# Documentation for fancybox.sty: Box tips and tricks for LaTeX

Timothy Van Zandt
tvz@Princeton.EDU

Version 1.2
February 27, 191998

`fancybox.sty`, together with its documentation, gives extensive answers to and solutions for many questions about how to frame or rotate this or that in LaTeX. It also contains commands for shadow, double and oval frames.

## Contents

# 1 Fancy frames

`fancybox.sty` has five variants of LaTeX's `\fbox` command:

> `\shadowbox`, `\doublebox`, `\ovalbox` (with `\thinlines`) and `\Ovalbox` (with `\thicklines`).

Here are examples:[1]

```
\shadowbox{\large\bf New Glarus Birdwatch}
```

**New Glarus Birdwatch**

```
\doublebox{\large\bf New Glarus Birdwatch}
```

**New Glarus Birdwatch**

```
\ovalbox{\large\bf New Glarus Birdwatch}
```

**New Glarus Birdwatch**

```
\Ovalbox{\large\bf New Glarus Birdwatch}
```

**New Glarus Birdwatch**

The distance between the box and the frame is `\fboxsep`, as with LaTeX's `\fbox` command. The commands use other parameters as well:

`\shadowbox` The width of the frame is `\fboxrule` (the same as with `\fbox`). The width of the shadow is `\shadowsize` (default: `4pt`).

`\doublebox` The width of the inner frame is .75`\fboxrule`, and the width of the outer frame is 1.5`\fboxrule`. The distance between the two frames is 1.5`\fboxrule` plus .5pt.

[1] In this documentation, the default value of `\fboxsep` has been changed from 3pt to 6pt.

`\ovalbox` The width of the frame is set by the `\thinlines` declaration. The diameter of the corner arcs is set with the `\cornersize` command.

`\cornersize{`*num*`}`

sets the diameter of the corners arcs to *num* times the lessor of the width and height of the box.

`\cornersize*{`*dim*`}`

sets the diameter of the corner arcs to *dim*. This is all approximate, because LaTeX has a limited range of arc sizes to choose from. The default is

`\cornersize{.5}`

`\Ovalbox` This is like `\ovalbox`, except that the width of the lines is set by the `\thicklines` declaration.

There are no analogs to LaTeX's `\framebox` command, which has various optional arguments not supported by `\fbox`. You can get the exact same functionality by putting the argument of the above framing commands in a `\makebox`.

There is also a variant `\fancyoval` of LaTeX's `\oval` picture object. The difference is that `\oval` always makes the diameter of the corner arcs as large as possible, and `\fancyoval` uses the `\cornersize` command to set the diameter.

## 2 A short course on boxes

The `\shadowbox`, `\doublebox`, `\ovalbox` and `\Ovalbox` commands described in the previous section are examples of LR-box commands, meaning that their argument is processed in LR mode. LaTeX LR-box commands include `\mbox`, `\makebox`, `\fbox`, `\framebox`, `\sbox` and `\savebox`. All the PSTricks commands whose argument is text are LR-box commands, including, e.g, the framing, rotating, scaling and positioning commands, and some of the node commands. Any rotation command is an LR-box command.

The purpose of the rest of this documentation is to provide answers to, and solutions for, frequently asked questions about using LR-box commands with LaTeX. I will use `\fbox` for the leading example of a box framing command,[2] and `\rotateleft` for the leading example of a box rotation command. (`fancybox.sty` does not contain a `\rotateleft` command, as this must be implemented via `\special`'s, but there are numerous box-rotation style files around.) However, most of what is said here applies to any LR-box command.

In each LR-box command, the text is processed in restricted horizontal mode, which is referred to as "LR-mode" in Lamport's *LaTeX: User's Guide and Reference Manual.* In restricted horizontal mode, the input, consisting of regular characters and boxes, is made into one (long or short) line. There is no line-breaking, nor can there be vertical mode material such as an entire displayed equation. However, the fact that you can include another box means that this isn't really a restriction.

For one thing, alignment environments such as LaTeX's `tabular` are just boxes, and thus present no problem. Picture environments and the LR-box commands themselves are also just boxes. Entire paragraphs or other vertical mode material such as displayed equations can be nested in a `\parbox` or `minipage`.

# 3 Defining LR-box environments

To frame a `minipage`, you have to write

```
\fbox{%
  \begin{minipage}{3in}
    blah
  \end{minipage}}
```

You might want to define an environment `fminipage` that frames its contents, but you can't use

```
\newenvironment{fminipage}%
  {\fbox{\begin{minipage}}%
  {\end{minipage}}}
```

because the braces are not balanced in the definition.

[2] In the examples using `\fbox`, be aware that the default value of `\fboxsep` has been changed in this documentation from 3pt to 6pt.

`fancybox.sty` contains an `Sbox` environment that makes it easy to define your own LR-box environments. It is a variant of LaTeX's `\sbox` command that saves the contents of the environment in a storage bin that can be retrieved with the command `\TheSbox`.[3] For example, here is a framed minipage:

```
\begin{Sbox}
  \begin{minipage}{3in}
    blah
  \end{minipage}
\end{Sbox}
\fbox{\TheSbox}
```

and here is an `fminipage` environment that works:

```
\newenvironment{fminipage}%
  {\begin{Sbox}\begin{minipage}}%
  {\end{minipage}\end{Sbox}\fbox{\TheSbox}}
```

Let's see that it really works:

```
\begin{fminipage}{2in}
  Since the former doesn't use braces to delimit
  the contents of the box, $\ldots$
\end{fminipage}
```

Since the former doesn't use braces to delimit the contents of the box, . . .

[3]The difference between

```
\begin{Sbox}
  blah
\end{Sbox}
\TheSbox
```

and

```
\newsavebox{\mybox}
\sbox{\mybox}{blah}
\usebox{\mybox}
```

is that `Sbox` saves the contents globally, and `\TheSbox` erases the contents globally.

# 4 Math

In-line math, or pieces of a displayed equation (as opposed to a whole equation), are horizontal mode material, but most LR-box commands switch out of math mode when they occur in math mode. Thus, you have to explicitly switch back in to math mode when desired.[4] For example:

```
$x + y = \fbox{$\Omega$}$
```

$$x + y = \boxed{\Omega}$$

You also have to explicitly write

`\scriptstyle`, `\scriptscriptstyle` or `\displaystyle`

if you want one of these special math styles. For example, here I will frame an equation, but not the equation number:

```
\begin{equation}
  \fbox{$\displaystyle
    \int_{\Omega_0} \zeta(\omega) d\omega
    \geq \bar{r}$}
\end{equation}
```

$$\boxed{\displaystyle\int_{\Omega_0} \zeta(\omega) d\omega \geq \bar{r}} \tag{1}$$

Entire displayed equations or `eqnarray` environments work differently because they are vertical mode material. Thus, they have to go inside a `\parbox` or `minipage`. E.g.,

```
\newlength{\mylength}
\[
  \setlength{\fboxsep}{15pt}
  \setlength{\mylength}{\linewidth}
  \addtolength{\mylength}{-2\fboxsep}
```

[4]This is *not* true for the PSTricks LR-box commands.

```
    \addtolength{\mylength}{-2\fboxrule}
    \fbox{%
      \parbox{\mylength}{
        \setlength{\abovedisplayskip}{0pt}
        \setlength{\belowdisplayskip}{0pt}
        \begin{equation}
          x + y = z
        \end{equation}}}
  \]
```

$$\boxed{x + y = z \qquad (2)}$$

The outer `\[ \]` are just used to display the boxed equation, rather than actually switch into math mode. Note how I set the width of the `\parbox` so that the displayed box would exactly have width `\linewidth`.[5]

I also set the display skips to `0pt` and increased the size of `\fboxsep` so that I would have the same distance all around between the equation and the frame.

This is again a mouthful, and so I might instead define:[6]

```
  \newenvironment{FramedEqn}%
    {\setlength{\fboxsep}{15pt}
      \setlength{\mylength}{\linewidth}%
      \addtolength{\mylength}{-2\fboxsep}%
      \addtolength{\mylength}{-2\fboxrule}%
      \Sbox
      \minipage{\mylength}%
         \setlength{\abovedisplayskip}{0pt}%
         \setlength{\belowdisplayskip}{0pt}%
         \equation}%
    {\endequation\endminipage\endSbox
      \[\fbox{\TheSbox}\]}
```

[5] That is what `\mylength` is for. It is better to define a single scratch length that you reuse rather than creating a new one each time.

[6] The reason for using `\minipage` instead of `\begin{minipage}`, and so on, is that with AmS-LaTeX, `\begin` and `\end` cannot appear in the definition of a new equation environment.

`fancybox.sty` doesn't bother defining any such environments, because there are too many possible designs. But let's see if the one above works:

```
\begin{FramedEqn}
  \Rightarrow P\sim\xi(P_\gamma)- \frac{1}{3}
\end{FramedEqn}
```

$$\boxed{\Rightarrow P \sim \xi(P_\gamma) - \frac{1}{3} \qquad (3)}$$

`fancybox.sty` contains a `Beqnarray` environment, which is like the `eqnarray` environment but it is not vertical mode material. Instead, it produces a box just large enough to hold all the equations. For example:

```
\fbox{%
  \begin{Beqnarray*}
    x & = & y\\
    y & > & x \\
    \int_4^5 f(x)dx & = & \sum_{i\in F} x_i
  \end{Beqnarray*}}
```

$$\boxed{\begin{array}{rcl} x & = & y \\ y & > & x \\ \int_4^5 f(x)dx & = & \sum_{i\in F} x_i \end{array}}$$

The unstarred version produces standard equation numbers on the right (even with the `leqno` style option and AmS-LaTeX). It might not work with special equation numbering macros.

# 5 Floats

A common mistake is to put a whole `table`, `figure` or other float environment inside an LR-box command. Instead, you should put everything that is *inside* the environment (including the `\caption`, if you want that boxed too) inside a `minipage` of the desired width, and then put the `minipage` inside the LR-box command.

For example:

```
\begin{table}[h]
  \begin{center}
    \fbox{%
      \begin{minipage}{.8\textwidth}
        \begin{center}
          \begin{tabular}{rl}
            foo & bar
          \end{tabular}
        \end{center}
        \caption{A table of foo and bar.}
      \end{minipage}}
  \end{center}
\end{table}
```

| | |
|---|---|
| foo | bar |

Table 1: A table of foo and bar.

Note how I had to use `center` twice: once to center the framed box, and again to center the stuff inside the box.

That is a mouthful, and so I might define a `FramedTable` environment like the following, which sets the size of the `minipage` so that the framed box is exactly the width of the page (no need for the first `center` environment this time):

```
\newenvironment{FramedTable}%
  {\begin{table}[h]
    \begin{Sbox}%
    \setlength{\mylength}{\textwidth}%
    \addtolength{\mylength}{-2\fboxsep}%
    \addtolength{\mylength}{-2\fboxrule}%
```

```
    \begin{minipage}{\mylength}}%
  {\end{minipage}\end{Sbox}\fbox{\TheSbox}\end{table}}%
```

Now let's see if it works:

```
\begin{FramedTable}
  \begin{center}
    \begin{tabular}{rl}
      foo & bar
    \end{tabular}
  \end{center}
  \caption{A table of foo and bar.}
\end{FramedTable}
```

| | |
|---|---|
| foo | bar |

Table 2: A table of foo and bar.

The most common reason to want to rotate an entire float, caption and all, is to put it on a page by itself in landscape mode, centered both horizontally and vertically. Compared to the table framing we did above, we just have to replace `\fbox` by our box rotation command (e.g., `\rotateleft` or whatever), set the width of the minipage to `\textheight` (if you want to use the full size of the page), and use the float position specifier `[p]`. `fancybox.sty` contains an environment,

```
\begin{landfloat}{float}{rotation command}
  ...
\end{landfloat}
```

that automates this. It has two arguments: the name of the floating environment, and your rotation command. For example, if `\rotateleft{foo}` rotates `foo` by 90 degrees, and you want a landscape mode `table`, then try

```
\begin{landfloat}{table}{\rotateleft}
  ...
\end{landfloat}
```

If the whole document is in landscape mode, then `landfloat` gives you a portrait-mode float—good for a table that is too tall to fit in landscape mode.

If you don't add a caption to a float, it doesn't matter much what floating environment you use (e.g., `table`, `figure` or whatever). Thus, you can put anything in a landscape float. For example, suppose I have a very wide equation. Then I can write:

```
\begin{landfloat}{table}{\rotateleft}
  \begin{equation}
    ...
  \end{equation}
\end{landfloat}
```

`fancybox.sty` defines a generic caption, `\GenericCaption`, that doesn't affect the numbering of floats, doesn't make an entry in a list of floats, and doesn't add anything to the argument you give. I could have used this if I wanted to add a caption to the previous example.

# 6 Center, flushleft and flushright

There are two ways to box a `center`, `flushleft` or `flushright` environment. If you have long lines and you want TEX to do the line breaking, then put the environment inside a `minipage`.[7] If you have short lines and you want the frame to adjust itself to the size of the longest line, then use `fancybox.sty`'s

`Bcenter`, `Bflushleft` or `Bflushright`

environment instead.

These are basically just `tabular` environments with a single column, and so each line should end with `\\`, or `\\[`*dim*`]` to insert extra space, and the text on each line must be balanced. Also, each line is sitting in its own group, and so, e.g., if you want to change the font for the whole environment, you should do this before the environment rather than after. Like the `tabular` environment, the box that is produced has the baseline at the center, unless

[7] List and other such environments work best inside a `minipage` environment rather than a `\parbox`.

you include an optional argument `[t]` to align the box with the top line or `[b]` to align the box with the bottom line.

For example:

```
\setlength{\fboxsep}{10pt}%
\fbox{%
  \begin{Bcenter}
    Love of life\\
    and other short stories\\
    by Policarpa Salabarrieta
\end{Bcenter}}
```

Love of life
and other short stories
by Policarpa Salabarrieta

Compare this with:

```
\setlength{\fboxsep}{10pt}%
\fbox{%
  \begin{minipage}{6cm}
  \begin{center}
    Love of life and other short stories
    by Policarpa Salabarrieta
  \end{center}
  \end{minipage}}
```

Love of life and other short
stories by Policarpa Salabarrieta

In either case, if we want the resulting framed box centered, we have to include it inside another `center` environment.

Here is another example:

```
My list: \fbox{%
  \begin{Bflushleft}[b]
```

```
    Galanga root\\
    Coconut\\
    Tempeh
  \end{Bflushleft}}
```

| | |
|---|---|
| | Galanga root |
| | Coconut |
| My list: | Tempeh |

# 7 Lists

There are again two ways to box a list environment such as

`itemize`, `enumerate` or `description`.

You can put it in a `minipage` of pre-determined size and let TEX do the line breaking, or you can use

`Bitemize`, `Benumerate` or `Bdescription`

instead, and let the box adjust to the size of the longest line.

For example.

```
\fbox{%
  \begin{Bitemize}
    \item Groceries
    \item Clean hamster cages
    \item Pick up Peter
  \end{Bitemize}}
```

- Groceries
- Clean hamster cages
- Pick up Peter

Most of the usual list parameters are irrelevant except for `\itemsep` and `\labelsep`. These environments are also based on the `tabular` environment, and so each item should be balanced text. You can't write `\vspace{4pt}` either, but you can insert an extra amount of space before an item by writing `\item(4pt)` (or `\item(4pt)[label]` if you have a label). Also, you can break lines within an item using `\\` or `\\[`*dim*`]`. For example:

```
\fbox{%
  \begin{Bdescription}
    \item[David] Groceries
    \item[Eli]  Hamster cages\\
                Surreal numbers
    \item(3pt)[Doris] Pick up Peter
  \end{Bdescription}}
```

| | |
|---|---|
| **David** | Groceries |
| **Eli** | Hamster cages<br>Surreal numbers |
| **Doris** | Pick up Peter |

These environments also have an optional argument, `[t]` to align the box with the top line, and `[b]` to align the box with the bottom line.

```
To do:
\fbox{\setlength{\itemsep}{0pt}%
  \begin{Benumerate}[t]
    \item Groceries
    \item Hamster cages
    \item Pick up Peter
  \end{Benumerate}}
```

To do:

1. Groceries
2. Hamster cages
3. Pick up Peter

There is also a generic `\Blist` command that is analogous to LaTeX's `\list`. It has the same two obligatory arguments, plus a third optional `[t]` or `[b]` argument for changing the alignment.

# 8 Superimposing boxes

The command

`\boxput*(`$x$`,`$y$`){`*LR stuff1*`}{`*LR stuff2*`}`

puts *LR stuff1* either behind (the default) or in front of (with the `*`) *LR stuff2*.[8] The resulting box has the dimensions of *LR stuff2*.

The coordinates `(`$x$`,`$y$`)` determine where the center of *LR stuff1* is positioned. For example, `(0,0)` puts it at the center of *LR stuff2*, `(0,1)` puts it at the center-top, and `(-1,-1)` puts it in the bottom-left corner.

More generally, the origin of the coordinate system is at the center of *LR stuff2*, one unit in the vertical direction is half the vertical size of *LR stuff2*, and one unit in the horizontal direction is half the width of *LR stuff2*. Thus, $x$ and $y$ should always be numbers (without units such as `pt` or `cm`), with one exception: If $y$ is `b` or `B`, *LR stuff1* is positioned vertically at the baseline of *LR stuff2*. `(`$x$`,`$y$`)` is optional—the default is `(0,0)`.

Except for the funny coordinate system, `\boxput` is like the `\put` command in a `picture`. In particular, you can use `\makebox` in *LR stuff1* to fine-tune the positioning, and *LR-stuff1* can contain a `picture` environment.

You might use `\boxput` to put a "water mark" in the background of a box, or to put a label next to a box, when you don't want this label to take up any space. Here is a lazy example:

```
\boxput{\makebox(0,0){\Huge Censored!}}{\parbox{3in}{%
  The origin of the coordinate system is at the center of
  {\em LR stuff2}, and one unit in the x-direction
  is half the width of {\em LR stuff2}.}}
```

The origin of the coordinate system is at the center of *LR stuff2*, and one unit in the x-direction is half the width of *LR stuff2*.

Censored!

This would be a lot more interesting using PSTricks, with "Censored!" in the foreground, rotated 30 degrees, and red:

[8] You will only notice the difference between `\boxput` and `\boxput*` if you are using color implemented by `\special`'s

```
\boxput*{\rput{30}{\Huge\red Censored}}{\parbox{3in}{%
  blah blah}}
```

Here is another example using PSTricks:

```
\newcommand{\titledframe}[2]{%
  \boxput*(0,1){\psframebox*{#1}}%
    {\psframebox[framesep=12pt]{#2}}}
```

The following example illustrated roughly how it works, but "My title" does not blot out the frame behind it because this documentation does not use PSTricks.

```
\titledframe{My title}{%
  \parbox{2in}{The title is superimposed
    on the top part of the frame.}}
```

My title

The title is superimposed on the top part of the frame.

# 9 Framing a whole page

The commands

`\fancyput*(`$x$`,`$y$`){`*LR stuff*`}`
`\thisfancyput*(`$x$`,`$y$`){`*LR stuff*`}`

are pretty much like `\put` commands in a LaTeX picture environment whose origin is 1 inch down and to the right from the top left corner of the page.[9] The only differences are that (i) that any LR-mode material is permitted (including LaTeX `picture` environment, of course), (ii) the coordinate is optional `(0pt,0pt)` is substituted by default), and (iii) if the coordinate is included, you *must* specify the units.

`\thisfancyput` affects only the current page, and is a global declaration (analogous to `\thispagestyle`).

[9]Don't blame me for TeX's peculiar 1 inch margins.

If you include the optional `*`, then the command adds to, rather than replaces, other things that have been inserted with `\fancyput` or `\thisfancyput`.

These commands are particularly useful for framing a page, because you can get a frame that is, e.g., 1 inch from each side of the physical page without having to worry about what margins you are using for the document. Here is an example:

```
\thisfancyput(3.25in,-4.5in){%
  \setlength{\unitlength}{1in}\fancyoval(7,9.5)}
```

You could also use `\fancyput` to add some kind of "watermark" or background image (e.g., a light gray "DRAFT").

There are other commands that directly frame or in some other way box the page of text:

`\fancypage{`*cmds1*`}{`*cmds2*`}`
`\thisfancypage{`*cmds1*`}{`*cmds2*`}`

Each finished page, before adding the headers and footers, (and thus having width and height `\textwidth` and `\textheight`, is boxed with

*cmds1*{*pagebox*}

Thus, *cmds1* should be, or should end with, a command whose argument can be a box, such as `\fbox` or `\rotateleft`.

Then the headers and footers are added, using the new width of the page, and this is boxed with

*cmds2*{*pagebox*}

The same rules apply to *cmds2* as to *cmds1*.

Here is an example:

```
\thisfancypage{%
  \setlength{\fboxsep}{8pt}%
  \setlength{\shadowsize}{8pt}%
  \shadowbox}{}
```

**Warning:** The commands described in this section change LaTeX's output routine, and may not work with document styles that do the same. Also, bad arguments can cause serious errors with uninformative error messages.

# 10 Switching to landscape mode midstream

The most common reason to switch to landscape mode midstream is to rotate a float, such as a table or figure. This was discussed in Section ??.

If you want to rotate one or more pages, without rotating the headers and footers, use the

```
\begin{LandScape}{cmd}
  ...
\end{LandScape}
```

environment. *cmd* should be the command for rotating the page 90 degrees to the left or right. (E.g., `\rotateleft`, or `\rotate[l]`.)

If you want to rotate the headers, footers and margins as well, use the

```
\begin{Landscape}{paperwidth}{paperheight}{cmd}
  ...
\end{Landscape}
```

environment (the small `s` makes the difference) to rotate the pages left (counterclockwise), and use the `Landscape*` environment (same arguments) to rotate the pages right (clockwise). The three arguments are the width of the paper, the height of the paper, and the rotation command you are using. For example, if I have a portrait mode document using the US 8.5in by 11in paper, and if `\rotateleft{foo}` rotates `foo` 90 degrees counterclockwise, then I can write

```
\begin{Landscape}{8.5in}{11in}{\rotateleft}
```

You can use `\LandScape`, `\Landscape` and `\Landscape*`, rather then the `LandScape`, `Landscape` and `Landscape*` environments, if you want the rest of the document to be in landscape mode.

If your document is being printed in landscape mode, then these environments switch to portrait mode.

For example, suppose I have a landscape mode document, and I want to switch to portrait mode for the rest of the document, rotating the pages to the "right" with `\rotateright`. Then I would write

```
\Landscape*{11in}{8.5in}{\rotateright}
```

These environments switch the text width and height, leaving the margins exactly as they were before. It is quite possible that you want to make other changes to the page parameters after switching to landscape mode, but as Lamport points out the LaTeX *User's Guide and Reference Manual*, this generally doesn't work right in the middle of the document. `fancybox.sty` has a command `\UsePageParameters` which gets around this. It should be used right after you change the page parameters (and the page parameter changes should come right after the beginning of the landscape environment, or `\clearpage`).

**Warning:** The commands and environments described in this section change LaTeX's output routine, and may not work with document styles that do the same. Also, bad arguments can cause serious errors with uninformative error messages.

# 11 Verbatim

If you try to frame some verbatim text by typing

```
\fbox{%
  \begin{minipage}{5cm}
    \begin{verbatim}
      \My \Program \Listing
      if { foo } { bar } fi
    \end{verbatim}
  \end{minipage}}
```

you will get nonsense at best. This is because the argument to `\fbox` is read before the `\begin{verbatim}` is processed. But then it is too late for TeX to go back and interpret the contents of the verbatim environment literally rather than as special TeX commands and characters.

One solution is to use the `Sbox` environment:

```
\begin{Sbox}
\begin{minipage}{5cm}
\begin{verbatim}
\My \Program \Listing
```

```
if { foo } { bar } fi
\end{verbatim}
\end{minipage}
\end{Sbox}
\setlength{\fboxsep}{8pt}
\fbox{\TheSbox}
```

```
\My \Program \Listing
if { foo } { bar } fi
```

`fancybox.sty` also contains a command that “fixes” LaTeX’s LR-box commands for use with verbatim text:

`\VerbBox{`*cmd*`}{`*LR stuff*`}`

This is like

*cmd*{*LR stuff*}

but *LR stuff* can contain verbatim text.[10] For example:

```
\VerbBox{\fbox}{\verb+\foo{bar}+}
```

```
\foo{bar}
```

For footnotes, put the command `\VerbatimFootnotes` in the preamble, and then you can use verbatim commands or environments in the argument of `\footnote`. This is an optional feature because it might conflict with somebody else’s modification of the footnote system.

If you try to define your own framed verbatim environment with

```
\newenvironment{FramedVerb}%
  {\begin{Sbox}\begin{minipage}{5in}\begin{verbatim}}
  {\end{verbatim}\end{minipage}\end{Sbox}
    \setlength{\fboxsep}{8pt}\fbox{\TheSbox}}
```

[10]Or other tricks that involve `\catcode` changes, as occurs with some foreign language macros.

and then type

```
\begin{FramedVerb}
  if { foo } { bar } fi
\end{FramedVerb}
```

you will again run into trouble because after the `\begin{verbatim}`, LaTeX is searching for the literal string `\end{verbatim}` as the end of the verbatim text. It just skips right over the `\end{FramedVerb}` and may well continue to the end of the file or until it throws up.

`fancybox.sty` contains some verbatim environments that get around this problem and that have other advantages for LR-boxing verbatim listings, when compared to the standard LaTeX `verbatim` environment. Admittedly, many of their special features have nothing to do with boxes.

Here are the basic verbatim environments:

`Verbatim` Works pretty much like LaTeX's `verbatim`.

`LVerbatim` Like `Verbatim`, but `list` rather than `trivlist` is used to display the listing, and so it is indented from the left margin. (This is what I am using for verbatim listings in this document.)

`BVerbatim[`*pos*`]` Produces a box with the same width as the longest verbatim line. The baseline is in the center, unless you include the optional argument `[t]` for alignment with the top line or `[b]` for alignment with the bottom line.

`VerbatimOut{`*file*`}` Writes the verbatim text to *file*.

`SaveVerbatim{`*cmd*`}` Saves the verbatim text as *cmd*. *cmd* is defined globally, without checking whether *cmd* is already defined. Use obviously innocuous names like `\MyTemp`.

**Important:** For any of these verbatim environments, or new verbatim environments you define yourself (see below), nothing should come after `\begin{Verbatim}` or before `\end{Verbatim}` on the same line — not even spaces![11] If you put something after `\begin{Verbatim}` on the same line, it

[11]If you need to allow something to come before `\end{Verbatim}`, then you have two options:

- Put the command `\AltGetVerbatim` in the preamble. This switches to a scheme is simply ignored. However, if you put something after `\end{Verbatim}` on the same line, or if you misspell `\end{Verbatim}`, you will get an error such as

```
! File ended while scanning use of \Verbatim.
```

and the document will end at that point.

You can define new verbatim environments using `\newenvironment`. You just have to start the definition with

```
\VerbatimEnvironment
```

For example, here is the framed verbatim environment we tried earlier:

```
\newenvironment{FramedVerb}%
  {\VerbatimEnvironment
    \begin{Sbox}\begin{minipage}{5cm}\begin{Verbatim}}%
  {\end{Verbatim}\end{minipage}\end{Sbox}
    \setlength{\fboxsep}{8pt}\fbox{\TheSbox}}
```

Let's give it a try:

```
\begin{FramedVerb}
if { foo } { bar } fi
\end{FramedVerb}
```

```
if { foo } { bar } fi
```

Here are three commands for inputting a whole file verbatim. The file must end with a new line.

---

where anything preceding `\end{Verbatim}` is simply ignored. This can cause problems if you do really weird things with active characters or other commands within the verbatim environment (e.g., active conditionals that are not balanced within a line of verbatim text), but in this case you are probably a good enough hacker to use the next option.

- `\EndVerbatimTokens` is a token register that you can set to the tokens that should precede `\end{Verbatim}` on the same line, with their verbatim `\catcode`'s.

`\VerbatimInput{`*file*`}` Like `\Verbatim`.

`\LVerbatimInput{`*file*`}` Like `\LVerbatim`.

`\BVerbatimInput[`*pos*`]{`*file*`}` Like `\BVerbatim`.

Here are three commands for making use of verbatim text that has been saved to a command:

`\UseVerbatim{`*cmd*`}` Like `\Verbatim`.

`\LUseVerbatim{`*cmd*`}` Like `\LVerbatim`.

`\BUseVerbatim[`*pos*`]{`*cmd*`}` Like `\BVerbatim`.

The `SaveVerbatim` environment and the `\UseVerbatim` commands are useful for including verbatim text in the argument of `\marginpar`, `\fbox` and other commands. For example, here is another way to define the `FramedVerb` environment:

```
\newenvironment{FramedVerb}%
  {\VerbatimEnvironment
    \begin{SaveVerbatim}{\MyTemp}}%
  {\end{SaveVerbatim}%
    \setlength{\fboxsep}{8pt}%
    \fbox{\begin{minipage}{5cm}\UseVerbatim{\MyTemp}
      \end{minipage}}}
```

Here are some verbatim commands for short-pieces of (in-line) verbatim text:

`\Verb`*char literal char*

Like LaTeX's `\verb` command, but it will complain if it encounters a new line in *literal*.[12] For example:

```
The main use for the \Verb+SaveVerbatim+
environment and the \Verb+\UseVerbatim+
commands is to include $\ldots$
```

The main use for the `SaveVerbatim` environment and the `\UseVerbatim` commands is to include …

[12]Be careful that your word processing does not insert one for you.

`\UseVerb{`*cmd*`}` Like `\UseVerbatim`, but without any particular formatting. It is intended for including short pieces of literal text saved with `\SaveVerb` (below).[13]

`\SaveVerb[`*whatever*`]{`*cmd*`}`*char literal char*
This is like `\Verb`, but it saves *literal* as *cmd*, and then returns to the optional argument *whatever*. Like the `SaveVerbatim` environment, it defines *cmd* globally without checking whether *cmd* is already defined. Without the optional argument, the most common use is for including verbatim text in a `\marginpar`, `\section` or other command argument.

The optional argument can be used for special tricks. For example, all the listings of commands in this documentation use `\vitem` in a `description` environment, where `\vitem` is defined by:[14]

```
\newcommand{\vitem}%
  {\SaveVerb[{\item[\UseVerb{\MyTemp}]}]{\MyTemp}}
```

Whereas

```
\item[\Verb"\foo"]
```

would not work because after `\item` reads its argument it is too late to interpret `\foo` literally,

```
\vitem"foo"
```

does work because it is equivalent to

```
\SaveVerb{\MyTemp}"foo"\item[\UseVerb{\MyTemp}]
```

These environments and commands use various parameters that make it easy to customize their behavior. However, until you want to find the need for such customization, you might as well ignore the rest of this section.

Internally, `fancybox.sty` separates the reading and formatting of verbatim text. Most of the environments and commands perform both functions, but `SaveVerbatim` and `\SaveVerb` only read the text, while `UseVerbatim` (and company) and `\UseVerb` only format the text. `VerbatimOut` gets special treatment. The parameters that apply to each class of verbatim environment or command is listed in Table **??**.

[13]But it can also be used for multiple lines saved with the `SaveVerbatim` environment if you want to do the formatting yourself. E.g., try this in a tabbing environment with `\VerbatimTab` appropriately defined.

[14]The braces enclosing the optional argument of `\SaveVerb` prevent the `]` inside the argument from being mistaken for the end of the argument.

| *Where* | *What* |
|---|---|
| Environments that format | `\VerbatimSpace`<br>`\VerbatimTab`<br>`\VerbatimFont`<br>`\VerbatimFuzz`<br>`\EveryVerbatimLine`<br>`\EveryVerbatim`<br>`\ThisVerb` |
| Environments that read | `\EveryVerbatimCodes`<br>`\ThisVerbCodes` |
| `\Verb` and `\UseVerb` | `\VerbSpace`<br>`\VerbTab`<br>`\VerbFont`<br>`\EveryVerb`<br>`\ThisVerb` |
| `\Verb` and `\SaveVerb` | `\EveryVerbCodes`<br>`\ThisVerbCodes` |
| `VerbatimOut` | `\EveryVerbOutCodes`<br>`\ThisVerbCodes`<br>`\EveryVerbOutLine`<br>`\ThisVerb` |

Table 3: Parameters for the verbatim environments and commands.

All the parameters, including `\VerbatimFuzz`, are ordinary commands, and should be changed with `\renewcommand`.

Here is a description of each of the parameters for environments that format the verbatim text:

`\VerbatimSpace` The insertion text for spaces. The default is `\ `, which produces a blank space. Change it to `\ttspace` to get ␣.

`\VerbatimTab` The insertion text for tabs. The default is

```
\ \ \ \ \ \ \ \
```

`\VerbatimFont` The font to use for verbatim text. The default is `\tt`

`\VerbatimFuzz` This is the amount by which lines can be too long in a `Verbatim` or `LVerbatim` environment before you get overfull `\hbox` warnings. This threshold is usually .1pt, but the default definition of `\VerbatimFuzz` is `2pt` because verbatim lines won't break and are therefore often too long.

`\EveryVerbatimLine` This is inserted at the beginning of each line of verbatim environments or verbatim files. By default it does nothing. I like to indent each line in the verbatim environment in the input file by 2 spaces, so I define

```
\renewcommand{\EveryVerbatimLine}[2]{}
```

to eat those spaces. (But I have to remember to put in two spaces or space markers for blank lines too.) You might also use it to number the lines. For example:

```
\newcounter{VerbLineNo}
\renewcommand{\EveryVerbatimLine}%
  {\makebox[10pt][r]{%
     \stepcounter{VerbLineNo}%
     \tiny\rm\arabic{VerbLineNo}}%
  \hspace{10pt}}
\renewcommand{\EveryVerbatim}%
  {\setcounter{VerbLineNo}{0}}
\begin{SaveVerbatim}{\MyTemp}
\setlength{\fboxsep}{15pt}
\setlength{\mylength}{\linewidth}
\end{SaveVerbatim}
\fbox{\BUseVerbatim{\MyTemp}}
```

```
\setlength{\fboxsep}{15pt}
\setlength{\mylength}{\linewidth}
```

`\EveryVerbatim` Whatever else you want to say before formatting the verbatim text. By default, it does nothing.

`\ThisVerb` This is executed before any of the commands above, and then its value is cleared. Use this to customize a single verbatim formatting environment.

Here is a description of the parameters for environments that read the verbatim text:

`\EveryVerbatimCodes` This command is inserted just before reading the verbatim text. Use it to play with `\catcode`'s (see the *TEXbook*). For example, I might type

```
\renewcommand{\EveryVerbatimCodes}{\catcode`\"=14}
```

if I want to use " as a comment character in verbatim text.[15]

`\ThisVerbCodes` This command is executed before `\EveryVerbCodes`, and then it is cleared. Use this to fool with the `\catcode`'s of a single verbatim environment.

The parameters for `\Verb`, `\UseVerb` and `\SaveVerb` and the `VerbatimOut` environment are analogous to the similar commands for other environments.

Here is an example of the use of `\ThisVerb` to define a variant of `\Verb` that uses ␣ to mark spaces:

```
\newcommand{\SVerb}{%
  \renewcommand{\ThisVerb}%
    {\renewcommand{\VerbatimSpace}{\ttspace}}%
  \Verb}
```

[15]Here is another `\catcode` trick. We make " a short verbatim command, so that we can say `"foo"` instead of `\Verb"foo"`:

```
\def\MyQuote{"}    % \MyQuote is now the character ",
\def\temp{\Verb"}  %    in case I need it.
\catcode`\"=13     % Now " is like a command.
\let"\temp         % Now "foo" is like \Verb"foo"
\def\do{\noexpand\do\noexpand}       % Now " can be used in verbatim
\edef\dospecials{\dospecials\do\"}  % environments anyway.
```

Finally, without further comment, here are the definitions of the `example` and `example*` environments that were used for the examples in this document:

```
% 1. Save example verbatim to \jobname.tmp,
% 2. Input verbatim with \catcode`\"=14 (" is a comment).
% 3. Input again with \catcode\`"=9 (" is ignored).

\renewcommand{\EveryVerbatimLine}[2]{}

\renewcommand{\EveryVerbOutLine}[2]{}

\newcommand{\BeginExample}{%
  \VerbatimEnvironment\begin{VerbatimOut}{\jobname.tmp}}

\newcommand{\EndExample}{%
  \end{VerbatimOut}%.
  \renewcommand{\EveryVerbatimLine}{}%
  \renewcommand{\EveryVerbatimCodes}{\catcode`\"=14}%
  \LVerbatimInput{\jobname.tmp}%
  \catcode`\"=9}

\newenvironment{example}{\BeginExample}{\EndExample
  \begin{center}\input{\jobname.tmp}\end{center}}

\newenvironment{example*}{\BeginExample}%
  {\EndExample \input{\jobname.tmp}}

\newenvironment{example**}{\BeginExample}%
  {\EndExample \globaldefs=1 \input{\jobname.tmp}}
```

## 12 Changes

**Version 0.9, November 23, 1992** First release.

**Version 0.91, November 25, 1992**

1. `\shadowsize` is now a length, to be set with `\setlength` or `\addtolength`.

2. `\fancypage` split into `\fancypage` and `\fancyput`

**Version 0.92, December 20, 1992**

1. New `\boxput` command for superimposing boxes.
2. New `\VerbatimFootnotes` command.
3. New `\VerbBox` command for verbatim boxes.
4. New `Sbox` environment. Makes `\beginsbox` and `\endsbox` obsolete, but the latter have been retained for compatibility.

**Version 0.93, January 20, 1993**

1. New `\EndVerbatimTokens` for tokens that precede `\end{Verbatim}`.
2. New `\AltGetVerbatim`, to allow any tokens to precede `\end{Verbatim}`.
3. Fixed bug in `\BVerbatimInput` that caused second line to be repeated.
4. New `LandScape` environment, for rotating pages without rotating headers and footers.
5. Slide frames `Oval`, `oval`, `shadow` and `double` are defined when `seminar.sty` v0.93 (and maybe later) is loaded.

**Version 1.0, February 10, 1993**

1. Fixed bugs in `\boxput`.
2. New `Beqnarray` environment.