*Topic verbatim*”. [Online], <http://ctan.org/topic/verbatim>, [Accessed: 2017-MAR-22].