The ‘fancyvrb’ package
Fancy Verbatims in LaTeX

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

Packaging and documentation by

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

Version 2.8
May 15, 2010
Documentation revised May 15, 2010

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

'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.

[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 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}
```

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*).

[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.

```
% 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*).

```
1First verbatim line.
2Second 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.aFirst verbatim line.
8.bSecond verbatim line.
8.cThird 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}
```

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: `Second verbatim line.`

```
\newcommand*\FancyVerbStartString{FROM}
\newcommand*\FancyVerbStopString{TO}
\begin{Verbatim}[gobble=2]
  First verbatim line.
FROM
  Second verbatim line.
TO
  Third verbatim line.
\end{Verbatim}
```

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

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 20.

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:

```
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.

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

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}
```

```
┌────┐    st
│st  │
│ond │ and 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
```

```
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
```

```
\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:

```
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 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@gmx.de` and for bug reports Mario HASSLER `HASSLER@ippnv2.ipp.kfa-juelich.de`, Mikhail KOLODIN `myke@morrigan.spb.su`, Andreas Matthias, Ulrich M. Schwarz, 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.

11 Driver file

The next bit of code contains the documentation driver file for TeX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the `docstrip` program.

```
⟨*driver⟩
\documentclass{ltxdoc}
\GetFileInfo{fancyvrb.dtx}
\usepackage{color}
\usepackage{fancyvrb,shortvrb}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage[charter]{mathdesign}
```

```
\usepackage{url}
\newif\ifChangeBar
\IfFileExists{changebar.sty}%
             {\ChangeBartrue\usepackage[dvips,rightbars]{changebar}}{}
\EnableCrossrefs
\CodelineIndex
\RecordChanges
\OnlyDescription                % Comment it for implementation details
%\Oldmakeindex                   % Uncomment if your MakeIndex is pre-0.9
\hbadness=7000                  % Over and under full box warnings
\hfuzz=3pt
\begin{document}
  \DocInput{fancyvrb.dtx}
\end{document}
⟨/driver⟩
```

12 ‘fancyvrb’ code

<*fancyvrb>

> ***Disclaimer (D.G./S.R.)****: This is the original comments of the code by Timothy* VAN ZANDT*. We have not change them.*

12.1 Preambule

What we need.

```
\NeedsTeXFormat{LaTeX2e}
```

Who we are.

```
\def\fileversion{2.7a, with DG/SPQR fixes, and firstline=lastline fix}
\def\filedate{2008/02/07}
\ProvidesPackage{fancyvrb}[\filedate]
\message{Style option: ‘fancyvrb’ v\fileversion \space  <\filedate> (tvz)}
\csname fancyvrb@loaded\endcsname
\let\fancyvrb@loaded\endinput
```

12.2 Errors

```
\def\FV@Error#1#2{%
  \edef\@tempc{#2}\expandafter\errhelp\expandafter{\@tempc}%
  \errmessage{FancyVerb Error:^^J\space\space #1^^J}}

\def\FV@eha{Your command was ignored. Type <return> to continue.}
```

12.3 Verbatim footnotes

Color has to be protected.
Won’t work for some definitions of `\@makefntext`. If it weren’t for `\@makefntext`, I would do this properly by defining a footnote environment.

\VerbatimFootnotes

```
%% DG/SR modification begin - Jan. 21, 1998
%% Suggested by Bernard Gaulle to solve a compatibility problem with ‘french’
%% (it introduce the restriction to put \VerbatimFootnotes AFTER the preambule)
%%\def\VerbatimFootnotes{\let\@footnotetext\V@footnotetext}
\let\V@footnote\footnote
\def\VerbatimFootnotes{%
\let\@footnotetext\V@footnotetext%
\let\footnote\V@footnote}
%% DG/SR modification end
```

\V@footnotetext

```
\long\def\V@footnotetext{%
  \afterassignment\V@@footnotetext
  \let\@tempa}
```

\V@@footnotetext

```
\def\V@@footnotetext{%
  \insert\footins\bgroup
  \csname reset@font\endcsname
  \footnotesize
  \interlinepenalty\interfootnotelinepenalty
  \splittopskip\footnotesep
  \splitmaxdepth\dp\strutbox
  \floatingpenalty \@MM
  \hsize\columnwidth
  \@parboxrestore
  \edef\@currentlabel{\csname p@footnote\endcsname\@thefnmark}%
  \@makefntext{}%
  \rule{\z@}{\footnotesep}%
  \bgroup
  \aftergroup\V@@@footnotetext
  \ignorespaces}
```

\V@@@footnotetext

```
\def\V@@@footnotetext{\strut\egroup}
```

12.4 key=value

```
\RequirePackage{keyval}
```

Extensions to keyval.sty:
`\define@booleankey{<family>}{<key>}{<iftrue>}{<iffalse>}`
Defines a `<key>` for `<family>` that executes:
`<iftrue>` clause when value begins with ‘t’ or ‘T’, or is omitted.
`<iffalse>` clause otherwise.

\define@booleankey

```
\def\define@booleankey#1#2#3#4{%
  \@namedef{KV@#1@#2@default}{#3}%
  \@namedef{KV@#1@#2@false}{#4}%
  \@namedef{KV@#1@#2}##1{\KV@booleankey{##1}{#1}{#2}}
```

\KV@booleankey

```
\def\KV@booleankey#1#2#3{%
  \edef\@tempa{#1}\expandafter\KV@@booleankey\@tempa\relax\@nil{#2}{#3}}
```

\KV@@booleankey

```
\def\KV@@booleankey#1#2\@nil#3#4{%
  \@nameuse{KV@#3@#4@\if t#1default\else\if T#1default\else false\fi\fi}}
```

\FV@None

```
\def\FV@None{none}
```

\FV@Auto

```
\def\FV@Auto{auto}
```

\fvset

```
\def\fvset#1{\setkeys{FV}{#1}}
```

`\FV@Command` is for verbatim commands. Example:
`\def\VerbatimInput{\FV@Command{}{VerbatimInput}}`
`\FV@Command{<key=value>}{<name>}`:

1. Defines `\FV@KeyValues` to be `<key=value>`.
2. Looks for $*$, and adds 'showspaces' to `\FV@KeyValues` if found.
3. Looks for `[<key=value>]` argument, and adds it to `\FV@KeyValues` if found.
4. Executes `\FVC@<name>`.

\FV@Command

```
\def\FV@Command#1#2{%
  \@ifstar
    {\def\FV@KeyValues{#1,showspaces}\FV@@Command{#2}}%
    {\def\FV@KeyValues{#1}\FV@@Command{#2}}}
```

\FV@@Command

```
\def\FV@@Command#1{%
  \@ifnextchar[%
    {\FV@GetKeyValues{\@nameuse{FVC@#1}}}%
    {\@nameuse{FVC@#1}}}
```

\FV@GetKeyValues

```
\def\FV@GetKeyValues#1[#2]{%
  \expandafter\def\expandafter\FV@KeyValues\expandafter{\FV@KeyValues,#2}#1}
```

\@CustomVerbatimCommand

```
\def\@CustomVerbatimCommand#1#2#3#4{%
  \begingroup\fvset{#4}\endgroup  % If there are errors, it easier to locate.
%% DG/SR modification begin - Jan. 13, 1998
%%  \def\@tempa##1##2\@nil{\def\@tempa{##2}}%
%%  \expandafter\@tempa\string#3\@empty\@nil
%%  \@ifundefined{FVC@\@tempa}%
  \@ifundefined{FVC@#3}%
%% DG/SR modification end
    {\FV@Error{Command `\string#3' is not a FancyVerb command.}\@eha}%
    {#1{#2}{\FV@Command{#4}{#3}}}}
```

\CustomVerbatimCommand

```
\def\CustomVerbatimCommand{\@CustomVerbatimCommand\newcommand}
```

\RecustomVerbatimCommand

```
\def\RecustomVerbatimCommand{\@CustomVerbatimCommand\renewcommand}
```

`\FV@Environment` is for verbatim environments. Example:
`\def\Verbatim{\FV@Environment{}{Verbatim}}`
`\@namedef{Verbatim*}{\FV@Environment{showspaces}{Verbatim}}`
`\FV@Environment{<key=value>}{<name>}`:

1. Defines `\FV@KeyValues` to be `<key=value>`.
2. Sets `\catcode`\^^M=13`, checks for '[', and resets `\catcode``^^M=5`.
3. Reads `[<key=value>]` and adds it to `\FV@KeyValues` if '[' was found.
4. Executes `\FVB@<name>`.

`\FV@Environment` is like `\FV@Command`, except:

- It omits step 2 (looking for $*$), and
- It sets `\catcode`\^^M=13` when checking for the '[' of the optional argument, so that it does not skip over `^^M` if there is no '['.

\FV@Environment

```
\def\FV@Environment#1#2{%
  \def\FV@KeyValues{#1}%
  \catcode`\^^M=\active
  \@ifnextchar[%
    {\catcode`\^^M=5 \FV@GetKeyValues{\@nameuse{FVB@#2}}}%
    {\catcode`\^^M=5 \@nameuse{FVB@#2}}}
```

\CustomVerbatimEnvironment

```
\def\CustomVerbatimEnvironment{\@CustomVerbatimEnvironment\newenvironment}
```

\RecustomVerbatimEnvironment

```
\def\RecustomVerbatimEnvironment{\@CustomVerbatimEnvironment\renewenvironment}
```

\@CustomVerbatimEnvironment

```
\def\@CustomVerbatimEnvironment#1#2#3#4{%
  \begingroup\fvset{#4}\endgroup  % If there are errors, it easier to locate.
  \@ifundefined{FVB@#3}%
    {\FV@Error{`#3' is not a FancyVerb environment.}\@eha}%
    {#1{#2}{\FV@Environment{#4}{#3}}{\@nameuse{FVE@#3}}%
     #1{#2*}{\FV@Environment{#4,showspaces}{#3}}{\@nameuse{FVE@#3}}}}
```

\DefineVerbatimEnvironment

```
\def\DefineVerbatimEnvironment#1#2#3{%
  \@namedef{#1}{\FV@Environment{#3}{#2}}%
  \@namedef{end#1}{\@nameuse{FVE@#2}}%
  \@namedef{#1*}{\FV@Environment{#3,showspaces}{#2}}%
  \@namedef{end#1*}{\@nameuse{FVE@#2}}}
```

Then commands or environments set key values with `\FV@UseKeyValues`, within a group.

\FV@UseKeyValues

```
\def\FV@UseKeyValues{%
  \ifx\FV@KeyValues\@empty\else
    \def\KV@prefix{KV@FV@}%
    \expandafter\KV@do\FV@KeyValues,\relax,%
    \def\FV@KeyValues{}%
  \fi}
```

12.5 Scanning

Scanning macros:

- read a line at a time from an environment or a file,
- save the line in `\FV@Line`, and
- execute `\FV@PreProcessLine`.

`\FV@PreProcessLine`:

- Determines whether line is in a user-specified ranges, and if so,
- Strips the line of a user-specified no. of tokens, and
- Executes `\FV@ProcessLine{<line>}`.

Two things are common to scanning an environment or reading a file:

- `\FV@CatCodes` $\Longrightarrow$ Change `\catcode`'s.
- `\FV@PreProcessLine` $\Longrightarrow$ See above.

12.6 Codes

\FV@CatCodes

```
\def\FV@CatCodes{%
  \let\do\@makeother\dospecials  % The usual stuff.
  \FV@ActiveWhiteSpace           % See below.
  \FV@FontScanPrep               % See below.
  \FV@CatCodesHook               % A style hook.
  \FancyVerbCodes}               % A user-defined hook.
```

\FV@ActiveWhiteSpace

```
\def\FV@ActiveWhiteSpace{%
  \catcode`\^^M=\active          % End of line
  \catcode`\ =\active            % Space
  \catcode`\^^I=\active}         % Tab
```

CM Ligatures:

Left-quote:	“	¡	¿
Letter f:	ff	fi	fl ffi ffl
Hyphen:	--	---	

\FV@CatCodesHook

```
\def\FV@CatCodesHook{}
```

\FV@AddToHook

```
\def\FV@AddToHook#1#2{%
  \expandafter\def\expandafter#1\expandafter{#1#2\relax}}
```

\FancyVerbCodes

```
\define@key{FV}{codes}[]{\def\FancyVerbCodes{#1\relax}}
\define@key{FV}{codes*}{%
  \expandafter\def\expandafter\FancyVerbCodes\expandafter{%
    \FancyVerbCodes#1\relax}}
\fvset{codes}
```

```
\define@key{FV}{commandchars}[\\\{\}]%
  {\def\@tempa{#1}%
    \ifx\@tempa\FV@None
      \let\FV@CommandChars\relax
    \else
      \FV@DefineCommandChars#1\relax\relax\relax
    \fi}
```

\FV@DefineCommandChars

```
\def\FV@DefineCommandChars#1#2#3{%
  \def\FV@CommandChars{%
    \catcode`#1=0\relax\catcode`#2=1\relax\catcode`#3=2\relax}}
```

```
\FV@AddToHook\FV@CatCodesHook\FV@CommandChars
```

\FV@CommentChar

```
\define@key{FV}{commentchar}[\%]{%
  \def\@tempa{#1}%
  \ifx\@tempa\FV@None
    \let\FV@CommentChar\relax
  \else
    \def\FV@CommentChar{\catcode`#1=14}%
  \fi}
```

```
\FV@AddToHook\FV@CatCodesHook\FV@CommentChar
```

```
\fvset{commandchars=none,commentchar=none}
```

12.7 Preprocess line

These is tedious and takes up macro space, but it doesn't slow things down much when the parameters are not used.
`start`, `stop` and `gobble` parameters:
if value is a number, then after the assignment the next token is `\relax`.
"Preprocessing the line" involves:

- Checking whether this line falls in the range specified by start or stop strings. or `firstline` or `lastline` line-numbers, or a mixture.
- Gobble'ing initial characters.

```
\define@key{FV}{firstline}{%
  \afterassignment\FV@ParseStart\@tempcnta=0#1\relax\@nil{#1}}
```

\FV@ParseStart

```
\def\FV@ParseStart#1\relax\@nil#2{%
  \ifx\@nil#1\@nil
    \edef\FancyVerbStartNum{\the\@tempcnta}%
    \let\FancyVerbStartString\relax
  \else
    \edef\FancyVerbStartString{#2}%
  \fi}
```

\KV@FV@firstline@default

```
\def\KV@FV@firstline@default{%
  \let\FancyVerbStartNum\z@
  \let\FancyVerbStartString\relax}
```

```
\define@key{FV}{lastline}{%
  \afterassignment\FV@ParseStop\@tempcnta=0#1\relax\@nil{#1}}
```

\FV@ParseStop

```
\def\FV@ParseStop#1\relax\@nil#2{%
  \ifx\@nil#1\@nil
    \edef\FancyVerbStopNum{\the\@tempcnta}%
    \let\FancyVerbStopString\relax
  \else
    \edef\FancyVerbStopString{#2}%
  \fi}
```

\KV@FV@lastline@default

```
\def\KV@FV@lastline@default{%
  \let\FancyVerbStopNum\z@
  \let\FancyVerbStopString\relax}
```

```
\fvset{firstline,lastline}
```

```
\newcount\FV@CodeLineNo
```

If `\FV@FindStartStop` determines the line should be printed, it executes `\FV@@PreProcessLine`

\FV@PreProcessLine

```
\def\FV@PreProcessLine{%
  \global\advance\FV@CodeLineNo\@ne
  \FV@FindStartStop}
```

\FV@@PreProcessLine

```
\def\FV@@PreProcessLine{%
  \FV@StepLineNo
  \FV@Gobble
  \expandafter\FV@ProcessLine\expandafter{\FV@Line}}
```

The definition of `\FV@FindStartStop` changes, so that we don't have to check irrelevant conditions with each line.

Here's how it works; for simplicity we assume `firstline` and `lastline` have been specified, rather than start or stop strings.

The first time `\FV@FindStartStop` is called:

1. It calls `\FV@DefineFindStart`. (The name roughly means "define `\FV@FindStartStop` to detect when we've reached the start of the specified range".) This checks the range parameters that have been specified, and then redefines `\FV@FindStartStop` (yes, the function that has called this code) to contain only the necessary checks, for efficiency reasons:

   - If we haven't printed any lines yet, which is true when `\FV@FindStartStop` is first called, we only need to check the current line-number against `firstline`, so we redefine `\FV@FindStartStop` to be `\FV@FindStartNum`.
   - If `firstline` isn't set, we only have to check the current line-number against `lastline`, so we redefine `\FV@FindStartStop` to be `\FV@DefineFindStop`. (The name roughly means "define `\FV@FindStartStop` to detect when we've reached the stop (end) of the specified range", but unlike the START case, it includes code (`\FV@@PreProcessLine`) to print the current line.)

2. It then calls the newly-defined `\FV@FindStartStop` to see if the current line being processed is within the range to be printed.

`\FV@FindStartStop` is redefined once as described above, but it can be redefined again:

1. When we have reached `firstline`, i.e. have entered the range to be printed, we redefine `\FV@FindStartStop` to be `\FV@DefineFindStop`. (We know we've reached the start, so there's no point checking that again on future calls.)
2. When we reach `lastline` and have printed it, we won't want to print any more lines, so we redefine `\FV@FindStartStop` to `\relax` so we do nothing (and do it efficiently).

   Note that when `firstline` = `lastline` (= 6, say), it is `\FV@FindStartNum` (which `\FV@FindStartStop` has initially been redefined to) that notices when we reach line 6, and prints it, and redefines `\FV@FindStartStop` to be `\FV@DefineFindStop`. Then we read line 7, so `\FV@DefineFindStop` has to check whether we've passed `lastline`, and therefore shouldn't print the current line, before redefining `\FV@FindStartStop` to `\relax` as explained above.

   By contrast, when `firstline` is less than `lastline` (= 6, 8, say), when we reach line 8, `\FV@DefineFindStop` sees that we're at the end of the range, and *does* print the line, before redefining `\FV@FindStartStop` to `\relax` as above.

   This slight weirdness is because several of the macros are defining themselves and one another, resulting in a slightly weird execution flow.

`\FV@FindStartStop`

```
\def\FV@FindStartStop{\FV@DefineFindStart\FV@FindStartStop}
```

```
%% \def\FV@DefinePreProcessLine{%
%%   \setcounter{FancyVerbLine}{0}%
%%   \FV@DefineFindStart}
```

`\FV@DefineFindStart`

```
\def\FV@DefineFindStart{%
  \ifx\FancyVerbStartString\relax
    \ifnum\FancyVerbStartNum<\tw@
      \FV@DefineFindStop
    \else
      \let\FV@FindStartStop\FV@FindStartNum
    \fi
  \else
    \let\FV@FindStartStop\FV@FindStartString
  \fi}
```

\FV@FindStartNum

```
\def\FV@FindStartNum{%
  \ifnum\FancyVerbStartNum>\FV@CodeLineNo\else
    \FV@DefineFindStop
    \expandafter\FV@@PreProcessLine
  \fi}
```

\FV@FindStartString

```
%% SR modification begin - 1996
\def\FV@FindStartString{%
 \expandafter\FV@@FindStartString
{\meaning\FV@Line}%
{\meaning\FancyVerbStartString}%
}
```

\FV@@FindStartString

```
\def\FV@@FindStartString#1#2{%
\edef\@fooA{#1}\edef\@fooB{#2}%
  \ifx\@fooA\@fooB
    \FV@DefineFindStop
  \fi
}
%% SR modification end
```

\FV@DefineFindStop

```
\def\FV@DefineFindStop{%
  \ifx\FancyVerbStopString\relax
    \ifnum\FancyVerbStopNum<\@ne
      \let\FV@FindStartStop\FV@@PreProcessLine
    \else
      \let\FV@FindStartStop\FV@FindStopNum
    \fi
  \else
    \let\FV@FindStartStop\FV@FindStopString
  \fi}
```

\FV@FindStopNum

```
\def\FV@FindStopNum{%
  \ifnum\FancyVerbStopNum>\FV@CodeLineNo
  \else
    \let\FV@FindStartStop\relax
    \ifeof\FV@InFile\else
      \immediate\closein\FV@InFile
    \fi
  \fi
  \ifnum\FancyVerbStopNum<\FV@CodeLineNo
  \else
    \FV@@PreProcessLine
  \fi}
```

\FV@FindStopString

```
%% SR modification begin - 1996
\def\FV@FindStopString{%
 \expandafter\FV@@FindStopString
{\meaning\FV@Line}%
{\meaning\FancyVerbStopString}%
}
```

\FV@@FindStopString

```
\def\FV@@FindStopString#1#2{%
\edef\@fooA{#1}\edef\@fooB{#2}%
  \ifx\@fooA\@fooB
    \let\FV@FindStartStop\relax
    \ifeof\FV@InFile\else
      \immediate\closein\FV@InFile
    \fi
  \else
    \expandafter\FV@@PreProcessLine
  \fi}
%% SR modification end
```

Gobblings. `\FV@Gobble` does nothing, or strips some tokens from the line and stores the result in `\FV@Line` again. We use LaTeX's `\renewcommand` to define a command for gobbling up to 9 arguments. This is not the same as removing 9 tokens, but is easier.

\FV@@Gobble

```
\def\FV@@Gobble{%
  \expandafter\expandafter\expandafter\FV@@@Gobble
  \expandafter\FV@@@@Gobble\FV@Line
    \@nil\@nil\@nil\@nil\@nil\@nil\@nil\@nil\@nil\@nil\@@nil}
```

\FV@@@Gobble

```
\def\FV@@@Gobble#1\@nil#2\@@nil{\def\FV@Line{#1}}
```

\FV@Gobble

```
\define@key{FV}{gobble}{%
  \@tempcnta=#1\relax
  \ifnum\@tempcnta<\@ne
    \let\FV@Gobble\relax
  \else
    \ifnum\@tempcnta>9
      \FV@Error{gobble parameter must be less than 10}\FV@eha
    \else
      \renewcommand{\FV@@@@Gobble}[\@tempcnta]{}%
      \let\FV@Gobble\FV@@Gobble
    \fi
  \fi}
```

\FV@@@@Gobble

```
\def\FV@@@@Gobble{}
```

\KV@FV@gobble@default

```
\def\KV@FV@gobble@default{\let\FV@Gobble\relax}
```

```
\fvset{gobble}
```

12.8 Scanning environments

\FV@Scan

```
\def\FV@Scan{%
    \FV@CatCodes
    \VerbatimEnvironment
    \FV@DefineCheckEnd
    \FV@BeginScanning}
```

`\VerbatimEnvironment`:

This saves the name of the current environment as `\FV@EnvironName`, if the latter is not already defined. Then `\FV@CheckEnd` knows how to find the end as long as either:

- `\begin` and `\end` are not used within the definition of the environment, OR
- `\VerbatimEnvironment` is used in the definition before the first `\begin`.

\VerbatimEnvironment

```
\def\VerbatimEnvironment{%
  \ifx\FV@EnvironName\relax\xdef\FV@EnvironName{\@currenvir}\fi}
```

```
\let\FV@EnvironName\relax
```

12.9 Check end

We have to check the argument of the first `\end{}` in each line, compare it with `\FV@EnvironName`, and return `\iftrue` if it matches and `\iffalse` otherwise.

There are four cases (R=regular):

case	: i	ii	iii	iv
catcode of \	: R	12	12	R
catcode of {}	: R	R	12	12

For uniformity, we use `![]` instead of `\{}` in all the definitions.

We first set the catcodes of `\{}` to those in effect in the verbatim environment. Then we define:

`!def!FV@CheckEnd#1[!FV@@CheckEnd#1\end{}!@nil]`

If have their usual catcodes, we define:

`!def!FV@@CheckEnd#1\end#2#3!@nil[!def!@tempa[#2]]`

If have catcode 12, we define:

```
!def!FV@@CheckEnd#1\end{#2}#3!@nil[!def!@tempa[#2]]
```

```
\begingroup
\catcode`\!=0
\catcode`\[=1
\catcode`\]=2
```

Case i:

```
!gdef!FV@CheckEnd@i#1[!FV@@CheckEnd#1\end{}!@nil]
!gdef!FV@@CheckEnd@i#1\end#2#3!@nil[!def!@tempa[#2]!def!@tempb[#3]]
!gdef!FV@@@CheckEnd@i[\end{}]
```

Case ii:

```
\catcode`!\=12

!gdef!FV@CheckEnd@ii#1[!FV@@CheckEnd#1\end{}!@nil]
!gdef!FV@@CheckEnd@ii#1\end#2#3!@nil[!def!@tempa[#2]!def!@tempb[#3]]
!gdef!FV@@@CheckEnd@ii[\end{}]
```

Case iii:

```
!catcode`!{=12
!catcode`!}=12

!gdef!FV@CheckEnd@iii#1[!FV@@CheckEnd#1\end{}!@nil]
!gdef!FV@@CheckEnd@iii#1\end{#2}#3!@nil[!def!@tempa[#2]!def!@tempb[#3]]
!gdef!FV@@@CheckEnd@iii[\end{}]
```

Case iv:

```
!catcode`!\=0

!gdef!FV@CheckEnd@iv#1[!FV@@CheckEnd#1\end{}!@nil]
!gdef!FV@@CheckEnd@iv#1\end{#2}#3!@nil[!def!@tempa[#2]!def!@tempb[#3]]
!gdef!FV@@@CheckEnd@iv[\end{}]
```

```
\endgroup
```

\FV@BadCodes

```
\def\FV@BadCodes#1{%
  \FV@Error
    {\string\catcode\space of \expandafter\@gobble\string#1 is wrong:
    \the\catcode`#1}%
    {Only the following catcode values are allowed:
    ^^J\@spaces \expandafter\@gobble\string\\ \space\space --> 0 or 12.
    ^^J\@spaces \string{ \string} --> 1 and 2, resp., or both 12.
    ^^JTo get this error, either you are a hacker or you got bad advice.}%
  \def\FV@CheckEnd##1{\iftrue}}
```

\FV@DefineCheckEnd

```
\def\FV@DefineCheckEnd{%
  \ifnum\catcode`\\=\z@
```

```
    \ifnum\catcode`\{=\@ne
      \let\FV@CheckEnd\FV@CheckEnd@i
      \let\FV@@CheckEnd\FV@@CheckEnd@i
      \let\FV@@@CheckEnd\FV@@@CheckEnd@i
    \else
      \ifnum\catcode`\{=12
        \let\FV@CheckEnd\FV@CheckEnd@iv
        \let\FV@@CheckEnd\FV@@CheckEnd@iv
        \let\FV@@@CheckEnd\FV@@@CheckEnd@iv
      \else
        \FV@BadCodes\{%
      \fi
    \fi
  \else
    \ifnum\catcode`\\=12
      \ifnum\catcode`\{=\@ne
        \let\FV@CheckEnd\FV@CheckEnd@ii
        \let\FV@@CheckEnd\FV@@CheckEnd@ii
        \let\FV@@@CheckEnd\FV@@@CheckEnd@ii
      \else
        \ifnum\catcode`\{=12
          \let\FV@CheckEnd\FV@CheckEnd@iii
          \let\FV@@CheckEnd\FV@@CheckEnd@iii
          \let\FV@@@CheckEnd\FV@@@CheckEnd@iii
        \else
          \FV@BadCodes\{%
        \fi
      \fi
    \else
      \FV@BadCodes\\%
    \fi
  \fi}
```

12.10 Line-by-line scanning

We first skip everything after the beginning of the environment.

\FV@BeginScanning

```
\begingroup
\catcode`\^^M=\active
  \gdef\FV@BeginScanning#1^^M{%
    \def\@tempa{#1}\ifx\@tempa\@empty\else\FV@BadBeginError\fi%
    \FV@GetLine}%
\endgroup
```

\FV@BadBeginError

```
\def\FV@BadBeginError#1{%
  \expandafter\@temptokena\expandafter{\@tempa}%
  \FV@Error
```

```
   {Extraneous input `\the\@temptokena' between
     \string\begin{\FV@EnvironName}[<key=value>] and line end}%
  {This input will be discarded. Hit <return> to continue.}}
```

If `\FancyVerbGetLine` does not find a `^^M`, then we are at the end of the file, and `\FV@EOF` attempts to terminate the document. Otherwise, `\FV@EOF` is gobbled by `\FancyVerbGetLine`.

`\FV@GetLine`

```
%% DG/SR modification begin - May. 18, 1998 (added code to turn off ligatures)
%% \def\FV@GetLine{\expandafter\FV@CheckScan\FancyVerbGetLine}
\def\FV@GetLine{\@noligs\expandafter\FV@CheckScan\FancyVerbGetLine}
%% DG/SR modification end
```

`\FancyVerbGetLine`

```
\begingroup
\catcode`\^^M=\active
\gdef\FancyVerbGetLine#1^^M{%
  \@nil%
  \FV@CheckEnd{#1}%
  \ifx\@tempa\FV@EnvironName%            % True if end is found
    \ifx\@tempb\FV@@@CheckEnd\else\FV@BadEndError\fi%
    \let\next\FV@EndScanning%
  \else%
    \def\FV@Line{#1}%
    \def\next{\FV@PreProcessLine\FV@GetLine}%
  \fi%
  \next}%
\endgroup
```

`\FV@BadEndError`

```
\def\FV@BadEndError{%
  \expandafter\@temptokena\expandafter{\@tempb}%
  \FV@Error
     {Extraneous input `\the\@temptokena' between
       \string\end{\FV@EnvironName} and line end}%
     {This input will be discarded. Type <return> to continue.}}
```

`\FV@EndScanning`

```
\def\FV@EndScanning{%
  \edef\next{\noexpand\end{\FV@EnvironName}}%
  \global\let\FV@EnvironName\relax
  \next}
```

```
\@ifundefined{@currenvline}{\let\@currenvline\@empty}{}
```

`\FV@CheckScan`

```
\def\FV@CheckScan#1{\@ifnextchar\@nil{\@gobble}{\FV@EOF}}
\def\FV@CheckScan#1{\ifx\@nil#1\@empty\else\expandafter\FV@EOF\fi}
```

\FV@EOF

```
\def\FV@EOF{%
  \FV@Error{Couldn’t find ‘\string\end{\FV@EnvironName}’ to end
    a verbatim environment\@currenvline.}%
    {Probably you mistyped the environment name or included an extraneous
    ^^Jspace, or are using an improperly defined verbatim environment.
    ^^JHit return and I will try to terminate this job.}%
  \FV@EndScanning
  \end{document}}
```

12.11 Input

```
\newread\FV@InFile
```

\FV@Input

```
\def\FV@Input#1{%
  \immediate\openin\FV@InFile #1\relax
  \ifeof\FV@InFile
    \FV@Error{No verbatim file #1}\FV@eha
    \immediate\closein\FV@InFile
  \else
    \FV@CatCodes
    \expandafter\FV@@Input
  \fi}
```

TeX reports EOF when reading after the last newline character.
Thus, we read to InLine, and if TeX reports EOF:

- If InLine=Empty, previous line was last line in file (file ends nl).
- Otherwise, InLine holds last line in file (file doesn’t end in nl).

`\FV@@Input` handles both cases correctly.

\FV@@Input

```
\def\FV@@Input{%
  \def\FV@Line{}%
  \FV@ReadLine
  \ifeof\FV@InFile
    \ifx\FV@Line\@empty\else
      \FV@PreProcessLine
    \fi
    \immediate\closein\FV@InFile
  \else
    \FV@PreProcessLine
    \expandafter\FV@@Input
  \fi}
```

We also want to handle true comment characters correctly. This means that we keep accumulating text in InLine until we find a `^^M` (indicating that the line did not contain a comment character).

\FV@ReadLine

```
\begingroup
\catcode`\^^M=\active
\gdef\FV@ReadLine{%
  \ifeof\FV@InFile\else
    \immediate\read\FV@InFile to\@tempa%
    \expandafter\FV@@ReadLine\@tempa^^M\relax^^M\@nil%
  \fi}
```

#2 is empty if line ends in `^^M`, #2=`\relax` otherwise

\FV@@ReadLine

```
\gdef\FV@@ReadLine#1^^M#2^^M#3\@nil{%
  \expandafter\def\expandafter\FV@Line\expandafter{%
    \FV@Line#1}%
  \ifx\relax#2\@empty\expandafter\FV@ReadLine\fi}%
\endgroup
```

12.12 Formatting – Common components

Some things that are common to all verbatim formatting:

\FV@FormattingPrep

```
\def\FV@FormattingPrep{%
  \global\FV@CodeLineNo\z@
  \frenchspacing              % Cancels special punctuation spacing.
  \FV@SetupFont               % See below.
  \FV@DefineWhiteSpace        % See below.
  \FancyVerbDefineActive
  \FancyVerbFormatCom}        % A user-defined hook (formatcom parameter).
```

Fonts

```
\expandafter\ifx\csname selectfont\endcsname\relax
```

\FV@SetupFont

```
\def\FV@SetupFont{%
  \FV@BaseLineStretch
  \ifx\@currsize\small\normalsize\else\small\fi\@currsize
  \FV@FontSize
  \FV@FontFamily}

\else

\def\FV@SetupFont{%
  \FV@BaseLineStretch
  \FV@FontSize
  \FV@FontFamily
  \FV@FontSeries
  \FV@FontShape
  \selectfont
%% DG/SR modification begin - May. 18, 1998 (added code to turn off ligatures)
  \@noligs}
%% DG/SR modification end
```

```
\fi
```

\FV@FontSize

```
\define@key{FV}{fontsize}{%
  \def\@tempa{#1}%
  \ifx\@tempa\FV@Auto
    \let\FV@FontSize\relax
  \else
    \def\FV@FontSize{#1}%
  \fi}
```

\KV@FV@fontsize@default

```
\def\KV@FV@fontsize@default{\let\FV@FontSize\relax}
```

\FV@BaseLineStretch

```
\define@key{FV}{baselinestretch}[auto]{%
  \def\@tempa{#1}%
  \ifx\@tempa\FV@Auto
    \let\FV@BaseLineStretch\relax
  \else
    \def\FV@BaseLineStretch{\def\baselinestretch{#1}}%
  \fi}
```

V@FV@baselinestretch@default

```
\def\KV@FV@baselinestretch@default{\let\FV@BaseLineStretch\relax}
```

```
\define@key{FV}{fontfamily}{%
  \@ifundefined{FV@fontfamily@#1}%
    {\def\FV@FontScanPrep{}\def\FV@FontFamily{\fontfamily{#1}}}
    {\csname FV@fontfamily@#1\endcsname}}
```

\FV@FontSeries

```
\define@key{FV}{fontseries}{%
  \def\@tempa{#1}%
  \ifx\@tempa\FV@Auto
    \let\FV@FontSeries\relax
  \else
    \def\FV@FontSeries{\fontseries{#1}}%
  \fi}
```

\FV@FontShape

```
\define@key{FV}{fontshape}{%
  \def\@tempa{#1}%
  \ifx\@tempa\FV@Auto
    \let\FV@FontShape\relax
  \else
    \def\FV@FontShape{\fontshape{#1}}%
  \fi}
```

Font family styles have to define `\FV@FontScanPrep` and `\FV@FontFamily`.

`\FV@MakeActive`

```
\def\FV@MakeActive#1{%
  \catcode`#1=\active
  \def\next##1{\expandafter\def\expandafter\FV@MakeUnActive\expandafter{%
    \FV@MakeUnActive\def##1{\string##1}}}%
  \begingroup\lccode`~=`#1\relax\expandafter\next\expandafter~\endgroup}
```

`\FV@MakeUnActive`

```
\def\FV@MakeUnActive{}
```

```
\begingroup
\catcode`\`=\active
```

`\FV@fontfamily@tt`

```
\gdef\FV@fontfamily@tt{%
  \def\FV@FontScanPrep{\FV@MakeActive\`}%
%% SR modification begin - 1995
%%  \def\FV@FontFamily{\tt`{{\string`}}}}
  \def\FV@FontFamily{\ttfamily\edef`{{\string`}}}}
%% SR modification end
```

`\FV@fontfamily@cmtt`

```
\gdef\FV@fontfamily@cmtt{%
  \def\FV@FontScanPrep{\FV@MakeActive\`}%
  \def\FV@FontFamily{\edef`{{\string`}}\fontfamily{cmtt}}}
```

```
\endgroup
```

`\FV@fontfamily@cmtt-spanish`

```
\@namedef{FV@fontfamily@cmtt-spanish}{%
  \def\FV@FontScanPrep{}%
  \def\FV@FontFamily{\fontfamily{cmtt}}}
```

Fix me

`\FV@fontfamily@courier`

```
\@namedef{FV@fontfamily@courier}{%
  \def\FV@FontScanPrep{}%
%% SR modification begin - 1995
%%  \def\FV@FontFamily{\fontfamily{rpcr}}}
  \def\FV@FontFamily{\fontfamily{pcr}}}
%% SR modification end
```

\FV@fontfamily@helvetica

```
\@namedef{FV@fontfamily@helvetica}{%
  \def\FV@FontScanPrep{}%
%% SR modification begin - 1995
%%  \def\FV@FontFamily{\fontfamily{rphv}}}
  \def\FV@FontFamily{\fontfamily{phv}}}
%% SR modification end
```

```
\fvset{fontfamily=tt,fontsize=auto,fontshape=auto,fontseries=auto,
  baselinestretch=auto}
```

\FV@DefineWhiteSpace

We just define the active characters to be ordinary commands, which are easier to redefine. We do with any macros that use verbatim text.

```
\begingroup
\catcode`\ =\active
\catcode`\^^I=\active
```

\FV@DefineWhiteSpace

```
\gdef\FV@DefineWhiteSpace{\def {\FV@Space}\def^^I{\FV@Tab}}%
```

```
\endgroup
```

\FancyVerbDefineActive

```
\define@key{FV}{defineactive}[]{\def\FancyVerbDefineActive{#1\relax}}
\define@key{FV}{defineactive*}{%
  \expandafter\def\expandafter\FancyVerbDefineActive\expandafter{%
    \FancyVerbDefineActive#1\relax}}
\fvset{defineactive}
```

\FV@Space:

\FV@Space

```
\define@booleankey{FV}{showspaces}%
  {\def\FV@Space{{\FancyVerbSpace}}}%
  {\def\FV@Space{\ }}
```

```
{\catcode`\ =12 \gdef\FancyVerbSpace{\tt }}
```

```
\fvset{showspaces=false}
```

\FV@Tab:

\FV@Tab

```
\def\FV@Tab{\hbox to\FancyVerbTabSize\fontdimen2\font{\hss\FV@TabChar}}
```

\FancyVerbTabSize

```
\define@key{FV}{tabsize}{%
  \@tempcnta=#1\relax
  \ifnum\@tempcnta>100
    \FV@Error{Tab size too large: `\the\@tempcnta'. (Max size = 100)}\FV@eha
  \else
    \edef\FancyVerbTabSize{\the\@tempcnta}%
  \fi}
```

\FV@TabChar

```
\define@booleankey{FV}{showtabs}%
  {\def\FV@TabChar{\FancyVerbTab}}%
  {\let\FV@TabChar\relax}
```

```
\fvset{tabsize=8,showtabs=false}
```

Here is a weak attempt at a tab character. It may exceed the width of a space character when the verbatim font is small. The only proper way to do this is making it part of the verbatim font.

\FancyVerbTab

```
\def\FancyVerbTab{%
    \valign{%
      \vfil##\vfil\cr
      \hbox{$\scriptscriptstyle-$}\cr
      \hbox to 0pt{\hss$\scriptscriptstyle\rangle\mskip -.8mu$}\cr
      \hbox{$\scriptstyle\mskip -3mu\mid\mskip -1.4mu$}\cr}}
```

Obey Tabs:

```
\newbox\FV@TabBox
```

\FV@@ObeyTabsInit

```
\def\FV@@ObeyTabsInit{%
  \@tempdimb=\FancyVerbTabSize\fontdimen\tw@\font
  \edef\FV@ObeyTabSize{\number\@tempdimb}%
  \advance\@tempdimb\fontdimen\tw@\font
  \advance\@tempdimb-\FancyVerbTabSize sp  % Allow for rounding errors.
  \edef\FV@@ObeyTabSize{\number\@tempdimb}%
  \let\FV@ObeyTabs\FV@@ObeyTabs
  \let\FV@Tab\FV@TrueTab}
```

\FV@@ObeyTabs

```
\def\FV@@ObeyTabs#1{\setbox\FV@TabBox=\hbox{#1}\box\FV@TabBox}
```

```
\let\FV@ObeyTabs\relax
```

\FV@TrueTab

```
\def\FV@TrueTab{%
  \egroup
  \@tempdima=\FV@ObeyTabSize sp\relax
  \@tempcnta=\wd\FV@TabBox
  \advance\@tempcnta\FV@@ObeyTabSize\relax
  \divide\@tempcnta\@tempdima
  \multiply\@tempdima\@tempcnta
  \advance\@tempdima-\wd\FV@TabBox
  \setbox\FV@TabBox=\hbox\bgroup
    \unhbox\FV@TabBox\kern\@tempdima\hbox to\z@{\hss\FV@TabChar}}
```

\FV@ObeyTabsInit

```
\define@booleankey{FV}{obeytabs}%
  {\let\FV@ObeyTabsInit\FV@@ObeyTabsInit}%
  {\let\FV@ObeyTabsInit\relax}
```

```
\fvset{obeytabs=false}
```

\FancyVerbFormatCom

\FancyVerbFormatCom

```
\define@key{FV}{formatcom}[]{\def\FancyVerbFormatCom{#1\relax}}
\define@key{FV}{formatcom*}{%
  \expandafter\def\expandafter\FancyVerbFormatCom\expandafter{%
    \FancyVerbFormatCom#1\relax}}
```

```
\fvset{formatcom}
```

\FancyVerbFormatLine

```
\def\FancyVerbFormatLine#1{\FV@ObeyTabs{#1}}
```

12.13 List environments

Some parameters:

\FV@XLeftMargin

```
\define@key{FV}{xleftmargin}{\def\FV@XLeftMargin{#1}}
\let\FV@XLeftMargin\z@
```

\FV@XRightMargin

```
\define@key{FV}{xrightmargin}{\def\FV@XRightMargin{#1}}
\let\FV@XRightMargin\z@
```

\if@FV@ResetMargins

```
\define@booleankey{FV}{resetmargins}%
  {\let\if@FV@ResetMargins\iftrue}
  {\let\if@FV@ResetMargins\iffalse}
\fvset{resetmargins=false}
```

\FV@ListParameterHook

```
\define@key{FV}{listparameters}{\def\FV@ListParameterHook{#1}}
\def\FV@ListParameterHook{}
```

\FancyVerbHFuzz

```
\define@key{FV}{hfuzz}{%
  \@tempdima=#1\relax
  \edef\FancyVerbHFuzz{\number\@tempdima sp}}
\fvset{hfuzz=2pt}
```

\FV@InterLinePenalty

```
\define@booleankey{FV}{samepage}%
  {\def\FV@InterLinePenalty{\interlinepenalty\@M}}%
  {\let\FV@InterLinePenalty\relax}
\fvset{samepage=false}
```

Lists:

`\FV@List{}` is a rewriting of `\list{}{}\item[]`. The rewrite gives me more control. I might not get the vertical spacing exactly the same, but it is more likely that it will get better than worse.

The verbatim environment consists of a series of `\hbox`'s inserted in vertical mode.

We need to take care of the following:

- leftmargin
- rightmargin
- topskip
- botskip
- toppenalty
- botpenalty
- interlinepenalties

Note: A verbatim environment immediately after an `\item` starts on the same line as the `\item`'s label, unless we reset margins. The user can instead have the environment start on a new line by inserting '`\ `' between `\item` and the environment.

\FV@List

```
\def\FV@List#1{%
  \begingroup
  \FV@UseKeyValues
  \FV@LeaveVMode
  \if@inlabel\else\setbox\@labels=\box\voidb@x\fi
  \FV@ListNesting{#1}%
  \FV@ListParameterHook
  \FV@ListVSpace
  \FV@SetLineWidth
  \FV@InterLinePenalty
  \let\FV@ProcessLine\FV@ListProcessLine@i
  \FV@CatCodes
  \FV@FormattingPrep
  \FV@ObeyTabsInit
  \FV@BeginListFrame}
```

Cases where we need to leave vmode:

- After an in-line section (\if@noskipsec=T).
- After an \item command, if we reset margins.

Then we end \vmode, using @noparlist as a flag if in vmode. (Not the usual meaning of @noparlist.)

\FV@LeaveVMode

```
\def\FV@LeaveVMode{%
  \if@noskipsec
    \leavevmode
  \else
    \if@FV@ResetMargins\if@inlabel\leavevmode\fi\fi
  \fi
  \ifvmode\@noparlisttrue\else\@noparlistfalse\unskip\par\fi}
```

\FV@ListNesting

```
\def\FV@ListNesting#1{%
  \if@FV@ResetMargins
    \@listdepth=\z@
  \else
    \ifnum\@listdepth>5\relax
      \@toodeep
    \else
      \advance\@listdepth\@ne
    \fi
  \fi
  \rightmargin\z@
  \csname @list\romannumeral\the\@listdepth\endcsname
  \ifnum#1=\z@
    \rightmargin\z@
    \leftmargin\z@
  \fi}
```

\FV@ListVSpace contains selected parts of \@trivlist and \@item.
Here are the cases:
Vmode not in label or after @NOBREAK:

<topskip>	=	\topsep + \partopsep + \parskip
<botskip>	=	\topsep + \partopsep
<toppenalty>	=	\@beginparpenalty
<botpenalty>	=	\@endparpenalty

Vmode in label:

<topskip>	=	\parskip	% Expected anyway.
<botskip>	=	\topsep + \partopsep	% Omitted in LaTeX – a bug?
<toppenalty>	=	None	
<botpenalty>	=	\@endparpenalty	% Ditto.

Vmode after `@nobreak`:

`<topskip>`	= `\parskip`
`<botskip>`	= `\topsep + \partopsep`
`<toppenalty>`	= None
`<botpenatly>`	= `\@endparpenalty`

Hmode:

`<topskip>`	= `\topsep + \parskip`
`<botskip>`	= `\topsep`
`<toppenalty>`	= `\@beginparpenalty`
`<botpenatly>`	= `\@endparpenalty`

Notes:

- Except when in label or after nobreak, `\parskip` is added with `\addvspace`, so that net space is:

  `MAX{\topsep (+\partopsep) + \parskip , \lastskip }`

  (The usual `\@item` works the same way.)

- `\parskip` is added afterwards by a new paragraph, if any.

- `<botskip> == \@topsepadd`

`\FV@ListVSpace`

```
\def\FV@ListVSpace{%
  \@topsepadd\topsep
  \if@noparlist\advance\@topsepadd\partopsep\fi
  \if@inlabel
    \vskip\parskip
  \else
    \if@nobreak
      \vskip\parskip
      \clubpenalty\@M
    \else
      \addpenalty\@beginparpenalty
      \@topsep\@topsepadd
      \advance\@topsep\parskip
      \addvspace\@topsep
    \fi
  \fi
  \global\@nobreakfalse
  \global\@inlabelfalse
  \global\@minipagefalse
  \global\@newlistfalse}
```

`\leftmargin`	:= totalleftmargin
`\rightmargin`	:= totalrightmargin
`\@totalleftmargin`	:= totalleftmargin of enclosing environment.

`\FV@SetLineWidth`

```
\def\FV@SetLineWidth{%
  \if@FV@ResetMargins\else
    \advance\leftmargin\@totalleftmargin
  \fi
  \advance\leftmargin\FV@XLeftMargin\relax
  \advance\rightmargin\FV@XRightMargin\relax
  \linewidth\hsize
  \advance\linewidth-\leftmargin
  \advance\linewidth-\rightmargin
  \hfuzz\FancyVerbHFuzz\relax}
```

We have to insert the right interline penalties (`\interlinepenalty`, `\clubpenalty`, `\widowpenalty`). We could process the environment as one long paragraph and let TeX insert the penalties, but this might cause problems for a very long environment.

Line 1 : Insert `\@labels` (maybe) plus current_line

Line 2 : Save current line

Line 3 : penalty = interline + club ; Insert last line ; Save current line.

Line 4 : penalty = interline ; Insert last line ; Save current line.

Then at the end:

Next line = 1 : Add null line.

Next line = 2 : Nothing.

Next line = 3 : penalty = interline + club + widow ; Insert last line.

Next line = 4 : penalty = interline + widow ; Insert last line.

`\FV@ListProcessLine`

```
\def\FV@ListProcessLine#1{%
  \hbox to \hsize{%
    \kern\leftmargin
    \hbox to \linewidth{%
      \FV@LeftListNumber
      \FV@LeftListFrame
      \FancyVerbFormatLine{#1}\hss
%% DG/SR modification begin - Jan. 28, 1998 (for numbers=right add-on)
%%      \FV@RightListFrame}%
      \FV@RightListFrame
      \FV@RightListNumber}%
%% DG/SR modification end
    \hss}}
```

\FV@ListProcessLine@i

```
\def\FV@ListProcessLine@i#1{%
  \hbox{%
    \ifvoid\@labels\else
      \hbox to \z@{\kern\@totalleftmargin\box\@labels\hss}%
    \fi
    \FV@ListProcessLine{#1}}%
  \let\FV@ProcessLine\FV@ListProcessLine@ii}
```

\FV@ListProcessLine@ii

```
\def\FV@ListProcessLine@ii#1{%
  \setbox\@tempboxa=\FV@ListProcessLine{#1}%
  \let\FV@ProcessLine\FV@ListProcessLine@iii}
```

\FV@ListProcessLine@iii

```
\def\FV@ListProcessLine@iii#1{%
  {\advance\interlinepenalty\clubpenalty\penalty\interlinepenalty}%
  \box\@tempboxa
  \setbox\@tempboxa=\FV@ListProcessLine{#1}%
  \let\FV@ProcessLine\FV@ListProcessLine@iv}
```

\FV@ListProcessLine@iv

```
\def\FV@ListProcessLine@iv#1{%
  \penalty\interlinepenalty
  \box\@tempboxa
  \setbox\@tempboxa=\FV@ListProcessLine{#1}}%
```

\FV@EndList

```
\def\FV@EndList{%
  \FV@ListProcessLastLine
  \FV@EndListFrame
  \@endparenv
  \endgroup
  \@endpetrue}
```

\FV@ListProcessLastLine

```
\def\FV@ListProcessLastLine{%
  \ifx\FV@ProcessLine\FV@ListProcessLine@iv
    {\advance\interlinepenalty\widowpenalty\penalty\interlinepenalty}%
    \box\@tempboxa
  \else
    \ifx\FV@ProcessLine\FV@ListProcessLine@iii
      {\advance\interlinepenalty\widowpenalty
        \advance\interlinepenalty\clubpenalty
        \penalty\interlinepenalty}%
      \box\@tempboxa
    \else
      \ifx\FV@ProcessLine\FV@ListProcessLine@i
        \FV@Error{Empty verbatim environment}{}%
        \FV@ProcessLine{}%
      \fi
    \fi
  \fi}
```

Verbatim environment:

Verbatim

```
\def\FV@VerbatimBegin{\FV@List\z@}
\def\FV@VerbatimEnd{\FV@EndList}
```

\FVB@Verbatim

```
\def\FVB@Verbatim{\FV@VerbatimBegin\FV@Scan}
```

\FVE@Verbatim

```
\def\FVE@Verbatim{\FV@VerbatimEnd}
```

```
\DefineVerbatimEnvironment{Verbatim}{Verbatim}{}
```

With `\UseVerbatim`, we have to take care of some of the things `\end{}` would do.

\FV@UseVerbatim

```
\def\FV@UseVerbatim#1{%
  \FV@VerbatimBegin#1\FV@VerbatimEnd
  \@doendpe\global\@ignorefalse\ignorespaces}
```

\VerbatimInput

```
\def\VerbatimInput{\FV@Command{}{VerbatimInput}}
```

\FVC@VerbatimInput

```
\def\FVC@VerbatimInput#1{\FV@UseVerbatim{\FV@Input{#1}}}
```

LVerbatim environment:

LVerbatim

```
\def\FV@LVerbatimBegin{\FV@List\@ne}
\def\FV@LVerbatimEnd{\FV@EndList}
```

\FVB@LVerbatim

```
\def\FVB@LVerbatim{\FV@LVerbatimBegin\FV@Scan}
```

\FVE@LVerbatim

```
\def\FVE@LVerbatim{\FV@LVerbatimEnd}
```

```
\DefineVerbatimEnvironment{LVerbatim}{LVerbatim}{}
```

\FV@LUseVerbatim

```
\def\FV@LUseVerbatim#1{%
  \FV@LVerbatimBegin#1\FV@LVerbatimEnd
  \@doendpe\global\@ignorefalse\ignorespaces}
```

\LVerbatimInput

```
\def\LVerbatimInput{\FV@Command{}{LVerbatimInput}}
```

\FVC@LVerbatimInput

```
\def\FVC@LVerbatimInput#1{\FV@LUseVerbatim{\FV@Input{#1}}}
```

Frames:

\FV@Frame@none

```
\def\FV@Frame@none{%
  \let\FV@BeginListFrame\relax
  \let\FV@LeftListFrame\relax
  \let\FV@RightListFrame\relax
  \let\FV@EndListFrame\relax}
```

\FV@Frame@single

```
\def\FV@Frame@single{%
  \let\FV@BeginListFrame\FV@BeginListFrame@Single
  \let\FV@LeftListFrame\FV@LeftListFrame@Single
  \let\FV@RightListFrame\FV@RightListFrame@Single
  \let\FV@EndListFrame\FV@EndListFrame@Single}
```

\FV@Frame@lines

```
\def\FV@Frame@lines{%
  \let\FV@BeginListFrame\FV@BeginListFrame@Lines
  \let\FV@LeftListFrame\relax
  \let\FV@RightListFrame\relax
  \let\FV@EndListFrame\FV@EndListFrame@Lines}
```

\FV@Frame@topline

```
\def\FV@Frame@topline{%
  \let\FV@BeginListFrame\FV@BeginListFrame@Lines
  \let\FV@LeftListFrame\relax
  \let\FV@RightListFrame\relax
  \let\FV@EndListFrame\relax}
```

\FV@Frame@bottomline

```
\def\FV@Frame@bottomline{%
  \let\FV@BeginListFrame\relax
  \let\FV@LeftListFrame\relax
  \let\FV@RightListFrame\relax
  \let\FV@EndListFrame\FV@EndListFrame@Lines}
```

\FV@Frame@leftline

```
%% To define a frame with only a left line
\def\FV@Frame@leftline{%
  % To define the \FV@FrameFillLine macro (from \FV@BeginListFrame)
  \ifx\FancyVerbFillColor\relax
```

```
    \let\FV@FrameFillLine\relax
  \else
    \@tempdima\FV@FrameRule\relax
    \multiply\@tempdima-\tw@
    \edef\FV@FrameFillLine{%
      {\noexpand\FancyVerbFillColor{\vrule\@width\number\@tempdima sp}%
      \kern-\number\@tempdima sp}}%
  \fi
  \let\FV@BeginListFrame\relax
  \let\FV@LeftListFrame\FV@LeftListFrame@Single
  \let\FV@RightListFrame\relax
  \let\FV@EndListFrame\relax}
```

\FV@BeginListFrame@Single

```
\def\FV@BeginListFrame@Single{%
  \lineskip\z@
  \baselineskip\z@
  \ifx\FancyVerbFillColor\relax
    \let\FV@FrameFillLine\relax
  \else
    \@tempdima\FV@FrameRule\relax
    \multiply\@tempdima-\tw@
    \edef\FV@FrameFillLine{%
      {\noexpand\FancyVerbFillColor{\vrule\@width\number\@tempdima sp}%
      \kern-\number\@tempdima sp}}%
  \fi
%% DG/SR modification begin - May. 19, 1998
%%  \FV@SingleFrameLine
  \FV@SingleFrameLine{\z@}%
%% DG/SR modification end
  \penalty\@M
  \FV@SingleFrameSep
  \penalty\@M}
```

\FV@Label

```
%% DG/SR modification begin - May. 19, 1998
\define@key{FV}{label}{%
  \def\@tempa{#1}%
  \ifx\@tempa\FV@None
    \let\FV@LabelBegin\relax
    \let\FV@LabelEnd\relax
  \else
    \FV@Label@i#1\@nil%
  \fi}
```

\FV@Label@i

```
\def\FV@Label@i{\@ifnextchar[{\FV@Label@ii}{\FV@Label@ii[]}}
```

\FV@Label@ii

```
\def\FV@Label@ii[#1]#2\@nil{%
  \def\@tempa{#1}%
  \ifx\@tempa\empty
    \def\FV@LabelBegin{#2}%
  \else
    \def\FV@LabelBegin{#1}%
    \def\FV@LabelPositionBottomLine{\@ne}%
  \fi
  \def\FV@LabelEnd{#2}}
```

```
\fvset{label=none}
```

\FV@LabelPosition

```
\define@key{FV}{labelposition}[none]{%
  \@ifundefined{FV@LabelPosition@#1}%
    {\FV@Error{Label position ‘#1’ not defined.}\FV@eha}%
    {\@nameuse{FV@LabelPosition@#1}}}
```

\FV@LabelPosition@none

```
\def\FV@LabelPosition@none{%
  \let\FV@LabelPositionTopLine\relax%
  \let\FV@LabelPositionBottomLine\relax}
```

\FV@LabelPosition@topline

```
\def\FV@LabelPosition@topline{%
  \def\FV@LabelPositionTopLine{\@ne}%
  \let\FV@LabelPositionBottomLine\relax}
```

\FV@LabelPosition@bottomline

```
\def\FV@LabelPosition@bottomline{%
  \let\FV@LabelPositionTopLine\relax%
  \def\FV@LabelPositionBottomLine{\@ne}}
```

\FV@LabelPosition@all

```
\def\FV@LabelPosition@all{%
  \def\FV@LabelPositionTopLine{\@ne}%
  \def\FV@LabelPositionBottomLine{\@ne}}
```

```
\fvset{labelposition=topline}
%% DG/SR modification end
```

\FV@SingleFrameLine

```
%% DG/SR modification begin - May. 19, 1998
%% \def\FV@SingleFrameLine{%
\def\FV@SingleFrameLine#1{%
%% DG/SR modification end
  \hbox to\z@{%
    \kern\leftmargin
%% DG/SR modification begin - Jun. 22, 1998
    \ifnum#1=\z@
      \let\FV@Label\FV@LabelBegin
    \else
      \let\FV@Label\FV@LabelEnd
    \fi
    \ifx\FV@Label\relax
%% DG/SR modification end
      \FancyVerbRuleColor{\vrule \@width\linewidth \@height\FV@FrameRule}%
%% DG/SR modification begin - Jun. 22, 1998
    \else
      \ifnum#1=\z@
        \setbox\z@\hbox{\strut\enspace\FV@LabelBegin\enspace\strut}%
      \else
        \setbox\z@\hbox{\strut\enspace\FV@LabelEnd\enspace\strut}%
      \fi
      \@tempdimb=\dp\z@
      \advance\@tempdimb -.5\ht\z@
      \@tempdimc=\linewidth
      \advance\@tempdimc -\wd\z@
      \divide\@tempdimc\tw@
      \ifnum#1=\z@              % Top line
        \ifx\FV@LabelPositionTopLine\relax
          \FancyVerbRuleColor{\vrule \@width\linewidth \@height\FV@FrameRule}%
        \else
          \FV@FrameLineWithLabel
        \fi
      \else                     % Bottom line
        \ifx\FV@LabelPositionBottomLine\relax
          \FancyVerbRuleColor{\vrule \@width\linewidth \@height\FV@FrameRule}%
        \else
          \FV@FrameLineWithLabel
        \fi
      \fi
    \fi
%% DG/SR modification end
    \hss}}
```

\FV@FrameLineWithLabel

```
%% DG/SR modification begin - May. 19, 1998
\def\FV@FrameLineWithLabel{%
 \ht\z@\@tempdimb\dp\z@\@tempdimb%
 \FancyVerbRuleColor{%
   \vrule \@width\@tempdimc \@height\FV@FrameRule
   \raise\@tempdimb\box\z@
   \vrule \@width\@tempdimc \@height\FV@FrameRule}}
%% DG/SR modification end
```

\FV@BeginListFrame@Lines

```
\def\FV@BeginListFrame@Lines{%
  \begingroup
    \lineskip\z@skip
%% DG modification begin - June 18, 1997 (effect of \baselineskip too earlier)
%%    \baselineskip\z@skip
%%    \FV@SingleFrameLine
%% DG/SR modification begin - May. 19, 1998
%%    \FV@SingleFrameLine
    \FV@SingleFrameLine{\z@}%
%% DG/SR modification end
    \kern-0.5\baselineskip\relax
    \baselineskip\z@skip
%% DG modification end
    \kern\FV@FrameSep\relax
  \endgroup}%
```

\FV@EndListFrame@Lines

```
\def\FV@EndListFrame@Lines{%
  \begingroup
    \baselineskip\z@skip
    \kern\FV@FrameSep\relax
%% DG/SR modification begin - May. 19, 1998
%%    \FV@SingleFrameLine
    \FV@SingleFrameLine{\@ne}%
%% DG/SR modification end
  \endgroup}
```

\FV@SingleFrameSep

```
\def\FV@SingleFrameSep{%
  \hbox to \z@{%
    \kern\leftmargin
    \hbox to\linewidth{%
      \FancyVerbRuleColor{%
%% DG modification begin - June 18, 1997 (\FV@FrameSep missing)
        \ifx\FancyVerbFillColor\relax
          \vrule\@width 0pt\@height\FV@FrameSep\relax
        \fi
%% DG modification end
        \vrule\@width\FV@FrameRule\relax
        \ifx\FancyVerbFillColor\relax
          \hfil
        \else
          {\FancyVerbFillColor\leaders\hrule\@height\FV@FrameSep\hfil}%
        \fi
%% DG modification begin - June 18, 1997 (\FV@FrameSep missing)
        \ifx\FancyVerbFillColor\relax
          \vrule\@width 0pt\@height\FV@FrameSep\relax
        \fi
%% DG modification end
```

```
        \vrule\@width\FV@FrameRule\relax}}%
    \hss}}
```

\FV@LeftListFrame@Single

```
\def\FV@LeftListFrame@Single{%
  \strut
  {\FancyVerbRuleColor{\vrule \@width\FV@FrameRule}}%
  \FV@FrameFillLine
%% DG modification begin - June 18, 1997 (to fill color on left side)
%%  \kern\FV@FrameSep}
  \ifx\FancyVerbFillColor\relax
    \kern\FV@FrameSep
  \else
    {\noexpand\leavevmode\FancyVerbFillColor{\vrule\@width\FV@FrameSep}}%
  \fi}
%% DG modification end
```

\FV@RightListFrame@Single

```
\def\FV@RightListFrame@Single{%
%% DG modification begin - June 18, 1997 (to fill color on right side)
%%  \kern\FV@FrameSep
  \ifx\FancyVerbFillColor\relax
    \kern\FV@FrameSep
  \else
    {\noexpand\leavevmode\FancyVerbFillColor{\vrule\@width\FV@FrameSep}}%
  \fi
  {\noexpand\leavevmode\FancyVerbRuleColor{\vrule\@width\FV@FrameRule}}}
%% DG modification end
```

\FV@EndListFrame@Single

```
\def\FV@EndListFrame@Single{%
  \penalty\@M
  \FV@SingleFrameSep
  \penalty\@M
%% DG/SR modification begin - May. 19, 1998
%%  \FV@SingleFrameLine}
  \FV@SingleFrameLine{\@ne}}
%% DG/SR modification end
```

\FV@FrameRule

```
\define@key{FV}{framerule}{%
  \@tempdima=#1\relax
  \edef\FV@FrameRule{\number\@tempdima sp\relax}}
```

\KV@FV@framerule@default

```
\def\KV@FV@framerule@default{\let\FV@FrameRule\fboxrule}
```

\FV@FrameSep

```
\define@key{FV}{framesep}{%
  \@tempdima=#1\relax
  \edef\FV@FrameSep{\number\@tempdima sp\relax}}
```

\KV@FV@framesep@default

```
\def\KV@FV@framesep@default{\let\FV@FrameSep\fboxsep}
```

```
\fvset{framerule,framesep}
```

\FancyVerbRuleColor

```
\define@key{FV}{rulecolor}{%
  \def\@tempa{#1}%
  \ifx\@tempa\FV@None
    \let\FancyVerbRuleColor\relax
  \else
    \let\FancyVerbRuleColor\@tempa
  \fi}
```

\FancyVerbFillColor

```
\define@key{FV}{fillcolor}{%
  \def\@tempa{#1}%
  \ifx\@tempa\FV@None
    \let\FancyVerbFillColor\relax
  \else
    \let\FancyVerbFillColor\@tempa
  \fi}
```

```
\fvset{rulecolor=none,fillcolor=none}
```

\FV@Frame@double

```
\def\FV@Frame@double{%
  \let\FV@FrameBegin\FV@FrameBegin@double
  \let\FV@FrameLine\FV@FrameLine@double
  \let\FV@FrameEnd\FV@FrameEnd@double}
```

```
\define@key{FV}{frame}[none]{%
  \@ifundefined{FV@Frame@#1}%
    {\FV@Error{Frame style ‘#1’ not defined.}\FV@eha}%
    {\@nameuse{FV@Frame@#1}}}
```

```
\fvset{frame=none}
```

Code line numbers:

```
\newcounter{FancyVerbLine}
```

\FV@SetLineNo

```
\define@key{FV}{firstnumber}[auto]{%
  \def\@tempa{#1}\def\@tempb{auto}%
  \ifx\@tempa\@tempb
```

```
    \def\FV@SetLineNo{%
      \c@FancyVerbLine\FV@CodeLineNo
      \advance\c@FancyVerbLine\m@ne}%
  \else
    \def\@tempb{last}%
    \ifx\@tempa\@tempb
      \let\FV@SetLineNo\relax
    \else
%% DG/SR modification begin - Jan. 19, 1998
%%      \def\FV@SetLineNo{\c@FancyVerbLine#1}%
      \def\FV@SetLineNo{%
        \c@FancyVerbLine#1
        \advance\c@FancyVerbLine\m@ne}%
%% DG/SR modification end
    \fi
  \fi}
```

\if@FV@NumberBlankLines

```
\define@booleankey{FV}{numberblanklines}%
  {\let\if@FV@NumberBlankLines\iftrue}
  {\let\if@FV@NumberBlankLines\iffalse}
\fvset{numberblanklines=true}
```

\FV@refstepcounter

```
%% DG/SR modification begin - May. 20, 1998
%%\def\refstepcounter#1{% Adapted from latex.ltx
\def\FV@refstepcounter#1{%
%% DG/SR modification end
  \stepcounter{#1}%
  \protected@edef\@currentlabel
    {\csname p@#1\endcsname\arabic{FancyVerbLine}}}
```

\FV@StepLineNo

```
\def\FV@StepLineNo{%
  \FV@SetLineNo%
  \def\FV@StepLineNo{%
    \if@FV@NumberBlankLines%
      \FV@refstepcounter{FancyVerbLine}%
    \else%
      \ifx\FV@Line\empty%
      \else%
        \FV@refstepcounter{FancyVerbLine}%
      \fi%
    \fi}%
  \FV@StepLineNo}
```

\theFancyVerbLine

```
%% DG/SR modification begin - 1995
%%\def\theFancyVerbLine{\rm\tiny\arabic{FancyVerbLine}}
\def\theFancyVerbLine{\rmfamily\tiny\arabic{FancyVerbLine}}
%% DG/SR modification end
```

```
\define@key{FV}{numbers}[none]{%
  \@ifundefined{FV@Numbers@#1}%
    {\FV@Error{Numbers style ‘#1’ not defined.}\FV@eha}%
    {\@nameuse{FV@Numbers@#1}}}
```

(D.G. – Dec. 20, 1995 and Jan. 28, 1998):
Add-on to allow a step when printing the lines counter (“`stepnumber`” keyword)
Add-on to allow the counter to be printed on right side (`numbers=right`)

\FV@Numbers@none

```
%% DG modification begin - Dec. 20, 1995 and Jan. 28, 1998
%%\def\FV@Numbers@none{\let\FV@LeftListNumber\relax}
\def\FV@Numbers@none{%
\let\FV@LeftListNumber\relax
\let\FV@RightListNumber\relax}
```

```
\newcount\FV@StepNumber
\define@key{FV}{stepnumber}{\FV@StepNumber#1}
```

\KV@FV@stepnumber@default

```
\def\KV@FV@stepnumber@default{\FV@StepNumber\@ne}
```

```
\fvset{stepnumber}
```

\FV@Numbers@left

```
%% DG modification begin - Dec. 20, 1995
%%\def\FV@Numbers@left{%
%%  \def\FV@LeftListNumber{\hbox to\z@{%
%%    \hss\theFancyVerbLine\kern\FV@NumberSep}}}
\def\FV@Numbers@left{%
%% DG/SR modification begin - Apr. 28, 1998
  \let\FV@RightListNumber\relax
%% DG/SR modification end
  \def\FV@LeftListNumber{%
  \@tempcnta=\FV@CodeLineNo
  \@tempcntb=\FV@CodeLineNo
  \divide\@tempcntb\FV@StepNumber
  \multiply\@tempcntb\FV@StepNumber
  \ifnum\@tempcnta=\@tempcntb
%% DG/SR modification begin - Apr. 28, 1998
%%    \hbox to\z@{\hss\theFancyVerbLine\kern\FV@NumberSep}%
      \if@FV@NumberBlankLines
        \hbox to\z@{\hss\theFancyVerbLine\kern\FV@NumberSep}%
      \else
        \ifx\FV@Line\empty
        \else
          \hbox to\z@{\hss\theFancyVerbLine\kern\FV@NumberSep}%
        \fi
      \fi
%% DG/SR modification end
  \fi}}
```

\FV@Numbers@right

```
\def\FV@Numbers@right{%
%% DG/SR modification begin - Apr. 28, 1998
  \let\FV@LeftListNumber\relax
%% DG/SR modification end
  \def\FV@RightListNumber{%
  \@tempcnta=\FV@CodeLineNo
  \@tempcntb=\FV@CodeLineNo
  \divide\@tempcntb\FV@StepNumber
  \multiply\@tempcntb\FV@StepNumber
  \ifnum\@tempcnta=\@tempcntb
%% DG/SR modification begin - Apr. 28, 1998
%%    \hbox to \z@{\kern\FV@NumberSep\theFancyVerbLine\hss}%
    \if@FV@NumberBlankLines
      \hbox to\z@{\kern\FV@NumberSep\theFancyVerbLine\hss}%
    \else
      \ifx\FV@Line\empty
      \else
        \hbox to\z@{\kern\FV@NumberSep\theFancyVerbLine\hss}%
      \fi
    \fi
%% DG/SR modification end
  \fi}}
%% DG modification end
```

\FV@NumberSep

```
\define@key{FV}{numbersep}{%
  \@tempdima=#1\relax
  \edef\FV@NumberSep{\number\@tempdima sp\relax}}
```

```
\fvset{numbers=none,numbersep=12pt,firstnumber=auto}
```

12.14 BVerbatim

BVerbatim

```
\def\FV@BVerbatimBegin{%
  \begingroup
    \FV@UseKeyValues
    \FV@BeginVBox
    \let\FV@ProcessLine\FV@BProcessLine
    \FV@FormattingPrep
    \FV@ObeyTabsInit}%
\def\FV@BVerbatimEnd{\FV@EndVBox\endgroup}
```

\FV@BeginVBox

```
\def\FV@BeginVBox{%
  \leavevmode
  \hbox\ifx\FV@boxwidth\relax\else to\FV@boxwidth\fi\bgroup
  \ifcase\FV@baseline\vbox\or\vtop\or$\vcenter\fi\bgroup}
```

\FV@EndVBox

```
\def\FV@EndVBox{\egroup\ifmmode$\fi\hfil\egroup}
```

\FV@boxwidth

```
\define@key{FV}{boxwidth}{%
  \def\@tempa{#1}\def\@tempb{auto}%
  \ifx\@tempa\@tempb
    \let\FV@boxwidth\relax
  \else
    \@tempdima=#1\relax
    \edef\FV@boxwidth{\number\@tempdima sp}%
  \fi}
```

\KV@FV@boxwidth@default

```
\def\KV@FV@boxwidth@default{\let\FV@boxwidth\relax}
```

\FV@baseline

```
\define@key{FV}{baseline}{%
  \if t#1\@empty\let\FV@baseline\@ne\else
    \if c#1\@empty\let\FV@baseline\tw@\else\let\FV@baseline\z@\fi
  \fi}
```

```
\fvset{baseline=b,boxwidth}
```

\FV@BProcessLine

```
\def\FV@BProcessLine#1{\hbox{\FancyVerbFormatLine{#1}}}
```

\FVB@BVerbatim

```
\def\FVB@BVerbatim{\FV@BVerbatimBegin\FV@Scan}
```

\FVE@BVerbatim

```
\def\FVE@BVerbatim{\FV@BVerbatimEnd}
```

```
\DefineVerbatimEnvironment{BVerbatim}{BVerbatim}{}
```

\FV@BUseVerbatim

```
\def\FV@BUseVerbatim#1{\FV@BVerbatimBegin#1\FV@BVerbatimEnd}
```

\BVerbatimInput

```
\def\BVerbatimInput{\FV@Command{}{BVerbatimInput}}
```

\FVC@BVerbatimInput

```
\def\FVC@BVerbatimInput#1{\FV@BUseVerbatim{\FV@Input{#1}}}
```

12.15 Save verbatim

SaveVerbatim

```
\def\SaveVerbatim{\FV@Environment{}{SaveVerbatim}}
```

\FVB@SaveVerbatim

```
\def\FVB@SaveVerbatim#1{%
  \@bsphack
  \begingroup
    \FV@UseKeyValues
%%     \FV@BeginVBox
%%     \let\FV@ProcessLine\FV@BProcessLine
%%     \FV@FormattingPrep
%%     \FV@ObeyTabsInit%
%%
    \def\SaveVerbatim@Name{#1}%
    \gdef\FV@TheVerbatim{}%
    \def\FV@ProcessLine##1{%
      \expandafter\gdef\expandafter\FV@TheVerbatim\expandafter{%
        \FV@TheVerbatim\FV@ProcessLine{##1}}}%
    \gdef\FV@TheVerbatim{}%
    \FV@Scan}
```

\FVE@SaveVerbatim

```
\def\FVE@SaveVerbatim{%
  \expandafter\global\expandafter\let
  \csname FV@SV@\SaveVerbatim@Name\endcsname\FV@TheVerbatim
%%  \expandafter\gdef
%%     \csname FV@SV@\SaveVerbatim@Name\endcsname{\FV@TheVerbatim}
%%     \FV@EndVBox
%%  \endgroup}
  \endgroup\@esphack}
```

SaveVerbatim

```
\DefineVerbatimEnvironment{SaveVerbatim}{SaveVerbatim}{}
```

\FV@CheckIfSaved

```
\def\FV@CheckIfSaved#1#2{%
  \@ifundefined{FV@SV@#1}%
  {\FV@Error{No verbatim text has been saved under name ‘#1’}\FV@eha}%
  {#2{\csname FV@SV@#1\endcsname}}}
```

\UseVerbatim

```
\def\UseVerbatim{\FV@Command{}{UseVerbatim}}
```

\FVC@UseVerbatim

```
\def\FVC@UseVerbatim#1{\FV@CheckIfSaved{#1}{\FV@UseVerbatim}}
```

\LUseVerbatim

```
\def\LUseVerbatim{\FV@Command{}{LUseVerbatim}}
```

\FVC@LUseVerbatim

```
\def\FVC@LUseVerbatim#1{\FV@CheckIfSaved{#1}{\FV@LUseVerbatim}}
```

\BUseVerbatim

```
\def\BUseVerbatim{\FV@Command{}{BUseVerbatim}}
```

\FVC@BUseVerbatim

```
\def\FVC@BUseVerbatim#1{\FV@CheckIfSaved{#1}{\FV@BUseVerbatim}}
```

12.16 Verbatim out

```
\newwrite\FV@OutFile
```

\VerbatimOut

```
\def\VerbatimOut{\FV@Environment{}{VerbatimOut}}
```

\FVB@VerbatimOut

```
\def\FVB@VerbatimOut#1{%
  \@bsphack
  \begingroup
    \FV@UseKeyValues
    \FV@DefineWhiteSpace
    \def\FV@Space{\space}%
    \FV@DefineTabOut
    \def\FV@ProcessLine{\immediate\write\FV@OutFile}%
    \immediate\openout\FV@OutFile #1\relax
    \let\FV@FontScanPrep\relax
%% DG/SR modification begin - May. 18, 1998 (to avoid problems with ligatures)
    \let\@noligs\relax
%% DG/SR modification end
    \FV@Scan}
```

\FVE@VerbatimOut

```
\def\FVE@VerbatimOut{\immediate\closeout\FV@OutFile\endgroup\@esphack}
```

VerbatimOut

```
\DefineVerbatimEnvironment{VerbatimOut}{VerbatimOut}{}
```

\FV@DefineTabOut

```
\def\FV@DefineTabOut{%
  \def\FV@Tab{}%
  \@tempcnta=\FancyVerbTabSize\relax
  \loop\ifnum\@tempcnta>\z@
    \edef\FV@Tab{\FV@Tab\space}%
    \advance\@tempcnta\m@ne
  \repeat}
```

12.17 Short verbatim

\SaveVerb

Note "\outer\def^^M{}". This is so that verbatim commands report an error when encountering an end-of-line, rather than scanning to the end of the file each time there is a missing verbatim delimiter.

If scanning fails (and thus TeX ignores \FV@GetVerb), #1 is defined to be empty, a group is ended, but \FancyVerbAfterSave is not invoked.

\FV@Command

```
\def\SaveVerb{\FV@Command{}{SaveVerb}}
```

\FVC@SaveVerb

```
\begingroup
\catcode`\^^M=\active%
\gdef\FVC@SaveVerb#1#2{%
  \@namedef{FV@SV@#1}{}%
  \begingroup%
    \FV@UseKeyValues%
    \FV@CatCodes%
    \outer\def^^M{\FV@EOL}%
    \global\let\@tempg\FancyVerbAfterSave%
    \catcode`#2=12%
    \def\@tempa{\def\FancyVerbGetVerb####1####2}%
    \expandafter\@tempa\string#2{\endgroup\@namedef{FV@SV@#1}{##2}\@tempg}%
    \FancyVerbGetVerb\FV@EOL}%
\endgroup
```

\FV@EOL

```
\def\FV@EOL{%
  \endgroup
  \FV@Error%
    {Could not find the end delimiter of a short verb command}%
    {You probably just forget the end delimiter of a \string\Verb\space or
      \string\SaveVerb^^J%
      command, or you broke the literal text across input lines.^^J%
      Hit <return> to procede.}}
```

\FancyVerbAfterSave

```
\define@key{FV}{aftersave}{\def\FancyVerbAfterSave{#1}}
\fvset{aftersave=}
```

\FV@UseVerb

```
\def\FV@UseVerb#1{\mbox{\FV@UseKeyValues\FV@FormattingPrep#1}}
```

\UseVerb

```
\def\UseVerb{\FV@Command{}{UseVerb}}
```

\FVC@UseVerb

```
\def\FVC@UseVerb#1{%
  \@ifundefined{FV@SV@#1}%
    {\FV@Error{Short verbatim text never saved to name ‘#1’}\FV@eha}%
    {\FV@UseVerb{\@nameuse{FV@SV@#1}}}}
```

\Verb

```
\def\Verb{\FV@Command{}{Verb}}
```

\FVC@Verb

```
\begingroup
\catcode‘\^^M=\active%
\gdef\FVC@Verb#1{%
  \begingroup%
    \FV@UseKeyValues%
    \FV@FormattingPrep%
    \FV@CatCodes%
    \outer\def^^M{}%
    \catcode‘#1=12%
    \def\@tempa{\def\FancyVerbGetVerb####1####2}%
    \expandafter\@tempa\string#1{\mbox{##2}\endgroup}%
    \FancyVerbGetVerb\FV@EOL}%
\endgroup
```

\DefineShortVerb

```
\def\DefineShortVerb{\FV@Command{}{DefineShortVerb}}
```

\FVC@DefineShortVerb

```
\def\FVC@DefineShortVerb#1{%
  \@ifundefined{FV@CC@\string#1}%
    {\FVC@@DefineShortVerb#1}%
    {\FV@Error{‘\expandafter\@gobble\string#1’ is already a short
      verb character.}\FV@eha}}
```

\FVC@@DefineShortVerb

```
\def\FVC@@DefineShortVerb#1{%
  \begingroup
    \lccode‘\~=‘#1%
    \lowercase{\gdef\@tempg{\edef~}\global\let\@temph~}%
  \endgroup
  \expandafter\let\csname FV@AC@\string#1\endcsname\@temph
  \expandafter\edef\csname FV@CC@\string#1\endcsname{\the\catcode`#1}%
  \expandafter\let\csname FV@KV@\string#1\endcsname\FV@KeyValues
  \@tempg{%
    \let\noexpand\FV@KeyValues\expandafter\noexpand
      \csname FV@KV@\string#1\endcsname
    \noexpand\FVC@Verb\expandafter\@gobble\string#1}%
  \expandafter\def\expandafter\dospecials\expandafter{\dospecials\do#1}%
  \expandafter\def\expandafter\@sanitize\expandafter{\@sanitize\@makeother#1}%
  \catcode`#1=\active}%
```

\UndefineShortVerb

```
\def\UndefineShortVerb#1{%
  \@ifundefined{FV@CC@\string#1}%
    {\FV@Error{`\expandafter\@gobble\string#1' is not a short
      verb character}\FV@eha}%
    {\FV@UndefineShortVerb#1}}
```

\FV@UndefineShortVerb

```
\def\FV@UndefineShortVerb#1{%
  \catcode`#1=\csname FV@CC@\string#1\endcsname
%% DG/SR modification begin - Jun. 12, 1998
  \expandafter\let\csname FV@CC@\string#1\endcsname\relax
%% DG/SR modification end
  \begingroup
    \lccode`\~=`#1%
    \lowercase{\gdef\@tempg{\let~}}%
  \endgroup
  \expandafter\@tempg\csname FV@AC@\string#1\endcsname
  \def\@tempa##1\do#1##2\@nil##3\@nil##4\@@nil{##3\def\dospecials{##1##2}\fi}%
  \expandafter\@tempa\dospecials\@nil\iftrue\@nil\do#1\@nil\iffalse\@nil\@@nil
  \def\@tempa##1\@makeother#1##2\@nil##3\@nil##4\@@nil{%
    ##3\def\@sanitize{##1##2}\fi}%
  \expandafter\@tempa\@sanitize\@nil\iftrue\@nil\do#1\@nil\iffalse\@nil\@@nil}
```

Moving verbatim. Need to worry about using separate identifier for this class of verbatim, and

\SaveMVerb

```
\def\SaveMVerb{\FV@Command{}{SaveMVerb}}
```

\FVC@SaveMVerb

```
\begingroup
\catcode`\^^M=\active%
\gdef\FVC@SaveMVerb#1#2{%
  \@ifundefined{FV@SVM@#1}{}%
    {\FV@Error{Moving verbatim name `#1' already used}%
```

```
      {I will overwrite the old definition. Hit <return> to continue.}}%
  \global\@namedef{FV@SVM@#1}{}%
  \begingroup%
    \let\FV@SavedKeyValues\FV@KeyValues%
    \FV@UseKeyValues%
    \FV@CatCodes%
    \outer\def^^M{}%
    \global\let\@tempg\FancyVerbAfterSave%
    \catcode`#2=12%
    \def\@tempa{\def\FancyVerbGetVerb####1####2}%
    \expandafter\@tempa\string#2{%
      \if@filesw
        \FV@DefineWhiteSpace%
        \let\FV@Space\space%
        \let\FV@Tab\space%
        \FV@MakeUnActive%
        \let\protect\string
        \immediate\write\@auxout{%
          \noexpand\SaveGVerb[\FV@SavedKeyValues]{#1}\string#2##2\string#2}%
      \fi
      \endgroup%
      \@namedef{FV@SV@#1}{##2}%
      \@tempg}%
    \FancyVerbGetVerb\FV@EOL}%
\endgroup
```

\SaveGVerb

```
\def\SaveGVerb{\FV@Command{}{SaveGVerb}}
```

\FVC@SaveGVerb

```
\begingroup
\catcode`\^^M=\active%
\gdef\FVC@SaveGVerb#1#2{%
  \global\@namedef{FV@SVG@#1}{}%
  \begingroup%
    \FV@UseKeyValues%
    \FV@CatCodes%
    \outer\def^^M{}%
    \catcode`#2=12%
    \def\@tempa{\def\FancyVerbGetVerb####1####2}%
    \expandafter\@tempa\string#2{\endgroup\global\@namedef{FV@SVG@#1}{##2}}%
    \FancyVerbGetVerb\FV@EOL}%
\endgroup
```

\UseMVerb

```
\def\UseMVerb{\protect\pUseMVerb}
```

\pUseMVerb

```
\def\pUseMVerb{\FV@Command{}{pUseMVerb}}
```

\FVC@pUseMVerb

```
1367 \def\FVC@pUseMVerb#1{%
1368   \expandafter\ifx \csname FV@SVM@#1\endcsname\relax
1369     \expandafter\ifx \csname FV@SVG@#1\endcsname\relax
1370       \@warning{Moving verbatim text not defined for name ‘#1’}\FV@eha
1371       {\bf ??}%
1372     \else
1373       \FV@UseVerb{\@nameuse{FV@SVG@#1}}%
1374     \fi
1375   \else
1376     \FV@UseVerb{\@nameuse{FV@SVM@#1}}%
1377   \fi}
```

```
1378 \expandafter\ifx\csname documentclass\endcsname\relax
```

lrbox

```
1379   \def\lrbox#1{%
1380     \edef\@tempa{%
1381       \endgroup
1382       \setbox#1\hbox{%
1383         \begingroup\aftergroup}%
1384           \def\noexpand\@currenvir{\@currenvir}}%
1385           %\def\noexpand\@currenvline{\on@line}}%
1386     \@tempa
1387       \@endpefalse
1388       \bgroup
1389         \ignorespaces}
1390   \def\endlrbox{\unskip\egroup}
```

```
1391 \fi
```

```
1392 %% DG/SR modification begin -  Mar 21 2000
1393 %%\@input{fancyvrb.rc}
1394 \InputIfFileExists{fancyvrb.cfg}{}{}
1395 %% DG/SR modification end
```

Unused code - don’t take care! (DG/SR)
Error messages to void:
</fancyvrb>

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Change History

v2.0 Beta
General: First version personally shown by Timothy van Zandt. . . . 1

v2.5

General: First public release. 1

v2.6

General: . Three bug corrected, options numberblanklines, label, labelposition and leftline added and few precisions. 1

Add “leftline” value for the “frame” parameter (idea taken from a requirement of Christophe Roudet on June 9, 1998). 51

Add the “label” and “labelposition” parameters (suggested by Michael Friendly on May 15, 1998). 52

Add the “numberblanklines” parameter to allow to don’t number empty lines (suggested by Philippe Esperet on April 25, 1998). 58

Correct a bug for the – ligature in T1 encoding (reported by Mario Hassler on May 12, 1998). 39

Correct a bug to undefine the macro associated to short verbatim character (reported by Mikhail Kolodin on June 12, 1998). 66

Correct a bug which didn’t allow to use the label macro in verbatim env. (reported by Philippe Esperet on May 20, 1998). 58

v2.7

General: DG/SR changed how fancyvrb.cfg included 1

v2.7a

General: firstline bug 1

v2.8

General: fixed bug with several trailing spaces (hv) 1