The verbatimbox Package

Routines for placing stylized verbatim text into boxes, useful in places where the verbatim environment is inaccessible. Secondarily, for adding vertical buffer around an object.

Steven B. Segletes
SSegletes@verizon.net
August 8, 2023
v3.2

1 Description and Commands

The verbatimbox package allows verbatim material to be placed into \LaTeX{} boxes, for later recall. This function is useful in several situations, primarily in places where the verbatim environment is otherwise inaccessible (for example, in the midst of a tabular environment). It is also useful if a given verbatim text needs to be recalled multiple times within a document.

One very nice feature of this package is the optional argument to both environments and macros that allows custom stylization of the verbatim text (fontshapes, sizes, numbering lines, \textit{etc.}) through the use of pre-commands. For those who prefer this form of optional-argument customization, the environments and macros are \textit{also} provided in a “no-box” form that is output directly, rather than being saved in a box. The no-box forms can be used when the verbatim has to span across page breaks.

While there is application for its use within the verbatimbox application, this package also offers an independent command for conveniently providing vertical buffer space above and below an object.

The environments and macros provided by this package are given as follows:

\begin{verbatimbox}{pre-commands}\ldots\end{verbatimbox}
\begin{myverbatimbox}{pre-commands}{token}\ldots\end{myverbatimbox}
\verbfilebox[pre-commands]{filename}
\theverbatimbox[t]
\boxtopsep = \textit{length}
\boxbottomsep = \textit{length}
\addvbuffer[length]{object} OR
   \addvbuffer[above-length below-length]{object}
\begin{verbnobox}{pre-commands}\ldots\end{verbnobox}
\verbfilenobox[pre-commands]{filename}
In addition, there is a counter, \texttt{VerbboxLineNo}, which contains the current verbatim line number being processed. It’s use as part of the optional pre-
commands to \texttt{verbbox}, \texttt{myverbbox}, or \texttt{verbatimbox} (in the form of
\texttt{\arabic{VerbboxLineNo}:})
will cause the lines of the environment to be numbered.

\section{The \texttt{verbbox} Environment}

The \texttt{verbbox} environment places the content of the environment into a verbatim box. The box is not printed when the environment is closed. However, the box can be later recalled (even in a verbatim-restricted environment) by way of the command \texttt{\theverbbox}. Being placed into a box, it is important to remember that a \LaTeX box cannot span across page boundaries.

The width of the box into which the contents of the environment are placed is the width of the longest line in your environment, rather than the width of the page. This feature can be useful if the box is being recalled inline with other text, as in this Example, shown immediately to the left.

The environment has one optional argument, which can contain commands that will be executed before each line of verbatim text is output. Thus, they can include font size, series, shape, or family changes affecting the appearance of the verbatim text in the box. They can also include printed matter, like a bullet

\begin{verbbox}\[
\begin{itemize}
  \item first line
  \item second line
  \item third line
\end{itemize}\]
\end{verbbox}
\theverbbox

or, as mentioned earlier, the lines can be numbered:

\begin{verbbox}\[
\begin{enumerate}
  \item first line
  \item second line
  \item third line
\end{enumerate}\]
\end{verbbox}
\theverbbox
The customization can make use of the VerbboxLineNo counter in ways that make it line specific. Note that the optional argument must be contained on a single line.

\newcommand*{\ifline}[3]{% 
  \ifthenelse{\value{VerbboxLineNo} = #1}{#2}{#3}}
\newcommand*{\highlight}{\color{red}\rmfamily\itshape\bfseries\large\bullet~}
\newcommand*{\nohighlight}{\arabic{VerbboxLineNo}:\hspace{1ex}}
\begin{verbbox}\[\ifline{2}{\highlight}{\nohighlight}\]
Line 1
Line 2
Line 3
\end{verbbox}
\theverbbox

1: Line 1
• \textit{Line 2}
3: Line 3

1.2 The myverbbox Environment

With the verbbox environment, one is limited to one verbatimbox at a time, since each new environment invocation overwrites the prior box. A new environment, myverbbox, has therefore been introduced with version 3.0 of the package. It is very similar to verbbox, except that it is not recalled with the use of \theverbbox. In addition to the optional argument which functions like that in verbbox, myverbbox must be supplied a mandatory argument which is the command name that will recall the box contents.

This environment was created for the situation when there is a need to recall more than one verbatimbox in a given restricted environment. Here is an example where two verbatimboxes must be inserted into the tabular environment (where verbatim is prohibited):

\begin{myverbbox}{\vtheta}\theta\end{myverbbox}
\begin{myverbbox}{\valpha}\alpha\end{myverbbox}
\begin{tabular}{|c|c|} \hline
\valpha & $\alpha$ \\ \hline
\vtheta & $\theta$ \\ \hline
\end{tabular}
In this example the command $\alpha$ will recall a verbatim box with the verbatim content $\alpha$. Likewise for $\theta$.

### 1.3 The \verbfilebox Command

The \verbfilebox command is comparable to the aforementioned environments, except that it is a command which takes the contents of a file specified in the mandatory argument and places it in a verbatim box. Like verbbox, the boxed content can be recalled with an invocation of theverbbox. The optional argument contains pre-commands and functions in the same manner as described for the verbbox environment.

### 1.4 The theverbbox Command

As already mentioned in the prior sections, the theverbbox command is issued to recall the contents of a box created by either the verbbox environment or the verbfilebox command. This command can be issued in places where the verbatim environment is otherwise inaccessible, such as in the tabular environment or inside footnotes.

There is a [t] option to this command, which preconditions theverbbox for environments where it will be framed, by adding 3pt (by default) of space above the boxed content, since the top of the box is otherwise clipped to the verbatim contents. So in the following example, the left invocation of theverbbox is done without the [t] option, whereas for the right invocation, it is done with the [t] option. As you can see, the framing on the right-hand box is more symmetric.

\begin{tabular}{|l|l|}
\hline
\texttt{\boxtopsep} & \texttt{\boxbottomsep} \\
\hline
1: first line & 1: first line \\
2: second line & 2: second line \\
3: third line & 3: third line \\
\hline
\end{tabular}

If one wishes a different vertical-buffer preconditioning of theverbbox, one can either reset the lengths \boxtopsep and \boxbottomsep or else one can use the \addvbuffer command described below.
1.5 The \addvbuffer Command

The prior section already showed how when a verbatimbox is framed, it may need some buffer space added above the box which is otherwise (by design) clipped to the verbatim content. While the \[t\] option will work for \theverbbox, that option does not exist for any of the boxes created through the myverbbox environment. Furthermore, the user may sometimes wish to add an arbitrary vertical buffer above and below an object, without changing the underlying \boxtopsep and \boxbottomsep lengths associated with the \[t\] option of \theverbbox.

The \addvbuffer command is made to do this, and is useful in many situations inside and outside of verbatimbox manipulations. The mandatory argument to the command is the object over and under which to place vertical buffer. As of version 3.0 of this package, the command comes with an optional argument to specify how much buffer to use. If the optional argument is omitted, the buffer will be the amounts specified by the \boxtopsep and \boxbottomsep lengths.

The optional argument can take two forms. If the optional argument comprises a single length, that length is symmetrically added above and below the object. So, for example, \fbox{\addvbuffer[5pt]{\theverbbox}} will produce

| 1: first line |
| 2: second line |
| 3: third line |

On the other hand, the optional argument may comprise two lengths, in which case the first length is added above the object and the second is added below the object. Thus, \fbox{\addvbuffer[15pt 5pt]{\theverbbox}} will produce

| 1: first line |
| 2: second line |
| 3: third line |

But \addvbuffer need not only be used for \theverbbox. It can be used on other objects, for example a \parbox

| This is a test of the emergency broadcast system |
or even on just plain text \fbox{\addvbuffer[15pt 5pt]{A test}}:

\begin{quote}
\fbox{A test}
\end{quote}

It can even be used to remove vertical space from an objects size, by adding negative buffer \fbox{\addvbuffer[-5pt -5pt]{A test}}:

\begin{quote}
\fbox{A test}
\end{quote}

This ability to work on a variety of objects makes \addvbuffer a powerful command in many applications.

Likewise, if two length variables are to be used in the optional argument, a non-absorbed space must exist between them. Thus, if the first of the two lengths is a macro or length register, an empty group {} should be placed at the end of it, so that the subsequent space is not absorbed:

\begin{quote}
\fbox{\addvbuffer{baselineskip}{}{.5\baselineskip}{A test}}
\end{quote}

\begin{quote}
\fbox{A test}
\end{quote}

One may not use a “\ ” hard space to separate the two arguments, but instead the method prescribed above.

1.6 The “nobox” Alternatives

Alternatives are provided for the \verbbox environment and the \verbfilebox macro, which output the content directly, rather than place it into a box. These alternatives are named \verbbox and \verbfilebox. Like their boxed counterparts, these alternatives make use of the convenient optional pre-commands that allow for custom stylization of the verbatim content.

Because they are not placed into a box, but instead output directly, their content cannot be recalled at a later time in the document. While the boxed versions have a particular use in environments where verbatim is inaccessible, the no-boxed alternatives may be used when the content must span a page break.
2 Quirks

The use of a single optional argument for the verbbox environment has caused problems in the past, if the first item in the environment was a command, rather than text. This problem has been resolved as of version 3.0 of the package (for backward compatibility, we have kept the earlier incarnation of verbbox, now renamed as origverbbox). However, a few minor quirks remain even with the new version (which, incidentally, can be avoided by using the myverbbox environment). Note that the commands of this package do not enclose the created boxes in frames.¹ Thus, frames shown in these examples are for illustrative purposes only.

The primary quirk of verbbox is that its optional argument does not take effect until the second line (on the line following the environment invocation), so that

\begin{verbbox}[	extsc{LARGE}]first line  
second line  
\end{verbbox}  
\fbox{\theverbbox}

would look like:

\textbf{first line}  
\textbf{second line}

The solution, in the presence of an optional argument, is to begin the first line of the environment contents on the following line. Thus,

\begin{verbbox}[	extsc{LARGE}]  
first line  
second line  
\end{verbbox}  
\fbox{\theverbbox}

will yield the expected result:

\textbf{first line}  
\textbf{second line}

No such requirement is necessary if there are no optional arguments. So, for example, \begin{verbbox}Hello World\end{verbbox} will produce, upon issue of \theverbbox, the box “Hello World.” As with all verbatim environments, the \end{verbbox} must be followed by a linebreak, or the subsequent text on that line will be lost.

¹In all these examples, the verbbox material is explicitly placed in an \fbox, to expressively show the limits of the boxed material. Throughout this section, \fboxsep has been set to 0pt.
The `myverbbox` environment does not suffer the above quirk, so that
\begin{myverbbox}\[\scriptsize\]{\mybox}Hello World\end{myverbbox}
gives, upon issue of \mybox, the expected result: Hello World.

The optional argument of `verbbox` may contain both printing or nonprinting
material. The printing material will appear (as LaTeX, not as `verbatim`) at
the beginning of every line of the `verbbox`. However, it is recommended to place
nonprinting material at the lead of the optional argument, or the box width
may be improperly set, as in this following example:

```
\begin{verbbox}\[\rmfamily\textsc{Debug}:\footnotesize\]
  first line
  second line
\end{verbbox}
\fbox{\theverbbox}
Debug: first line
Debug: second line
```

Note how the box does not extend to the end of the text line. When placing
printing commands before non-printing commands that change the fontsize,
the reverse is also possible, where the box is set too large for the actual text.

The `myverbbox` environment behaves identically in this regard. Below we show
the prior example, this time created in the `myverbbox` environment, except that
the non-printing optional material has been placed before the printing optional
material, so that the box size is correctly interpreted:

```
\begin{myverbbox}\[\texttt{Debug}:\footnotesize\]
  first line
  second line
\end{myverbbox}
\fbox{\themyverbbox}
```

There are differences, however, in how `verbbox` and `myverbbox` process
the optional argument, which relate to overcoming the problems associated with
a single optional argument for the `verbbox` environment. In particular, while the
`myverbbox` optional argument should behave as one would otherwise expect, the
optional argument of `verbbox` will ignore spaces (which can be overcome with\hskip\hspace\relax) and will choke on the dollar symbol ($) as a way to enter and exit
math mode (which can be overcome by using the \(\relax\) syntax instead). Thus,
an optional argument like
```
[\arabic{VerbboxLineNo}\cdot\hspace{1em}]$\ldots$
```
which works fine for `myverbbox`, would need to appear as
```
[\arabic{VerbboxLineNo}\hspace{1em}\cdot](\hspace{1em})\hspace{1em}$\ldots$
```
in the `verbbox` environment. In that case, it would number the lines of the
environment in the following manner:

```
1. first line
2. second line
3. third line
```
There is also a quirk with the macro \addvbuffer. If the \addvbuffer macro adds negative space below an argument, the argument is vertically shifted downward by the same amount. For example, compare the following constructions, \fbox{gb} and \fbox{\addvbuffer[-3pt]{gb}}: \fbox{gb} \quad \fbox{gb}. In the latter case, the box is bottom-trimmed, but the result is shifted downward. This adverse outcome can generally be overcome with a countering \raisebox, \fbox{gb} \quad \fbox{gb}. However, if the \raisebox would otherwise place the new box above the baseline, the effect of the \addvbuffer on the bottom of the box is spoiled, as in the comparison given by the constructions \fbox{\addvbuffer[-3pt]{X}} versus \fbox{\raisebox{3pt}{\addvbuffer[-3pt]{X}}} which produces this result: \fbox{X} \quad \fbox{X}. Note how 3pt are not trimmed from the bottom of the second box. For this reason, the \raisebox remedy has not been incorporated into the definition of \addvbuffer; instead, the quirk has been allowed to stand.

3 Acknowledgements

I would like to thank Dr. David Carlisle for his great assistance in helping me to understand the nuances of optional arguments used in verbatim environments, allowing me to correct a longstanding bug in the \verbbox environment:

http://tex.stackexchange.com/questions/109030/optional-arguments-in-verbatim-environments

I would also like to acknowledge the use of three lines of code created by Prof. Enrico Gregorio, to strip the leading backslash from a \LaTeX command name

http://tex.stackexchange.com/questions/42318/removing-a-backslash-from-a-character-sequence

4 Code Listing

\def\verbatimboxVersionNumber{v3.2}
\ProvidesPackage{verbatimbox}
[2023/08/08 \verbatimboxVersionNumber]
Routines for placing verbatim text into boxes, useful in places where the verbatim environment is inaccessible. Secondarily, for adding vertical buffer around an object.]
\%
\% This work may be distributed and/or modified under the
\% conditions of the \LaTeX Project Public License, either version 1.3
\% of this license or (at your option) any later version.
\% The latest version of this license is in
\% http://www.latex-project.org/lppl.txt
\% and version 1.3c or later is part of all distributions of \LaTeX

9
\NeedsTeXFormat{LaTeX2e}
\usepackage{readarray}
% Following 3 lines thanks to Prof. Enrico Gregorio, from:
% http://tex.stackexchange.com/questions/42318/
% removing-a-backslash-from-a-character-sequence
\begingroup\lccode'|='\\lowercase{\endgroup\def\removebs#1{\if#1|\else#1\fi}}
\@ifundefined{verbatim@processline}{\RequirePackage{verbatim}}{}
\newcommand{\@macro@name}[1]{\expandafter\removebs\string#1}
%
\global\newsavebox{\savedverbbox}
\newcounter{VerbboxLineNo}

% The verbbox environment is based on
% the boxedverbatim environment found in moreverb.sty
% The optional argument allows the user to modify properties of the text
% such as fontsize
%
\newenvironment{origverbbox}[1][]{%
\def\verbatim@processline{%
 {\setbox0=\hbox{\the\verbatim@line}\
 \hsize=\wd0 \the\verbatim@line\par}}%
 @minipagetrue%
 @tempswatrue%
 \setbox0=\vbox{bgroup #1 \verbatim}
}
%
@endverbatim
\unskip\setbox0=\lastbox %
\egroup
\global\sbox{\savedverbbox}{\box0}
}

% David Carlisle provided the \verbbox@inner approach to avoid
% problem when no optional argument is provided to verbbox environment:
% http://tex.stackexchange.com/questions/109030/optional-arguments-in
% -verbatim-environments
\newcommand\verbbox@inner[1][]{{\nfss@catcodes\scantokens{\gdef\@tmp{#1}}}}
\def\@tmp{}

\newenvironment{verbbox}{%$
\setcounter{VerbboxLineNo}{-1}$%
% FOR SOME REASON, USING \my@par INSTEAD OF \par PREVENTS EXTRA SPACE
% ABOVE verbbox WHEN USING OPTIONAL ARGUMENTS
% let\my@par\par%
\def\verbatim@processline{%
% FIRST \@tmp APPLIES OPTIONAL ARGUMENT TO EACH VERBATIM LINE
% SECOND \@tmp MAKES SURE ANY PRINTED MATTER OF OPTIONAL ARGUMENT
% IS ACCOUNTED FOR IN VERBATIM BOX WIDTH
{\addtocounter{VerbboxLineNo}{1}}%
\@tmp\setbox0=\hbox{\@tmp\the\verbatim@line}%
\hsize=\wd0 \the\verbatim@line/my@par}%
@minipagetrue%
\begin{myverbbox}[12][12]\end{myverbbox}

% The myverbbox environment is altered from verbbox environment
% The optional argument allows the user to modify properties of
% the text such as fontsize
% The mandatory argument is a command which is formed so as to
% regurgitate the boxed content created within the environment
%
\newenvironment{myverbbox}[2][12][12]\end{myverbbox}

% THE FIRST #1 ACCOUNTS FOR NON-PRINTING COMMANDS; THE SECOND #1 IS FOR
% PRINTED OPTIONAL MATERIAL
% \addtocounter{VerbboxLineNo}{1}%
% #1=\setbox0=\hbox{#1}{the\verbatim@line}%
% \textsize=\wd0 the\verbatim@line\par}%
% \@minipagetrue%
% \@tempswatrue%
% \global\edef\sv@name{\@macro@name{#2}}%
% \@ifundefined{\sv@name content}{%
% \expandafter\newsavebox\expandafter{\sv@name content}%
% \expandafter\global\expandafter\edef\csname\sv@name\endcsname{\usebox{\sv@name content}}%
% \@tempswatrue%
% \@tempswatrue%
% \global\edef\sv@name{\@macro@name{#2}}%
% \@ifundefined{\sv@name content}{%
% \expandafter\newsavebox\expandafter{\sv@name content}%
% \expandafter\global\expandafter\edef\csname\sv@name\endcsname{\usebox{\sv@name content}}%
% \setbox0=\vbox{\begin{verbatim}}%
% \end{verbatim}%
% \unskip\setbox0=\lastbox%
% \egroup%
% \global\sbox{\sv@name content}{\box0}%
% }%
% The following two "nobox" commands are basically versions of
% \verbatiminput and \verbatim that have been adapted to take the
% optional argument style of this package. No boxes are created,
% but breaking across page boundaries is not a problem here, as
% it would be with a box.

\newcommand\verbfilenobox[2][{}]{%  
  \setcounter{VerbboxLineNo}{0}  
  \def\verbatim@processline{%  
    \addtocounter{VerbboxLineNo}{1}  
    \setbox0=\hbox{\verbatim@line}  
    \hsize=\wd0 \verbatim@line\par}  
  \verbatiminput{#2}  
  \let\verbatim@processline\sv@verbatim@processline}  
\newenvironment{verbnobox}{%  
  \setcounter{VerbboxLineNo}{-1}  
  \let\my@par\verbatim@line\par}{}