The 'fancyvrb' package Fancy Verbatims in LaTeX

Timothy Van Zandt Princeton University Princeton - USA

Packaging and documentation by

Denis Girou (CNRS/IDRIS - France), Sebastian Rahtz† (Elsevier - GB) and Herbert Voß (FU Berlin - DE)

January 20, 2024

Abstract

This package provides very sophisticated facilities for reading and writing verbatim $T_{\!\!\!\!\!E}\!X$ code. Users can perform common tasks like changing font family and size, numbering lines, framing code examples, colouring text and conditionally processing text.

Report bugs to hvoss@tug.org

Thanks to Claudio Beccari, Karl Berry, Jean-François Burnol, Philippe Esperet, Michael Friendly, Mario Hassler, Ohad Kammar, Mikhail Kolodin, Andreas Matthias, Frank Mittelbach, Rolf Niepraschk, Will Robertson, Ulrich M. Schwarz, Thomas Siegel, Clemens Steinke, Vladimir Volovich, and Yukai Chou

CONTENTS

Contents

I	Package fancyvrb			4	
1	Introduction			4	
2	Verbatim material in footnotes			4	
3	Improved verbatim commands			5	
4	Ver	Verbatim environments			
	4.1	Custo	mization of verbatim environments	5	
		4.1.1	Comments	5	
		4.1.2	Initial characters to suppress	6	
		4.1.3	Customization of formatting	6	
		4.1.4	Changing individual line formatting	7	
		4.1.5	Fonts	7	
		4.1.6	Types and characteristics of frames	8	
		4.1.7	Label for the environment	10	
		4.1.8	Line numbering	11	
		4.1.9	Selection of lines to print	13	
		4.1.10	Spaces and tab characters	14	
		4.1.11	Space between lines	15	
		4.1.12	Vertical space before and after	15	
		4.1.13	Escape characters for inserting commands	16	
		4.1.14	Margins	17	
			Overfull box messages	18	
		4.1.16	Page breaks	18	
		4.1.17	Catcode characters	18	
		4.1.18	Active characters	18	
			Reference label	19	
	4.2 Different kinds of verbatim environments		19		
		4.2.1	Verbatim environment	19	
		4.2.2	BVerbatim environment	19	
		4.2.3	LVerbatim environment	20	
			Personalized environments	20	
5	Sav	Saving and restoring verbatim text and environments 21			

	CONTENTS
6 Writing and reading verbatim files	23
7 Automatic pretty printing	24
8 Known problems	24
9 Conclusion	24
II Package fancyvrb-ex	25
10 Example environment	25
11 CenterExample environment	26
12 SideBySideExample environment	26

Part I

Package fancyvrb

1 Introduction

'fancyvrb' is the development of the *verbatim* macros of the 'fancybox' package, Section 11 of [1]. It offers six kinds of extended functionality, compared to the standard LTFX verbatim environment:

- 1. verbatim commands can be used in footnotes,
- 2. several verbatim commands are enhanced,
- 3. a variety of verbatim environments are provided, with many parameters to change the way the contents are printed; it is also possible to define new customized verbatim environments.
- 4. a way is provided to save and restore verbatim text and environments,
- 5. there are macros to write and read files in verbatim mode, with the usual versatility,
- 6. you can build *example* environments (showing both result and verbatim text), with the same versatility as normal verbatim.

The package works by scanning a line at a time from an environment or a file. This allows it to pre-process each line, rejecting it, removing spaces, numbering it, etc, before going on to execute the body of the line with the appropriate catcodes set.

2 Verbatim material in footnotes

After a \VerbatimFootnotes macro declaration (to use after the preamble), it is possible to put verbatim commands and environments (the Lagent of the lagent

\text{\text{VerbatimFootnotes}}
\text{We can put verbatim\footnote{\verb+_Yes!_+} text in footnotes.}

We can put verbatim¹ text in footnotes.

¹_Yes!_

3 Improved verbatim commands

The \DefineShortVerb macro allows us to define a special character as an abbreviation to enclose verbatim text and the \UndefineShortVerb macro suppresses the special meaning of the specified character (the same functionalities are provided in the \text{ETr}X 'shortvrb' package):

To make matters more versatile, we can nominate *escape* characters in verbatim text (using the \Verb macro or with a 'shortverb' character defined), to perform formatting or similar tasks, using the commandchars parameter as shown for environments in paragraph 4.1.13.

4 Verbatim environments

Several verbatim environments are available, each with a lot of parameters to customize them. In the following examples we use the Verbatim environment, which is the equivalent of the standard verbatim. The parameters can be set globally using the \fvset macro or in an optional argument after the start of the environment^{2,3}.

```
\frac{1}{2} \quad \text{begin{Verbatim}}{ \text{First verbatim line.} \\ 2 \quad \text{Second verbatim line.} \\ \text{Second verbatim} \\ \text{end{Verbatim}} \\ \text{verbatim} \\ \text{end{Verbatim}} \\ \text{verbatim} \\ \text{verbatim}
```

4.1 Customization of verbatim environments

The appearance of verbatim environments can be changed in many and varied ways; here we list the keys that can be set.

4.1.1 Comments

commentchar (character): character to define comments in the verbatim code, so that lines starting with this character will not be printed (*Default: empty*).

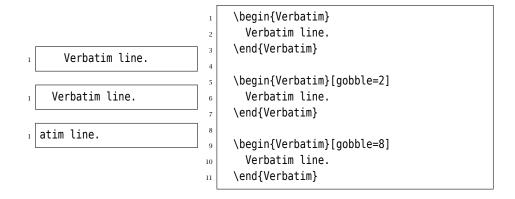
²For clarification in this paper, note that we generally indent each verbatim line with two spaces. ³This mechanism uses the '**keyval**' package from the standard Lagraphics distribution, written by David Carlisle.

```
\text{\lambda} \text{
```

Take care to a special effect if the comment character is not the first non blank one: it is because this character is in fact managed as the TEX comment one, that is to say that it gobble the newline character too. So, in this case, the current line will be joined with the next one and, more, the last one will be lost if it contain a comment, as 'fancyvrb' print a line only after finding it end character, which will never occured in this case...

4.1.2 Initial characters to suppress

gobble (integer): number of characters to suppress at the beginning of each line (up to a maximum of 9), mainly useful when environments are indented (*Default: empty*—no character suppressed).



4.1.3 Customization of formatting

formatcom (command): command to execute before printing verbatim text (*Default: empty*).

formatcom* (command): add definition to an existing one

```
| First verbatim line. | Second verbatim line. | Secon
```

4.1.4 Changing individual line formatting

The macro \FancyVerbFormatLine defines the way each line is formatted. Its default value is \def\FancyVerbFormatLine#1{#1}, but we can redefine it at our convenience (FancyVerbLine is the name of the line counter):

```
\renewcommand{\FancyVerbFormatLine}[1]{%
                                     \makebox[0cm][l]{$\Rightarrow$}#1}
⇒ First verbatim line.
                                   \begin{Verbatim}
                               3
                                     First verbatim line.
  Second verbatim line.
                               4
   Third verbatim line.
                                     Second verbatim line.
                               5
                                     Third verbatim line.
                               6
                                   \end{Verbatim}
                                   \renewcommand{\FancyVerbFormatLine}[1]{%
                                     \ifodd\value{FancyVerbLine}%
                               2
                                       \MakeUppercase{#1}\else#1\fi}
   FIRST VERBATIM LINE.
                                   \begin{Verbatim}
   Second verbatim line.
                                     First verbatim line.
                               5
   THIRD VERBATIM LINE.
                                     Second verbatim line.
                               6
                                     Third verbatim line.
                                   \end{Verbatim}
```

4.1.5 Fonts

fontfamily (family name): font family to use. tt, courier and helvetica are pre-defined (*Default: tt*).

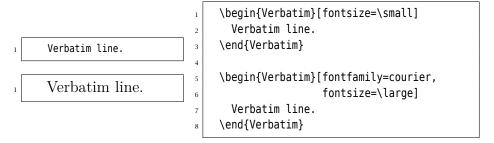
We can guess that PostScript fonts are available

```
\text{Verbatim line.} \text{\login{Verbatim} {fontfamily=helvetica}} \text{\verbatim line.} \text{\verbatim} \text{\verbatim} \\ \text{\verbatim} \\ \text{\verbatim} \\ \text{\verbatim} \\ \\ \text{\verbatim} \\ \text{\verbati
```

fontsize (font size) : size of the font to use (*Default: auto* — the same as the current font). If you use the 'relsize' package too, you can require a

change of the size proportional to the current one (for instance: fontsize=\relsize{-2}).

We can guess that PostScript fonts are available



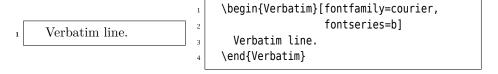
fontshape (font shape): font shape to use (*Default: auto*—the same as the current font).

We can guess that PostScript fonts are available

```
\text{Verbatim line.} \text{\left} \text{\le
```

fontseries (series name): MEX font 'series' to use (*Default: auto*—the same as the current font).

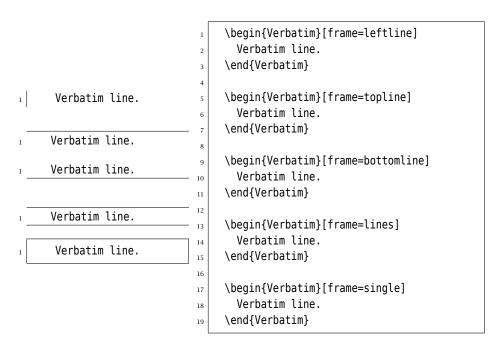
We can guess that PostScript fonts are available



4.1.6 Types and characteristics of frames

frame (none|leftline|topline|bottomline|lines|single): type of frame around the verbatim environment (*Default: none*—no frame). With leftline and single modes, a space of a length given by the Lagrange macro is added between the left vertical line and the text.

Problem at the top of a page...



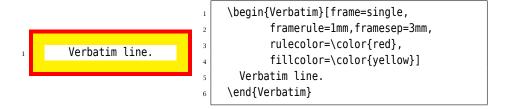
framerule (dimension): width of the rule of the frame (*Default: 0.4pt if framing specified*).

framesep (dimension): width of the gap between the frame and the text (*Default:* \ fboxsep).

```
\text{Verbatim line.} \text{\left} \text{\le
```

rule color (color command) : color of the frame rule, expressed in the standard $\mbox{\sc MT}_{E\!X}$ way ($Default:\ black$).

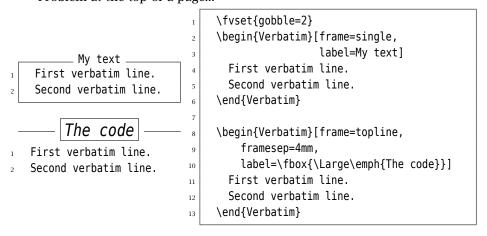
fillcolor (color command): color used to fill the space between the frame and the text (its thickness is given by framesep) (*Default: none*—no color).



4.1.7 Label for the environment

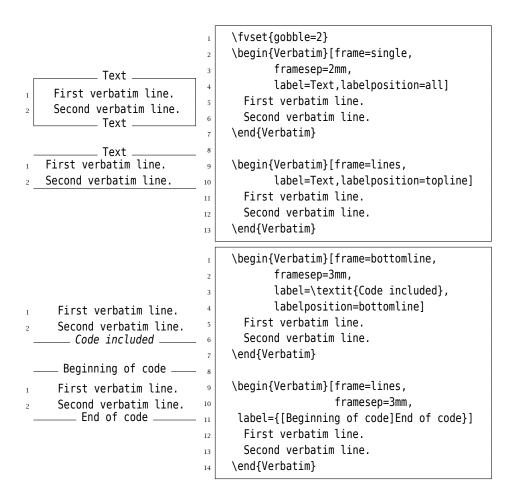
label ({[string]string}) : label(s) to print on top, bottom or both frame lines of the environment to describe it content (*Default: empty* — no label). If the label(s) contains special characters, as a comma or an equal sign, it must be put inside a group. If only one string is given, it will be used for both top and bottom lines (if the two are printed), but if an optional first label is given too, this one will be used for the top line and the second one for the bottom line. Note also that, if another value than topline, bottomline, lines or single is used for the frame parameter, the label(s) will not be printed.

Problem at the top of a page...



labelposition (none|topline|bottomline|all): position where to print the label if one is defined, which must be coherent with the kind of frame chosen (*Default: none if the label is empty, topline if one label is defined and all if two are defined*). Of course, some incompatible options (like frame=topline,labelposition=bottomline) will not print the label.

Problem at the top of a page...



4.1.8 Line numbering

numbers (none|left|right): numbering of the verbatim lines (*Default: none*—no numbering). If requested, this numbering is done *outside* the verbatim environment.

```
\begin{Verbatim}[gobble=2,numbers=left]
                                 First verbatim line.
                          2
                                 Second verbatim line.
                          3
First verbatim line.
                               \end{Verbatim}
                          4
Second verbatim line.
                               \begin{Verbatim}[gobble=2,
                          6
First verbatim line.
                                      numbers=right,numbersep=0pt]
                          7
Second verbatim line.
                                 First verbatim line.
                          8
                                 Second verbatim line.
                               \end{Verbatim}
```

numbersep (dimension): gap between numbers and verbatim lines (*Default: 12pt*).

firstnumber (auto|last|integer): number of the first line (*Default: auto* — numbering starts from 1). last means that the numbering is continued from the previous verbatim environment. If an integer is given, its value will be used to start the numbering.

```
\fvset{gobble=2,
                                      numbers=left,numbersep=3pt}
                           2
                               \begin{Verbatim}
                           3
                                 Verbatim line.
Verbatim line.
                               \end{Verbatim}
                           5
Verbatim line.
                               \begin{Verbatim}[firstnumber=last]
                                 Verbatim line.
                               \end{Verbatim}
Verbatim line.
                          10
                               \begin{Verbatim}[firstnumber=100]
                          11
                                 Verbatim line.
                          12
                               \end{Verbatim}
```

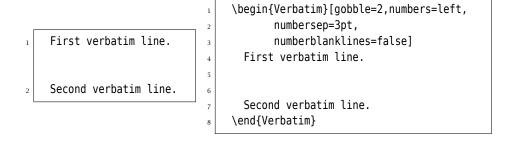
stepnumber (integer): interval at which line numbers are printed (*Default: 1*—all lines are numbered).

```
| Second verbatim line. | Third verbatim line. | Calculate the condition of the con
```

The macro \theFancyVerbLine defines the typesetting style of the numbering, and the counter used is FancyVerbLine:

```
\renewcommand{\theFancyVerbLine}{%
                                      \textcolor{red}{\small
                                        8.\alph{FancyVerbLine}}}
      First verbatim line.
                                    \begin{Verbatim}[gobble=2,
      Second verbatim line.
                                           numbers=left,numbersep=2pt]
8.b
                                5
      Third verbatim line.
                                      First verbatim line.
8.c
                                6
                                      Second verbatim line.
                                      Third verbatim line.
                                    \end{Verbatim}
```

numberblanklines (boolean): to number or not the empty lines (really empty or containing blank characters only) (*Default: true*—all lines are numbered).



4.1.9 Selection of lines to print

firstline (integer): first line to print (*Default: empty* — all lines from the first are printed).

lastline (integer): last line to print (*Default: empty*—all lines until the last one are printed).

```
\begin{Verbatim}[gobble=2,firstline,lastline,
                                      numbers=left,numbersep=2pt]
First verbatim line.
                                First verbatim line.
                          3
Second verbatim line.
                                Second verbatim line.
Third verbatim line.
                          5
                                Third verbatim line.
                              \end{Verbatim}
                              \begin{Verbatim}[gobble=2,firstline=2,
                                      numbers=left,numbersep=2pt]
                          2
Second verbatim line.
                                First verbatim line.
                          3
Third verbatim line.
                                Second verbatim line.
                          4
                                Third verbatim line.
                              \end{Verbatim}
```

```
\text{begin{Verbatim}[gobble=2,lastline=1, numbers=left,numbersep=2pt]}

First verbatim line.

First verbatim line.

Second verbatim line.

Third verbatim line.

\text{cend{Verbatim}}
```

Instead of specifying a firstline at which to start printing a range of lines, you can define a start string; the start of the range is the first line that exactly equals the string. (The comparison is made before any characters are gobbled off the front of the line.) Similarly for a stop string. You can mix line-numbers and strings, e.g. start at firstline, and end at a stop string. Specifying the strings is a bit klunky. Initially you must define the strings with \newcommand* as in:

```
\newcommand*\FancyVerbStartString{FROM}
\newcommand*\FancyVerbStopString{TO}
\hegin{Verbatim}
First verbatim line.

FROM
Second verbatim line.

7 TO
Third verbatim line.

9 \end{Verbatim}
```

To redefine the strings, you must use \renewcommand*.

With the setting lastline= or lastline=0 nothing will be printed.

```
foo | begin{Verbatim}[frame=none,lastline=] | First verbatim line. | Second verbatim line. | Third verbatim line. | bar | 6 | bend{Verbatim} | bar | b
```

4.1.10 Spaces and tab characters

showspaces (boolean): print a special character representing each space (*Default: false*—spaces not shown).

In practice, all verbatim environments, $\ensuremath{\text{VerbatimInput}}$, and $\ensuremath{\text{Verb have a *}}$ variant, which sets showspaces=true:

```
\begin{Verbatim*}
Verbatimuline
                              Verbatim line
                               \end{Verbatim*}
                              \begin{filecontents}
                                [noheader,force]{foo.txt}
Verbatim with tab
                                          bar baz
                              foo
Verbatim \_with \_tab \_\_\_\_\_\_\_.
                              \end{filecontents}
                              \Verb|Verbatim with tab
foouuuuuubaruubaz
                              \Verb*|Verbatim with tab
                                                               . [
                              \VerbatimInput*{foo.txt}
```

There are also some parameters to determine the way tab characters are interpreted (using tabs is in fact a rather old-fashioned style of coding):

showtabs (boolean): explicitly show tab characters (*Default: false*—tab characters not shown).

obeytabs (boolean): position characters according to the tabs (*Default: false*—tab characters are added to the current position).

tabsize (integer): number of spaces given by a tab character (*Default: 8*).

Pay attention that the behaviour of the environment/command from the LATEX kernel is different. In LATEX Tabs are printed as visible space, but with fancyvrb as spaces or with showtabs as visible Tab.

4.1.11 Space between lines

baselinestretch (auto|dimension): value to give to the usual 'baselinestretch' LETEX parameter (*Default: auto*—its current value just before the verbatim command).

```
First verbatim line.
Second verbatim line.
Second verbatim line.

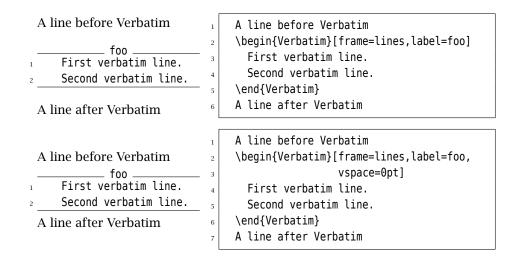
Second verbatim line.

Second verbatim line.

Vend{Verbatim}
```

4.1.12 Vertical space before and after

vspace (dimension): value to give to the usual vertical list space. (*Default:* \topsep—its current value just before the verbatim command).



4.1.13 Escape characters for inserting commands

commandchars (three characters): characters which define the character which starts a macro and marks the beginning and end of a group; thus lets us introduce *escape* sequences in verbatim code. Of course, it is better to choose special characters which are not used in the verbatim text! (*Default: empty*).

```
begin{Verbatim}[commandchars=\\\{\}]

textit{This is a comment}

First verbatim line.

Second verbatim line.

Third verbatim line.

Third verbatim line.

textcolor{red}{Third} verbatim line.

textcolor{red}{Third} verbatim line.

textcolor{red}{Verbatim}

begin{Verbatim}[commandchars=+\[\]]

textit[\textbf{Verbatim}] line].

textit[\textbf{Verbatim}] line].

textit[\textbf{Verbatim}] line].

textit[\textbf{Verbatim}] line].
```

Escaping of] is needed to prevent]] at the end which is not allowed. ALternatively you can write commandchars={+[]} or commandchars=+{[]}. Escaping is needed for all active characters. Pay attention that you do not use characters which are used in the code of fancyvrb itself, e.g. <>

Using this way, it is also possible to put labels to be able, later, to make reference to some lines of the verbatim environments:

4.1.14 Margins

xleftmargin (dimension): indentation to add at the start of each line (*Default: Opt* — no left margin).

xrightmargin (dimension): right margin to add after each line (*Default: 0pt* — no right margin).

```
\text{\lambda} \text{\left\begin{Verbatim} [frame=single, \text{\chiral} xrightmargin=1cm] \text{\chiral} Verbatim line. \text{\left\begin{Verbatim} \text{\chiral} \text{\
```

resetmargins (boolean): reset the left margin, which is useful if we are inside other indented environments (*Default: false*—no reset of the margin).

```
\begin{itemize}
                                 \item First item
                          2
• First item
                                 \begin{Verbatim}[frame=single]
                          3
                               Verbatim line.
                          4
     Verbatim line.
                                 \end{Verbatim}
                          5
                                 \item Second item
                          6
• Second item
                          7
                                 \begin{Verbatim}[frame=single,
                                                  resetmargins=true]
Verbatim line.
                               Verbatim line.
                          9
                                 \end{Verbatim}
                               \end{itemize}
```

4.1.15 Overfull box messages

hfuzz (dimension): value to give to the TEX \hfuzz dimension for text to format. This can be used to avoid seeing some unimportant *Overfull box* messages (*Default: 2pt*).

4.1.16 Page breaks

samepage (boolean): in very special circumstances, we may want to make sure that a verbatim environment is not broken, even if it does not fit on the current page. To avoid a page break, we can set the samepage parameter to true (Default: false).

4.1.17 Catcode characters

```
codes (macro): to specify catcode changes (Default: empty). codes* (macro): add to an existing definition.
```

For instance, this allows us to include formatted mathematics in verbatim text:

With codes* we can add code to an already existing definition of codes:

4.1.18 Active characters

defineactive (macro): to define the effect of *active* characters (*Default: empty*). defineactive* (macro): add the definition to an existing one.

This allows us to do some devious tricks: see the example in Section 6 on page 23.

4.1.19 Reference label

reflabel (<label>) : A label for use of \pageref.

```
First verbatim line.
Second verbatim line.

See the verbatim on page 19.

See The verbatim on page 19.

See See the verbatim on page 19.
```

4.2 Different kinds of verbatim environments

4.2.1 Verbatim environment

This is the 'normal' verbatim environment which we have been using up to now.

4.2.2 BVerbatim environment

This environment puts the verbatim material in a T_EX box. Some parameters do not work inside this environment (notably the framing ones), but two new ones are available:

boxwidth (auto|dimension): size of the box used (*Default: auto*—the width of the longest line is used).

baseline (b|c|t): position of the baseline (on the baseline, the center or the top of the box) (*Default: b*).

```
| begin{BVerbatim} [boxwidth=2cm] | First | Second | lend{BVerbatim} | Second | Second | Second | BVerbatim} | Second |
```

4.2.3 LVerbatim environment

This environment puts verbatim material into Large 'LR' mode (the so-called *left-to-right* mode, which in fact is the same thing that TeX itself calls *restricted horizontal mode*).

4.2.4 Personalized environments

It is easy to define personal customized environments. You can redefine the existing ones using the \RecustomVerbatimEnvironment macro or create your own ones, using the \DefineVerbatimEnvironment macro⁴. In each case, you specify the name of the new environment, the type of environment on which it is based, and a set of initial option values. The options can be overridden with an optional argument in the normal way:

⁴For verbatim commands, the \CustomVerbatimCommand and \RecustomVerbatimCommand macros also exist; for instance:

```
\DefineVerbatimEnvironment%
                                  {MyVerbatim}{Verbatim}
                                  {gobble=2,numbers=left,numbersep=2mm,
                                   frame=lines,framerule=0.8mm}
                                \begin{MyVerbatim}
First verbatim line.
                                 First verbatim line.
Second verbatim line.
                                  Second verbatim line.
                                \end{MyVerbatim}
First verbatim line.
                               \begin{MyVerbatim}[numbers=none,
Second verbatim line.
                           10
                                                   framerule=1pt]
                           11
                                 First verbatim line.
                           12
                                  Second verbatim line.
                           13
                                \end{MyVerbatim}
```

5 Saving and restoring verbatim text and environments

The \SaveVerb and \UseVerb macros allow us to save and restore verbatim material. \UseVerb itself is robust:

```
I have saved _verbatim_ and reuse it later as many times as I want

Using _verbatim_ 
verbatim . 

\[ \DefineShortVerb\{\|\} \ SaveVerb\{Verb\} |_verbatim_| \ I have saved \UseVerb\{Verb\} and reuse it later as many times as I want \ \subsection*\{Using \UseVerb\{Verb\}\} \\ \UseVerb\{Verb\}. \]
```

This also provides a solution to putting verbatim text inside \LaTeX commands which do not normally permit it:

```
\DefineShortVerb{\|}\SaveVerb{Verb}|_OK^| \marginpar{\UseVerb{Verb}}
```

_0K^

There is a useful ability to use verbatim text as the item text in a description list (something not normally permitted in LaTeX), using the aftersave parameter:

aftersave (macro): macro to dynamically save some verbatim material (*Default: empty*).

In the same way, we can use and restore (in normal, boxed and LR mode, using \UseVerbatim , \BUseVerbatim and \LUseVerbatim respectively) entire verbatim environments:

```
Verbatim line.
                                    \begin{SaveVerbatim}{VerbEnv}
                                    Verbatim line.
                                  \end{SaveVerbatim}
                              3
 and
                                  \UseVerbatim{VerbEnv}
     Verbatim line.
                                  and \UseVerbatim{VerbEnv}
 irst
            irst
                                    \begin{SaveVerbatim}[gobble=5]{VerbEnv}
 econd and econd.
                                    First
                                    Second
       irst
                                  \end{SaveVerbatim}
       econd
                                  \fbox{\BUseVerbatim{VerbEnv}}
                                  and \BUseVerbatim{VerbEnv}.
 and
                                  \LUseVerbatim{VerbEnv} and
       irst
                                  \LUseVerbatim{VerbEnv}
       econd
                                \begin{SaveVerbatim}{F00}
                                However, if I define a
                                reusable Verbatim by
 This works.
                                SaveVerbatim,
 I use Verbatim directly.
                                linenumbers works.
                                \end{SaveVerbatim}
However, if I define a
                                \begin{Verbatim}[numbers=left]
 reusable Verbatim by
                               This works.
 SaveVerbatim,
                             10 I use Verbatim directly.
 linenumbers works.
                               \end{Verbatim}
                             12
                             13 \UseVerbatim[numbers=left]{F00}
```

6 Writing and reading verbatim files

The command \VerbatimInput (the variants \BVerbatimInput and \LVerbatimInput also exist) allows inclusion of the contents of a file with verbatim formatting. Of course, the various parameters which we have described for customizing can still be used:

```
! A "hello" program
 program hello
   print *,"Hello world"
                                     \fvset{fontsize=\small}
 end program hello
                                   \VerbatimInput{hello.f90}
                              2
                              3
 ! A "hello" program
                                   \fvset{frame=single,numbers=left,
                                           numbersep=3pt}
                              5
 program hello
                                   \VerbatimInput{hello.f90}
                              6
   print *,"Hello world"
                              7
 end program hello
                                   \VerbatimInput[firstline=3,
                              8
                              9
                                       rulecolor=\color{green}]
 program hello
                                     {hello.f90}
                              10
   print *,"Hello world"
                              11
 end program hello
                                   \VerbatimInput[frame=lines,
                              12
                                       fontshape=sl,fontsize=\footnotesize]
                              13
                                     {hello.f90}
                              14
! A "hello" program
program hello
  print *,"Hello world"
end program hello
```

We can make use of the 'defineactive' parameter to set the comment lines in the program text in a different style:

```
\def\ExclamationPoint{\char33}
    ! A "hello" program
                                   \catcode'!=\active
                               2
                                   \VerbatimInput%
                               3
    program hello
                                     [defineactive=%
                               4
3
      print *,"Hello world"
                                       \def!{\color{cyan}\itshape
                               5
    end program hello
                                         \ExclamationPoint}]
                               6
                                     {hello.f90}
```

It is important to note that if the contents of the file does not fit on the page, it will be automatically broken across pages as needed (unless the samepage parameter has been set to true).

There is also a VerbatimOut environment to write verbatim text to an output file, in the same way:

7 Automatic pretty printing

Obviously, automatic *pretty printing* is outside the scope of this package. Nevertheless, this is specially interesting for verbatim inclusion of programming code files or fragments. In the FTEX world (not speaking of the *literate programming* way), there are software for some special languages, as the 'C++2LaTeX' package from Norbert KIESEL, but mainly two generic ones, which use completely different modes (an external preprocessor written in C and a TEX based solution): the 'LGrind' [3] system, currently maintened by Michael PIEFEL, and the 'listings' [4] package from Carsten Heinz.

Future versions of 'fancyvrb' and 'listings' packages are planned to cooperate, which will offer great advantages to both users of the two actual packages, and will allow 'fancyvrb' users to have automatic pretty printing of programming codes.

8 Known problems

• Vladimir VOLOVICH <vvv@vvv.vsu.ru> reported that the special character \th, available with T1 encoding, can't be included as verbatim with 'fancyvrb'. It can be true for other special characters too.

9 Conclusion

There are a few other possibilities that we have not described here. Note specially that it is possible to define a customization file (fancyvrb.cfg) loaded at the end of the package, to store definitions of your customized commands and environments and to redefine the attributes of existing ones.

Part II

Package fancyvrb-ex

This package defines some example environments which can write input code and output side by side or on top of each other. They are all used for this documentation of fancyvrb.

10 Example environment

```
begin{Example}
First verbatim line.
Second verbatim line.
Third verbatim line.

end{Example}

First verbatim line.
Second verbatim line.
Third verbatim line.
Third verbatim line.
```

First verbatim line. Second verbatim line. Third verbatim line.

- First verbatim line.
- Second verbatim line.
- 3 Third verbatim line.

First verbatim line. Second verbatim line. Third verbatim line.

11 CenterExample environment

```
First verbatim line. 1
Second verbatim line. 2
Third verbatim line. 3
```

First verbatim line. Second verbatim line. Third verbatim line.

12 SideBySideExample environment

```
begin{SideBySideExample}[xrightmargin=5cm,
frame=lines, numbers=left]
First verbatim line.
Second verbatim line.
Third verbatim line.
end{SideBySideExample}
```

```
First verbatim line. Second 1 verbatim line. Third verba- 2 Second verbatim line.

Third verbatim line. Third verbatim line.
```

References

- [1] Timothy VAN ZANDT, Documentation for 'fancybox': Box tips and tricks for ETEX. Available from CTAN:macros/latex/contrib/supported/fancybox, 1993.
- [2] Timothy VAN ZANDT, 'fancyvrb': Fancy Verbatims in LaTeX. Available from CTAN:macros/latex/contrib/supported/fancyvrb, 1998.
- [3] Various authors (current maintainer: Michael PIEFEL), *The 'LGrind' package*. Available from CTAN:support/lgrind, 1998.
- [4] Carsten Heinz, *The 'Listings' package*. Available from CTAN:macros/latex/c ontrib/supported/listings, 1996-1997.