Cahiers GUTenberg

VERBATIM REVISITED–THE FANCYVRB PACKAGE

Denis Girou, Sebastian Rahtz

Cahiers GUTenberg, nº 28-29 (1998), p. 158-179.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1998___28-29_158_0>

Verbatim Revisited – the ‘fancyvrb’ Package

Denis Girou[a] and Sebastian Rahtz[b]

[a]Cnrs/Idris
Bâtiment 506
B.P. 167
91403 Orsay Cedex
France
<Denis.Girou@idris.fr>

[b]Elsevier Science Ltd
The Boulevard
Langford Lane
Kidlington
Oxford
Grande-Bretagne
<s.rahtz@elsevier.co.uk>

Abstract. *This talk introduces Timothy* van Zandt *’s ‘fancyvrb’ [3] LaTeX package, which 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. The main part of this paper is a set of tutorial examples of how to create customized verbatim environments, and it concludes with a description of how ‘fancyvrb’ was used in the typesetting of the* LaTeX Graphics Companion.

Résumé. Nous présentons l’extension LaTeX ‘fancyvrb’ [3] de Timothy van Zandt, qui offre des possibilités très sophistiquées pour lire et écrire du code TeX en mode *verbatim*. On peut effectuer les opérations les plus courantes telles que changer de fonte et de corps, numéroter les lignes, entourer le texte d’un cadre, le colorier et le traiter conditionnellement. La plus grande partie de cet article est un ensemble d’exemples qui décrivent ces possibilités et comment créer des environnements personnalisés, et il se termine par la description de l’utilisation qui a été faite de ‘fancyvrb’ pour la composition du *LaTeX Graphics Companion*.

1. Introduction

‘fancyvrb’ is the development of the *verbatim* macros of the ‘fancybox’ package, Section 11 of [2]. 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.

Naturally, we have used 'fancyvrb' in preparing this article.

2. Verbatim material in footnotes

After a `\VerbatimFootnotes` macro declaration (to use after the preamble), it is possible to put verbatim commands and environments 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 *docstrip* package):

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

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

[1] `_Yes!_`

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 4.1.11.

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}
```

4.1. Customization of verbatim environments

The appearance of verbatim environments can be changed in many and varied ways; here we list some of 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}
```

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).

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.

```
  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{\FV@ObeyTabs{#1}}`, but we can redefine it at our convenience (`FancyVerbLine` is the name of the line counter):

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

  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.
\end{Verbatim}

\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).

```
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|topline|bottomline|lines|single) : type of frame around the verbatim environment (*Default: none* — no frame).

Verbatim line.

Verbatim line.

Verbatim line.

Verbatim line.

Verbatim line.

```
\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.

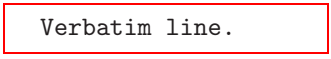
```
\begin{Verbatim}%
      [frame=single,framerule=1mm]
  Verbatim line.
\end{Verbatim}
```

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

Verbatim line.

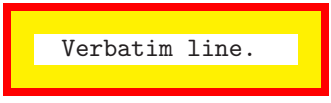
```
\begin{Verbatim}%
      [frame=single,framesep=5mm]
  Verbatim line.
\end{Verbatim}
```

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




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

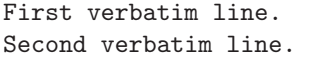
fillcolor (color command) : color used to fill the space between the frame and the text (its thickness is given by `framesep`) (*Default: none* — no color).

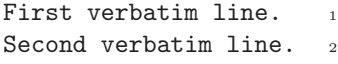



```
\begin{Verbatim}[frame=single,
       framerule=1mm,framesep=3mm,
       rulecolor=\color{red},
       fillcolor=\color{yellow}]
  Verbatim line.
\end{Verbatim}
```

4.1.7. 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.

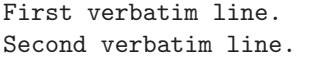





```
\begin{Verbatim}[gobble=2,
                 numbers=left]
  First verbatim line.
  Second verbatim line.
\end{Verbatim}

\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*).




```
\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.
```

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

4.1.8. 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.
\end{Verbatim}
```

4.1.9. 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*).

4.1.10. 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}
```

4.1.11. 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}
```

4.1.12. 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}
```

4.1.13. 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*).

4.1.14. 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*).

4.1.15. 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) ! 1/sqrt(z^2)
```

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

4.1.16. 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 172.

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
Second
```

```
First
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, using the `\DefineVerbatimEnvironment` macro; 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:

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

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

```
\DefineVerbatimEnvironment%
  {MyVerbatim}{Verbatim}
  {numbers=left,numbersep=5pt,
   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}}
\subsubsection*{It's \protect\UseVerb{Verb}}
```

_OK^

*It's _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}
```

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

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

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

st st
ond and ond.

st
ond

and

st
ond

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
```

```
\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,
    fontfamily=courier,fontshape=sl,
    fontsize=\footnotesize]
  {hello.f90}
```

```
1 ! A "hello" program
2
3 program hello
4   print *,"Hello world"
5 end program hello
```

```
3 program hello
4   print *,"Hello world"
5 end program hello
```

```
1 ! A "hello" program
2
3 program hello
4   print *,"Hello world"
5 end program hello
```

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:

```
1   I write that.
2   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. Example environments

A useful application of 'fancyvrb' is in building *example* environments (showing both (LA)TEX code and the result of it). The extra package `fvrb-ex.sty` includes several variations. These environments set the following characteristics:

`gobble=0,commentchar=£,commandchars=ğμű,numbersep=3pt`

so we must explicitly set `numbers=left` to have line numbering and `frame=...` to have framing.

A powerful and useful feature of these example environments is that they allow us to use *highlighting* attributes for the verbatim code. These can be de-activated if we want to run the code itself, and then generate, for example, either a coloured or a black and white version (if we use an indirect coding style) based on a simple switch at compile time.

The defined environments are:

Example : verbatim text with result below

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

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

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

CenterExample : verbatim text with *centered* result below

```
\fvset{frame=lines,framerule=1mm,numbers=left}
\begin{CenterExample}
  ğHLbμFirstű verbatim line.
  ğHLCBWzμSecondű verbatim line.
\end{CenterExample}
```

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

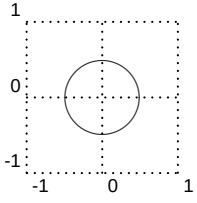
First verbatim line. Second verbatim line.

PCenterExample : verbatim text with *centered* result below, using a PSTricks `pspicture` environment (useful for graphic objects of TEX size zero, done using LATEX graphics macros or some from other graphic packages). It is also sometines useful to draw a grid underneath the graphic, which can easily be done using the `\showgrid` macro.

```
\begin{PCenterExample}(-0.5,-0.5)(0.5,0.5)
  \setlength{\unitlength}{1cm}
  \put(0,0){\circle{1}}
\end{PCenterExample}
\showgrid
\fvset{frame=lines,framerule=0.5mm}
\begin{PCenterExample}(-1,-1)(1,1)
  \setlength{\unitlength}{1cm}
  \put(0,0){\circle{1}}
\end{PCenterExample}
```

```
\setlength{\unitlength}{1cm}
\put(0,0){\circle{1}}
```

```
\setlength{\unitlength}{1cm}
\put(0,0){\circle{1}}
```



SideBySideExample : verbatim text on right side with result on left side

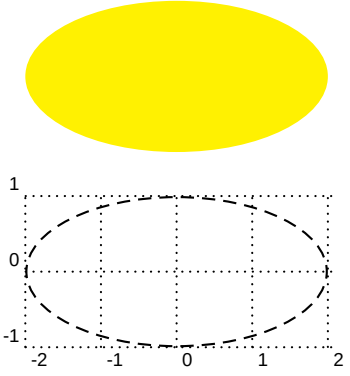
First Second

```
1 First
2 Second
```

```
\fvset{xrightmargin=3cm,numbers=left}
\begin{SideBySideExample}
  First
  Second
\end{SideBySideExample}
```

PSideBySideExample : verbatim text on right side and result on left side, using a PSTricks `pspicture` environment. In this environment too, we can draw a grid below the graphic.

```
\fvset{frame=single,xrightmargin=5cm}
\begin{PSideBySideExample}(-2,-1)(2,1)
  \psellipse*[linecolor=ğHLCBWzμyellowű](2,1)
\end{PSideBySideExample}
\showgrid
\begin{PSideBySideExample}(-2,-1)(2,1)
  \psellipse[linestyle=ğHLCBWzμdashedű](2,1)
\end{PSideBySideExample}
```




```
\psellipse*[linecolor=yellow](2,1)
```

```
\psellipse[linestyle=dashed](2,1)
```

8. ‘fancyvrb’ and *The LaTeX Graphics Companion*

The LaTeX Graphics Companion [1] contains some 400 examples of various packages, all of which were prepared using ‘fancyvrb’. Only 155 of them could be run directly, using the type of example environment described above; the rest wrote the code to an external file. These files were then run through LaTeX (possibly with a pre-processor, or other programs such as MetaPost), and the dvi file converted to Encapsulated PostScript which could be included on the next run of the book. To complicate matters, it was also necessary:

1. To assign a unique number to each example, and print it next to the output, so that the examples file could be usefully made available in TeX archives.
2. To format the code side by side with output depending on whether the page was odd or even; on even pages, it could extend into the left margin, on odd pages into the right.
3. To establish index terms for each example.
4. To allow for ‘continuation’ numbering, so that a numbered verbatim could be split up with comments.

Some examples were set side by side, others with output below the code; some PSTricks examples were set with a grid behind the picture. The checking of odd or even pages required generating a LaTeX `\label` for each example, and then checking the resulting recorded page number on the next run.

Perhaps the best demonstration of the abilities of 'fancyvrb' is the way the code necessary to make a complete LaTeX job could be generated, but not included on the printed pages. Thus to typeset example 8-1-4 (reproduced in Figure 1), the source file has the following code:

```
\def\PreambleCommands{\usepackage{chess}}
\begin{Example}{\xLcs{showboard}}
\usepackage {chess}
\font\Chess=xcheq at 18pt
\setlength{\fontdimen2\Chess}{0pt}
\board{B* * *KR}
    {* * * *r}
    { *R* * *}
    {* b p p }
    { *P*k*P*}
    {*p* P *p}
    { P *P* P}
    {* *N*N* }
\[ \showboard \]
\end{Example}
```

Note the definition of the `\usepackage` commands needed to run the example, and the definition of a term for the index (`\textpiece`). To run the code through LaTeX, the file written out must be prefixed by:

```
\documentclass{ppex}
\nonstopmode
\usepackage {cchess}
\pagestyle{empty}
\setlength\textwidth{159.0pt}
\begin{document}
\ResetPreambleCommands
\ReadyForTheFray
```

and followed by:

```
\usepackage {chess}
\font\Chess=xcheq at 18pt
\setlength{\fontdimen2\Chess}{0pt}
\board{B* * *KR}
      {* * * *r}
      { *R* * *}
      {* b p p }
      { *P*k*P*}
      {*p* P *p}
      { P *P* P}
      {* *N*N* }
\[ \showboard \]
```

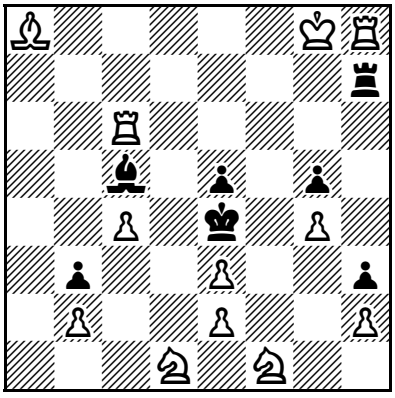


Figure 1 – Example 8-1-4 from the *LaTeX Graphics Companion* (reproduced with the permission of Addison Wesley)

```
\end{document}
```

The `\usepackage` command has to appear twice, once visible in the code, and once in the document preamble (since packages cannot be loaded after the start of the document environment). Writing out this prefix and suffix material is quite easy, using a customized `VerbatimOut` environment. When it is read back *in* for the printed code, 'fancyvrb' takes note of two macros, `\FancyVerbStartString` and `\FancyVerbStopString`. If these are defined, verbatim input ignores everything up to `\FancyVerbStartString`, and stops printing when it reaches `\FancyVerbStopString`:

```
\edef\FancyVerbStartString{\string\ReadyForTheFray}
\edef\FancyVerbStopString{\string\end{document}}
```

The strange commands `\ResetPreambleCommands` and `\ReadyForTheFray` are defined as follows, allowing for the `\usepackage` command to be printed, but have no effect:

```
\newcommand\gobblepreamble[2][]{}
\newcommand\ResetPreambleCommands{\let\usepackage\gobblepreamble}
\let\ReadyForTheFray\relax
```

While this is only a brief overview of some of the problems and solutions adopted for a complex typesetting job like *The LaTeX Graphics Companion*

(15 customized verbatim example environments were defined), the book demonstrates, we hope, the wide range of possibilities for 'fancyvrb'.

9. Conclusion

There are a few other possibilities that we have not described here. 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. This latter task can be performed using the `\RecustomVerbatimCommand` and `\RecustomVerbatimEnvironment` macros; for instance,

```
\RecustomVerbatimCommand{%
       \VerbatimInput}{VerbatimInput}{frame=single}.
```

The 'fancyvrb' package has an impressive array of functionality, combined with great ease of use and flexibility. We are convinced that it is a very good solution for all problems of verbatim inclusion under LaTeX.

References

[1] Michel Goossens, Sebastian Rahtz and Frank Mittelbach, *The LaTeX Graphics Companion*, Addison-Wesley, Reading, Massachusetts, 1997.

[2] Timothy van Zandt, *Documentation for fancybox.sty: Box tips and tricks for LaTeX*. Available from CTAN, `graphics/pstricks/latex`, 1993.

[3] Timothy van Zandt, *fancyvrb.sty: Fancy Verbatims in LaTeX*. Available from CTAN, `macros/latex/contrib/supported`, 1998.