The moreverb package[*]

Robin Fairbairns (`rf10@cam.ac.uk`)
after
Angus Duggan, Rainer Schöpf and Victor Eijkhout

2008-06-03

Contents

1 This package **1**
1.1 Tab expansion 1
1.2 Line numbering 2
1.3 Miscellanea 2

2 The code of the package **2**
2.1 Initial code 3
2.2 Writing to a file 3
2.3 Tab expansion 3

1 This package

This package uses the facilities provide by the verbatim package in the LaTeX 2$_\varepsilon$ *tools* distribution to provide a number of things that were rejected as unnecessary in the development of that package. (Nevertheless, the tab-expansion code in this package responds to one of the FAQs of `comp.text.tex`)

The package provides things in three broad areas:

- Tab expansion and related stuff,
- Line numbering,
- Miscellaneous: writing verbatim to a file (for example, for later re-input), and 'boxed' verbatim.

1.1 Tab expansion

The package enables you to specify the expected width of the tabulation, and also allows input of files containing tabs.

`verbatimtab`

`\begin{verbatimtab}[`⟨*tab width*⟩`]` reproduces its body verbatim, with the tabs expanded to the given width (the default value is 8).

`\verbatimtabinput`

[*]This file has version number v2.3, last revised 2008/06/03

`\verbatimtabinput[`⟨*tab width*⟩`]{`⟨*file name*⟩`}` is a file input version of the `verbatimtab` environment.

`\verbatimtabsize`

The size of the tabs is stored in `\verbatimtabsize`, and persists between uses of the environments. (I.e., an optional argument to one of them applies to all subsequent ones.)
To replace the value other than by use of an optional argument, you need to say:
`\renewcommand\verbatimtabsize{`⟨*value*⟩`\relax}`
There are no promises offered as to the performance if you omit the `\relax`!

1.2 Line numbering

Line numbering is often useful when reproducing code examples (useful, that is, for those of us who don't want to pretty-print such snippets).

`listing`

`\begin{listing}[`⟨*interval*⟩`]{`⟨*start line*⟩`}` numbers the lines of its body. The argument ⟨*start line*⟩ specifies the starting line number. The optional argument ⟨*interval*⟩ specifies the number of lines between numbered lines: that is, every line whose number $= 0 \pmod{\langle interval\rangle}$ will be numbered in the output. (In addition, line number 1 will always be numbered.) The default value of the ⟨*interval*⟩ is 1 (i.e., every line will be numbered).

`listingcont`

`\begin{listingcont}` continues from the place where the last listing left off.
The style in which the label is set can be altered, for either environment, by re-defining `\listinglabel`. Both environments also expand tabs.
'`*`' versions of both the listing environments are provided; these do the usual `verbatim*` thing of outputting spaces as '␣', but don't expand tabs.

`listinginput`

`\listinginput[`⟨*interval*⟩`]{`⟨*start line*⟩`}{`⟨*filename*⟩`}` is a file input version of listing. There is no '`*`' form.

1.3 Miscellanea

`verbatimwrite`

`\begin{verbatimwrite}{`⟨*filename*⟩`}` writes all text in its body to a file, the name of which it is given as an argument.

`boxedverbatim`

`\begin{boxedverbatim}` puts the contents of a verbatim environment in a framing box. If you try to do this in a naïve way, you find that the verbatim lines have all become the width of the page, so that the box is, more often than not, a very poor fit to the text it surrounds.

`verbatimcmd`

The `verbatimcmd` environment was provided by the LaTeX2.09 and early LaTeX 2ε versions of this package. However, its capabilities are now provided by `alltt`, which is defined by the `alltt` package, now part of the LaTeX base distribution, and so `verbatimcmd` has been withdrawn.

2 The code of the package

⟨*moreverb⟩

2.1 Initial code

Load the verbatim package if it's not already loaded.

```
\@ifundefined{verbatim@processline}{\RequirePackage{verbatim}}{}
```

2.2 Writing to a file

verbatimwrite `\begin{verbatimwrite}{`⟨*filename*⟩`}` writes all text in its body to a file, the name of which it is given as an argument. (This code was written by Rainer Schöpf.)

```
\newwrite \verbatim@out
\def\verbatimwrite#1{%
  \@bsphack
  \immediate\openout \verbatim@out #1
  \let\do\@makeother\dospecials
  \catcode`\^^M\active \catcode`\^^I=12
  \def\verbatim@processline{%
    \immediate\write\verbatim@out
      {\the\verbatim@line}}%
  \verbatim@start}
```

```
\def\endverbatimwrite{%
  \immediate\closeout\verbatim@out
  \@esphack}
```

2.3 Tab expansion

We define a few auxiliary macros and counters for expanding tabs. They are used by the `listing` and `verbatimtab` environments.

```
\newcount\tab@position \newcount\tab@size
```

`\verbatimtabsize` used to be a counter, but that seems to me overkill (LaTeX uses too many counters as it is...).

```
\def\verbatimtabsize{8\relax}
```

\@xobeytab `\@xobeytab` puts enough spaces in to get us to the next nominal tab stop

```
\def\@xobeytab{%
  \loop
    \toks@\expandafter{\the\toks@\@xobeysp}%
    \advance\tab@position-1
  \ifnum\tab@position>0 \repeat
}
```

\@vobeytabs `\@vobeytabs` initialises use of `\@xobeytab`. Needs to be executed within a group, as mustn't be allowed to leak out into the wide world.

```
\begingroup
  \catcode`\^^I=\active
  \gdef\@vobeytabs{\catcode`\^^I\active\let^^I\@xobeytab}%
\endgroup
```

`\verbatim@tabexpand` `\verbatim@tabexpand`⟨*body of line*⟩`\@nil` processes every character of a line by tail recursion, counting the characters and juggling things when a tab is encountered. (What used to be called 'line imaging'. . .)

```
\def\verbatim@tabexpand#1{%
  \ifx#1\@nil
%    \showthe\toks@
    \the\toks@
    \expandafter\par
  \else
    \ifx#1\@xobeytab
      \@xobeytab
    \else
```

We can safely put `\@xobeysp` into the token register, since it does precisely what we need

```
      \toks@\expandafter{\the\toks@#1}%
      \advance\tab@position\m@ne
    \fi
    \ifnum\tab@position=0 \tab@position\tab@size \fi
    \expandafter\verbatim@tabexpand
  \fi
}
```

`listing` `\begin{listing}[`⟨*interval*⟩`]{`⟨*start line*⟩`}`

Defines a verbatim environment with numbered lines; the optional argument ⟨*interval*⟩ specifies the number of lines between numbered lines, and the argument ⟨*start line*⟩ specifies the starting line.

`listingcont` `\begin{listingcont}`

Continues from the place where listing left off. The style in which the label is set can be altered by re-defining `\listinglabel`.

'`*`' versions of both environments are provided.

`\listing@line` `\listing@line` holds the current line number; its default value is 1, so one can merrily use `listingcont` throughout a document if there's but one stream of verbatim text being written.

```
\newcount\listing@line \listing@line=1
```

`\listing@step` `\listing@step` is another case where a counter used to be used, to no very obvious utility, but using up a valuable count register. Again, the value is modal; the trailing `\relax` is necessary.

```
\def\listing@step{1\relax}
```

Adding an `\hbox` in front of the line causes a line break, so I[1] go through this rigmarole to get the lines aligned nicely. I probably missed some obvious reason why `\hbox`es don't work[2].

```
\def\listinglabel#1{\llap{\small\rmfamily\the#1}\hskip\listingoffset\relax}
\def\thelisting@line{%
```

[1]The personal pronoun was present in the comments in the original version of this package; I'm not sure who it relates to — RF

[2]It's because an `\hbox` in vertical mode makes a complete paragraph in its own right; this problem could be dealt with in the fullness of time, but just now. . .

```
  \setbox0\hbox{\listinglabel\listing@line}%
  \@tempcnta=\listing@line
  \divide\@tempcnta\listing@step \multiply\@tempcnta\listing@step
  \ifnum\listing@line=\@ne
    \unhbox0
  \else
    \ifnum\@tempcnta=\listing@line
      \unhbox0
    \else
      \hskip\wd0
    \fi
  \fi}
```

`\listingoffset` `\listingoffset` is the separation between the line number and the actual line being listed; default value is `1.5em`

```
\providecommand\listingoffset{1.5em}
```

Define `\listing` simply to suck in parameters and then to use `\listingcont`

```
\newcommand\listing[2][1]{%
  \global\listing@line=#2\relax
  \gdef\listing@step{#1\relax}
  \listingcont}
```

`\listingcont` is the business end of the two environments.

```
\def\listingcont{%
  \tab@size=\verbatimtabsize
  \def\verbatim@processline{\tab@position\tab@size
    \thelisting@line \global\advance\listing@line1
    \toks@{}%
    \expandafter\verbatim@tabexpand\the\verbatim@line\@nil}%
  \@verbatim\frenchspacing\@vobeyspaces\@vobeytabs\verbatim@start}
```

Nothing special at the end of the two environments.

```
\let\endlisting=\endtrivlist
\let\endlistingcont=\endtrivlist
```

Now the same rigmarole for the '`*`' versions.

```
\expandafter\newcommand\csname listing*\endcsname[2][1]{%
  \global\listing@line=#2\relax
  \gdef\listing@step{#1\relax}
  \csname listingcont*\endcsname}
\@namedef{listingcont*}{%
  \def\verbatim@processline{%
    \thelisting@line \global\advance\listing@line1
    \the\verbatim@line\par}%
  \@verbatim\verbatim@start}
```

Nobbut a bit of hassle in the name definitions for the end of the environments

```
\expandafter\let\csname endlisting*\endcsname\endtrivlist
\expandafter\let\csname endlistingcont*\endcsname\endtrivlist
```

`listinginput` `\listinginput[`⟨*interval*⟩`]{`⟨*start line*⟩`}{`⟨*filename*⟩`}` is a file input version of listing.

```
\def\listinginput{%
  \@ifnextchar[%]
    {\@listinginput}%
    {\@listinginput[1]}}
\begingroup
  \catcode`\~=\active \lccode`\~=`\^^M \lccode`\N=`\N
  \lowercase{\endgroup
    \def\@listinginput[#1]#2#3{\begingroup
      \global\listing@line=#2
      \gdef\listing@step{#1\relax}
      \tab@size=\verbatimtabsize
      \def\verbatim@processline{\tab@position\tab@size
        \thelisting@line \global\advance\listing@line1
        \toks@{}%
        \expandafter\verbatim@tabexpand\the\verbatim@line\@nil}%
      \@verbatim\frenchspacing\@vobeyspaces\@vobeytabs
      \def\verbatim@addtoline##1~{%
        \verbatim@line\expandafter{\the\verbatim@line##1}}%
      \openin\verbatim@in@stream=#3
      \ifeof\verbatim@in@stream
        \PackageWarning{moreverb}{No file #3.}%
      \else
        \do@verbatimtabinput
        \closein\verbatim@in@stream
      \fi
      \endtrivlist\endgroup
    \@doendpe
  }%
}
```

verbatimcmd

`verbatimcmd` was a verbatim environment with the exception of the escape and grouping characters \, {, }. This is (err) exactly the specification of the `alltt` environment, and that is in the alltt package that is now part of the base distribution.

```
\def\verbatimcmd{%
  \PackageError{moreverb}{The verbatimcmd environment is obsolete}%
                         {Use alltt (from the LaTeX required package
                           alltt) in place of verbatimcmd}%
}
\let\endverbatimcmd\relax
```

boxedverbatim

`boxedverbatim` puts the contents of a verbatim environment in a framing box.
   (Written by Victor Eijkhout.)
Bug fixes:

- David Carlisle 1995-12-28, dealing with spacing issues (iirc)
- Moretn Høgholm 2008-06-01, positioning of frame in lists

First, redefine 'processline' to produce only a line as wide as the natural width of the line

```
\def\boxedverbatim{%
  \def\verbatim@processline{%
    {\setbox0=\hbox{\the\verbatim@line}%
    \hsize=\wd0 \the\verbatim@line\par}}%
```

Now save the verbatim code in a box

```
  \@minipagetrue           % DPC
  \@tempswatrue            % DPC
  \@totalleftmargin\z@     % MH
  \setbox0=\vbox\bgroup \verbatim
}
```

At the end of the environment, we (umm) simply have to stick the results into a frame.

```
\def\endboxedverbatim{%
  \endverbatim
  \unskip\setbox0=\lastbox % DPC
```

Now everything's in the box, so we can close it…

```
  \egroup
```

To change the code for centring, the next line needs a spot of hacking.

```
  \fbox{\box0}%
}
```

verbatimtab `\begin{verbatimtab}[`⟨*tab width*⟩`]` is a verbatim environment which expands tab characters; the optional argument specifies the distance between tab stops.

Executing `\obeylines` before looking for the optional argument prevents an empty first line of the environment becoming a `\par` token (this bug was reported by Werner Lemberg).

```
\newenvironment{verbatimtab}{\obeylines\@verbatimtab}{\endtrivlist}
```

Process the optional argument of the `verbatimtab`, now that we have protected ourselves from the dreaded `\par` tokens

```
\newcommand\@verbatimtab[1][\verbatimtabsize]{%
  \do@verbatimtab{#1}{%
     \@verbatim\frenchspacing\@vobeyspaces\@vobeytabs\verbatim@start}%
}
```

\do@verbatimtab Prepare a tabbing environment; `#1` is the value of the tab size (generally, originally, an optional argument), `#2` is the 'startup commands' to execute once an appropriate definition of `\verbatim@processline` has been established:

```
\def\do@verbatimtab#1#2{%
  \tab@size=#1
  \def\verbatim@processline{\tab@position\tab@size
    \toks@{}%
    \expandafter\verbatim@tabexpand\the\verbatim@line\@nil}%
  #2%
}
```

\verbatimtabinput `\verbatimtabinput[`⟨*tab width*⟩`]{`⟨*file name*⟩`}` is a file input version of the `verbatimtab` environment.

We use the input stream acquired by the verbatim package; we did after all require it to be loaded. (One has to admit that the name of that stream isn't actually part of the package's defined interface, but on the other hand there's no particular likelihood that it will ever change.)

We didn't (originally) use fancy features of `\newcommand` since the definition was inside a group, and hence global. So … 'traditional' code to provide a command with an optional argument (which may no longer be necessary):

```
147 \def\verbatimtabinput{%
148   \@ifnextchar[%]
149     {\@verbatimtabinput}%
150     {\@verbatimtabinput[\verbatimtabsize]}}
151 \begingroup
152 \catcode`\~=\active \lccode`\~=`\^^M \lccode`\N=`\N
153 \lowercase{\endgroup
154   \def\@verbatimtabinput[#1]#2{\begingroup
155     \do@verbatimtab{#1}{%
156       \@verbatim\frenchspacing\@vobeyspaces\@vobeytabs}%
157     \def\verbatim@addtoline##1~{%
158       \verbatim@line\expandafter{\the\verbatim@line##1}}%
159     \openin\verbatim@in@stream=#2
160     \ifeof\verbatim@in@stream
161       \PackageWarning{moreverb}{No file #2.}
162     \else
163       \@addtofilelist{#2}%
164       \do@verbatimtabinput
165       \closein\verbatim@in@stream
166     \fi
167   \endtrivlist\endgroup\@doendpe}%
168 }
```

`\do@verbatimtabinput` Written-out (tail recursion) loop for reading the file

```
169 \def\do@verbatimtabinput{%
170   \read\verbatim@in@stream to \verbtab@line
171   \ifeof\verbatim@in@stream
172   \else
173     \expandafter\verbatim@addtoline\verbtab@line
174     \verbatim@processline
175     \verbatim@startline
176     \expandafter\do@verbatimtabinput
177   \fi
178 }
179 ⟨/moreverb⟩
```

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.

Symbols

`\@vobeytabs` 24
`\@xobeytab` 18

B

`boxedverbatim` (environment) . *2*, 120

D

`\do@verbatimtab` . . . 140
`\do@verbatimtabinput` 169

E

environments:
  `boxedverbatim` . . 120
  `listing` 44
  `listingcont` 44
  `verbatimtab` 135
  `verbatimwrite` . . . 3
environments:boxedverbatim
  `boxedverbatim` . . . *2*
environments:listing
  `listing` *2*
environments:listingcont
  `listingcont` *2*
environments:verbatimcmd

verbatimcmd *2*
environments:verbatimtab

verbatimtab *1*
environments:verbatimwrite

verbatimwrite . . . *2*

L

listing (environment) *2*, 44
\listing@line 44
\listing@step 45
listingcont (environment) *2*, 44
\listinginput *2*, 85
\listingoffset 60

V

\verbatim@tabexpand 28
\verbatimcmd 114
verbatimcmd (environment) *2*
verbatimtab (environment) *1*, 135
\verbatimtabinput *1*, 147
\verbatimtabsize . . . *2*
verbatimwrite (environment) . . . *2*, 3