The ‘fancyvrb’ package
Fancy Verbatims in LaTeX

Timothy Van Zandt
Princeton University
Princeton – USA
email: tvz@Princeton.EDU

Packaging, documentation and support
Denis Girou (CNRS/IDRIS – France) – Denis.Girou@idris.fr
and
Sebastian Rahtz (Elsevier – GB) – s.rahtz@elsevier.co.uk

Version 2.6
July 17, 1998
Documentation revised July 17, 1998

Abstract

This package provides very sophisticated facilities for reading and writing verbatim TeX code. Users can perform common tasks like changing font family and size, numbering lines, framing code examples, colouring text and conditionally processing text.

Contents

1 Introduction **2**

2 Verbatim material in footnotes **3**

3 Improved verbatim commands **3**

4 Verbatim environments **3**
- 4.1 Customization of verbatim environments 4
  - 4.1.1 Comments 4
  - 4.1.2 Initial characters to suppress 4
  - 4.1.3 Customization of formatting 5
  - 4.1.4 Changing individual line formatting 5
  - 4.1.5 Fonts 5
  - 4.1.6 Types and characteristics of frames 6
  - 4.1.7 Label for the environment 8
  - 4.1.8 Line numbering 9
  - 4.1.9 Selection of lines to print 11
  - 4.1.10 Spaces and tab characters 12

4.1.11 Space between lines 12
4.1.12 Escape characters for inserting commands 12
4.1.13 Margins 13
4.1.14 Overfull box messages 14
4.1.15 Page breaks 14
4.1.16 Catcode characters 14
4.1.17 Active characters 14
4.2 Different kinds of verbatim environments 14
4.2.1 Verbatim environment 14
4.2.2 BVerbatim environment 15
4.2.3 LVerbatim environment 15
4.2.4 Personalized environments 15

5 Saving and restoring verbatim text and environments **16**

6 Writing and reading verbatim files **17**

7 Automatic pretty printing **19**

8 Known problems **19**

9 Thanks **19**

10 Conclusion **19**

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 LaTeX 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 LaTeX or 'fancyvrb' ones) in footnotes, unlike in standard LaTeX:

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

We can put verbatim[1] text in footnotes.

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 LaTeX 'shortvrb' package):

We can simply write `_verbatim_` material using a single `_delimiter_` And we can `_change_` the character.

```
\DefineShortVerb{\|}
We can simply write \Verb+_verbatim_+
material using a single |_delimiter_|
\UndefineShortVerb{\|}
\DefineShortVerb{\+}
And we can +_change_+ the character.
```

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 4.1.12.

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].

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

```
\begin{Verbatim}
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

---

[1] `_Yes!_`

[2] For clarification in this paper, note that we generally indent each verbatim line with two spaces.

[3] This mechanism uses the '**keyval**' package from the standard LaTeX graphics distribution, written by David Carlisle.

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*).

```
% A comment
Verbatim line.
```

```
\begin{Verbatim}[commentchar=!]
  % A comment
  Verbatim line.
  ! A comment that you will not see
\end{Verbatim}
```

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...

```
First line.   Second.
```

```
\begin{Verbatim}[commentchar=\%]
  First line. % First line
  Second.
  Third line. % Third line lost...
\end{Verbatim}
```

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).

```
  Verbatim line.

Verbatim line.

im line.
```

```
\begin{Verbatim}
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[gobble=2]
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[gobble=8]
  Verbatim line.
\end{Verbatim}
```

4.1.3 Customization of formatting

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

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

```
\begin{Verbatim}[formatcom=\color{red}]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

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):

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

```
\renewcommand{\FancyVerbFormatLine}[1]{%
  \makebox[0cm][l]{$\Rightarrow$}#1}
\begin{Verbatim}
  First verbatim line.
  Second verbatim line.
  Third verbatim line.
\end{Verbatim}
```

```
  FIRST VERBATIM LINE.
Second verbatim line.
  THIRD VERBATIM LINE.
```

```
\renewcommand{\FancyVerbFormatLine}[1]{%
  \ifodd\value{FancyVerbLine}%
    \MakeUppercase{#1}\else#1\fi}
\begin{Verbatim}
  First verbatim line.
  Second verbatim line.
  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*).

Verbatim line.

```
\begin{Verbatim}[fontfamily=helvetica]
  Verbatim line.
\end{Verbatim}
```

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}`).

```
Verbatim line.

Verbatim line.
```

```
\begin{Verbatim}[fontsize=\small]
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[fontfamily=courier,
                 fontsize=\large]
  Verbatim line.
\end{Verbatim}
```

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

```
Verbatim line.
```

```
\begin{Verbatim}[fontfamily=courier,
                 fontshape=it]
  Verbatim line.
\end{Verbatim}
```

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

```
Verbatim line.
```

```
\begin{Verbatim}[fontfamily=courier,
                 fontseries=b]
  Verbatim line.
\end{Verbatim}
```

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 LaTeX `\fboxsep` macro is added between the left vertical line and the text.

```
 Verbatim line.
```

```
 Verbatim line.
```

```
 Verbatim line.
```

```
 Verbatim line.
```

```
 Verbatim line.
```

```
\begin{Verbatim}[frame=leftline]
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[frame=topline]
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[frame=bottomline]
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[frame=lines]
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[frame=single]
  Verbatim line.
\end{Verbatim}
```

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

```
 Verbatim line.
```

```
\begin{Verbatim}[frame=single,
                 framerule=1mm]
  Verbatim line.
\end{Verbatim}
```

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

```
 Verbatim line.
```

```
\begin{Verbatim}[frame=single,
                 framesep=5mm]
  Verbatim line.
\end{Verbatim}
```

rulecolor (color command) : color of the frame rule, expressed in the standard LaTeX way (*Default: black*).

```
 Verbatim line.
```

```
\begin{Verbatim}[frame=single,
                 rulecolor=\color{red}]
  Verbatim line.
\end{Verbatim}
```

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).

```
Verbatim line.
```

```
\begin{Verbatim}[frame=single,
      framerule=1mm,framesep=3mm,
      rulecolor=\color{red},
      fillcolor=\color{yellow}]
  Verbatim line.
\end{Verbatim}
```

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.

```
My text
First verbatim line.
Second verbatim line.

The code
First verbatim line.
Second verbatim line.
```

```
\fvset{gobble=2}
\begin{Verbatim}[frame=single,
                 label=My text]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}

\begin{Verbatim}[frame=topline,
    framesep=4mm,
    label=\fbox{\Large\emph{The code}}]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

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.

```
———— Text ————
First verbatim line.
Second verbatim line.
———— Text ————

———— Text ————
First verbatim line.
Second verbatim line.
————————————
```

```
\fvset{gobble=2}
\begin{Verbatim}[frame=single,
       framesep=2mm,
       label=Text,labelposition=all]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}

\begin{Verbatim}[frame=lines,
       label=Text,labelposition=topline]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

```
  First verbatim line.
  Second verbatim line.
——— Code included ———

—— Beginning of code ——
  First verbatim line.
  Second verbatim line.
———— End of code ————
```

```
\begin{Verbatim}[frame=bottomline,
       framesep=3mm,
       label=\textit{Code included},
       labelposition=bottomline]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}

\begin{Verbatim}[frame=lines,
                 framesep=3mm,
 label={[Beginning of code]End of code}]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

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.

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

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

```
\begin{Verbatim}[gobble=2,numbers=left]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}

\begin{Verbatim}[gobble=2,
       numbers=right,numbersep=0pt]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

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

12pt).

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

```
\begin{Verbatim}[gobble=2,
       numbers=left,numbersep=2pt]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

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.

```
  1 Verbatim line.

  2 Verbatim line.

100 Verbatim line.
```

```
\fvset{gobble=2,
       numbers=left,numbersep=3pt}
\begin{Verbatim}
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[firstnumber=last]
  Verbatim line.
\end{Verbatim}

\begin{Verbatim}[firstnumber=100]
  Verbatim line.
\end{Verbatim}
```

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

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

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

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

```
8.a First verbatim line.
8.b Second verbatim line.
8.c Third verbatim line.
```

```
\renewcommand{\theFancyVerbLine}{%
  \textcolor{red}{\small
    8.\alph{FancyVerbLine}}}
\begin{Verbatim}[gobble=2,
       numbers=left,numbersep=2pt]
  First verbatim line.
  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).

```
1 First verbatim line.


2 Second verbatim line.
```

```
\begin{Verbatim}[gobble=2,numbers=left,
       numbersep=3pt,
       numberblanklines=false]
  First verbatim line.


  Second verbatim line.
\end{Verbatim}
```

4.1.9 Selection of lines to print

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

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

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

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

```
1 First verbatim line.
```

```
\begin{Verbatim}[gobble=2,lastline=1,
       numbers=left,numbersep=2pt]
  First verbatim line.
  Second verbatim line.
  Third verbatim line.
\end{Verbatim}
```

4.1.10 Spaces and tab characters

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

```
␣␣Verbatim␣line.
```

```
\begin{Verbatim}[showspaces=true]
  Verbatim line.
\end{Verbatim}
```

In practice, all verbatim environments have a `*` variant, which sets `showspaces=true`:

```
␣␣Verbatim␣line.
```

```
\begin{Verbatim*}
  Verbatim line.
\end{Verbatim*}
```

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*).

4.1.11 Space between lines

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

```
  First verbatim line.

  Second verbatim line.
```

```
\begin{Verbatim}[baselinestretch=2]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

4.1.12 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*).

```
% This is a comment
First verbatim line.
Second verbatim line.
Third verbatim line.

\textbf{Verbatim} line.
```

```
\begin{Verbatim}[commandchars=\\\{\}]
  \textit{% This is a comment}
  First verbatim line.
  \fbox{Second} verbatim line.
  \textcolor{red}{Third} verbatim line.
\end{Verbatim}

\begin{Verbatim}[commandchars=+\[\]]
  +textit[\textbf{Verbatim} line].
\end{Verbatim}
```

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

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

As I previously shown line 2, it is...

```
\begin{Verbatim}[commandchars=\\\{\},
       numbers=left,numbersep=2pt]
  First verbatim line.
  Second line.\label{vrb:Important}
  Third verbatim line.
\end{Verbatim}

  As I previously shown
line~\ref{vrb:Important}, it is...
```

4.1.13 Margins

xleftmargin (dimension) : indentation to add at the start of each line (*Default: 0pt* — no left margin).

```
   Verbatim line.
```

```
\begin{Verbatim}[frame=single,
                 xleftmargin=5mm]
  Verbatim line.
\end{Verbatim}
```

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

```
  Verbatim line.
```

```
\begin{Verbatim}[frame=single,
                 xrightmargin=1cm]
  Verbatim line.
\end{Verbatim}
```

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

- First item

  ```
  Verbatim line.
  ```

- Second item

```
Verbatim line.
```

```
\begin{itemize}
  \item First item
  \begin{Verbatim}[frame=single]
Verbatim line.
  \end{Verbatim}
  \item Second item
  \begin{Verbatim}[frame=single,
                   resetmargins=true]
Verbatim line.
  \end{Verbatim}
\end{itemize}
```

4.1.14 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*).

4.1.15 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*).

4.1.16 Catcode characters

codes (macro) : to specify *catcode* changes (*Default: empty*).

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

`x=1/sqrt(z**2) !` $\frac{1}{\sqrt{z^2}}$

```
\begin{Verbatim}[commandchars=\\\{\},
      codes={\catcode`$=3\catcode`^=7}]
  x=1/sqrt(z**2) ! $\frac{1}{\sqrt{z^2}}$
\end{Verbatim}
```

4.1.17 Active characters

defineactive (macro) : to define the effect of *active* characters (*Default: empty*).

This allows us to do some devious tricks: see the example in Section 6 on page 17.

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 TeX 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*).

```
First
SecondFirst
      Second
```

```
\fvset{gobble=2}
\begin{BVerbatim}
  First
  Second
\end{BVerbatim}
\begin{BVerbatim}[baseline=c]
  First
  Second
\end{BVerbatim}
```

```
First
Second     First
           Second
```

```
\begin{BVerbatim}[boxwidth=2cm]
  First
  Second
\end{BVerbatim}
\begin{BVerbatim}[boxwidth=2cm,
                  baseline=t]
  First
  Second
\end{BVerbatim}
```

4.2.3 LVerbatim environment

This environment puts verbatim material into LaTeX '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` macro[4]. 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:

[4] For verbatim commands, the `\CustomVerbatimCommand` and `\RecustomVerbatimCommand` macros also exist; for instance:
`\RecustomVerbatimCommand{\VerbatimInput}{VerbatimInput}{frame=lines}`

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

```
\RecustomVerbatimEnvironment
  {Verbatim}{Verbatim}
  {gobble=2,frame=single}
\begin{Verbatim}
  First verbatim line.
  Second verbatim line.
\end{Verbatim}
```

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

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

```
\DefineVerbatimEnvironment%
  {MyVerbatim}{Verbatim}
  {gobble=2,numbers=left,numbersep=2mm,
   frame=lines,framerule=0.8mm}
\begin{MyVerbatim}
  First verbatim line.
  Second verbatim line.
\end{MyVerbatim}

\begin{MyVerbatim}[numbers=none,
                   framerule=1pt]
  First verbatim line.
  Second verbatim line.
\end{MyVerbatim}
```

5 Saving and restoring verbatim text and environments

The `\SaveVerb` and `\UseVerb` macros allow us to save and restore verbatim material.

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

```
\DefineShortVerb{\|}
\SaveVerb{Verb}|_verbatim_|
I have saved \UseVerb{Verb} and reuse
it later as many times as I want
\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}}
```

`_OK^`

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*).

`\MyCommand` : my command

```
\newcommand{\Vitem}{%
  \SaveVerb[aftersave={%
    \item[\UseVerb{Vitem}]}]{Vitem}}
\DefineShortVerb{\|}
\begin{description}
  \Vitem|\MyCommand|: my command
\end{description}
```

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.
```

and

```
  Verbatim line.
```

```
\begin{SaveVerbatim}{VerbEnv}
  Verbatim line.
\end{SaveVerbatim}
\UseVerbatim{VerbEnv}
and \UseVerbatim{VerbEnv}
```

$\boxed{\begin{array}{l}\texttt{st}\\ \texttt{ond}\end{array}}$ and `st` `ond`.

```
     st
     ond
```

and

```
     st
     ond
```

```
\begin{SaveVerbatim}[gobble=5]{VerbEnv}
  First
  Second
\end{SaveVerbatim}

\fbox{\BUseVerbatim{VerbEnv}}
and \BUseVerbatim{VerbEnv}.

\LUseVerbatim{VerbEnv} and
\LUseVerbatim{VerbEnv}
```

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"
end program hello
```

```
! A "hello" program

program hello
  print *,"Hello world"
end program hello
```

```
program hello
  print *,"Hello world"
end program hello
```

```
! A "hello" program

program hello
  print *,"Hello world"
end program hello
```

```
\fvset{fontsize=\small}
\VerbatimInput{hello.f90}

\fvset{frame=single,numbers=left,
       numbersep=3pt}
\VerbatimInput{hello.f90}

\VerbatimInput[firstline=3,
    rulecolor=\color{green}]
  {hello.f90}

\VerbatimInput[frame=lines,
    fontshape=sl,fontsize=\footnotesize]
  {hello.f90}
```

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

```
! A "hello" program

program hello
  print *,"Hello world"
end program hello
```

```
\def\ExclamationPoint{\char33}
\catcode`!=\active
\VerbatimInput%
  [defineactive=%
    \def!{\color{cyan}\itshape
      \ExclamationPoint}]
  {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:

```
  I write that.
  And that too.
```

```
\begin{VerbatimOut}{file.txt}
  I write that.
  And that too.
\end{VerbatimOut}

\VerbatimInput[frame=single,
  numbers=left,numbersep=6pt]{file.txt}
```

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 LaTeX 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 Thanks

For interesting comments and suggestions, we would like to thank specially (alphabetic order): Philippe ESPERET `<esperet@marie.polytechnique.fr>`, Michael FRIENDLY `<friendly@hotspur.psych.yorku.ca>`, Rolf NIEPRASCHK `<niepraschk@ptb.de>` and for bug reports Mario HASSLER `<HASSLER@ippnv2.ipp.kfa-juelich.de>`, Mikhail KOLODIN `<myke@morrigan.spb.su>` and Vladimir VOLOVICH `<vvv@vvv.vsu.ru>`.

10 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.

References

[1] Timothy VAN ZANDT, *Documentation for 'fancybox': Box tips and tricks for LaTeX*. 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/contrib/supported/listings`, 1996-1997.