Grzegorz ‘Natror’ Murzynowski

The gmverb Package*

December 1, 2006

This is (a documentation of) file gmverb.sty, intended to be used with LaTeX 2$_\varepsilon$ as a package for a slight redefinition of the `\verb` macro and `verbatim` environment and for short verb marking such as `|\mymacro|`.

Written by Grzegorz ‘Natror’ Murzynowski,
natror at o2 dot pl

Many thanks to my TeX Guru Marcin Woliński for his TeXnical support.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{gmverb}
    [2006/12/01␣v0.78␣After␣shortvrb␣(FM)␣but␣my␣way␣(GM)]
```

Contents

Intro, Usage

This package redefines the `\verb` command and the `verbatim` environment so that the verbatim text can break into lines, with `%` (or another character chosen to be the comment char) as a ‘hyphen’. Moreover, it allows the user to define his own `verbatim`-like environments provided their contents would be not *horribly* long (as long as a macro’s argument may be at most).

This package also allows the user to declare a chosen char(s) as a ‘short verb’ e.g., to write `|\a\verbatim\example|` instead of `\verb|\a\verbatim\example|`.

The gmverb package redefines the `\verb` command and the `verbatim` environment in such a way that ␣, `{` and `\` are breakable, the first with no ‘hyphen’ and the other two with the comment char as a hyphen. I.e. `{`⟨*subsequent text*⟩`}` breaks into `{%`

[*] This file has version number v0.78 dated 2006/12/01.

⟨*subsequent text*⟩`}` and ⟨*text*⟩`\mymacro` breaks into ⟨*text*⟩`%`
`\mymacro`.

`\fixbslash` (If you don't like linebreaking at backslash, there's the `\fixbslash` declaration (observing the common scoping rules, hence OCSR) and an analogous declaration for the
`\fixlbrace` left brace: `\fixlbrace`.)

The default 'hyphen' is `%` since it's the default comment char. If you wish another
`\VerbHyphen` char to appear at the linebreak, use the `\VerbHyphen` declaration that takes `\`⟨*char*⟩ as the only argument. This declaration is always global.

`\verbeolOK` Another difference is the `\verbeolOK` declaration (OCSR). Within its scope, `\verb` allows an end of a line in its argument and typesets it just as a space.

As in the standard version(s), the plain `\verb` typesets the spaces blank and `\verb*` makes them visible.

`\MakeShortVerb` Moreover, gmverb provides the `\MakeShortVerb` macro that takes a one-char control sequence as the only argument and turns the char used into a short verbatim delimiter, e.g., after `\MakeShortVerb*\|` (as you guess, the declaration has its starred version, which is for visible spaces, and the non-starred for the spaces blank) you may type `|\mymacro|` to achieve `\mymacro` instead of typing `\verb+\mymacro+`. Because the char used in this example is my favourite and used just this way by DEK in the The TEXbook's
`\dekclubs` format, gmverb provides a macro `\dekclubs` as a shorthand for `\MakeShortVerb*\|`.

Be careful because such active chars may interfere with other things, e.g., the `|` with the vertical marker in tables and with the tikz package. If this happens, you can declare
`\DeleteShortVerb` e.g., `\DeleteShortVerb\|` and the previous meaning of the char used shall be restored.

One more difference between gmverb and shortvrb is that the chars `\active`ated by `\MakeShortVerb` in the math mode behave as if they were 'other', so you may type e.g., `$|$` to achieve | and `+` `\active`ated this way is in the math mode typeset properly etc.

`\dekclubs` There's one more declaration provided by gmverb: `\dekclubs`, which is a shorthand for `\MakeShortVerb*\|` .

As many good packages, this also does not support any options.

Installation

Just put the gmverb.sty somewhere in the texmf/tex/latex branch. Creating a texmf/tex/latex/gm directory may be advisable if you consider using other packages written by me.

Then you should refresh your TEX distribution's files' database most probably.

Contents of the gmverb.zip Archive

The distribution of the gmverb package consists of the following four files.

gmverb.sty
README
gmverbDoc.tex
gmverbDoc.pdf

Compiling the Documentation

The last of the above files (the .pdf, i.e., *this file*) is a documentation compiled from the .sty file by running LATEX on the gmverbDoc.tex file twice, then MakeIndex on the gmverb.idx file, and then LATEX on gmverbDoc.tex once more.

MakeIndex shell command:

```
makeindex -r gmverbDoc
```

The `-r` switch is to forbid MakeIndex to make implicit ranges since the (code line) numbers will be hyperlinks.

Compiling the documentation requires the packages: gmdoc (gmdoc.sty and gmdocc.cls), gmverb.sty, gmutils.sty and also some standard packages: hyperref.sty, color.sty, geometry.sty, multicol.sty, lmodern.sty and fontenc.sty that should be installed on your computer by default.

If you had not installed the mwcls classes (available on CTAN and present in TEX Live e.g.), the result of your compilation might differ a bit from the .pdf provided in this .zip archive in formatting: If you had not installed mwcls, the standard article.cls class would be used.

The Code

Preliminaries

```
\firstofone     \long\def\firstofone#1{#1}
\afterfi        \long\def\afterfi#1\fi{\fi#1}
```

The standard `\obeyspaces` command is only re`\catcode`ing of the space to be $_{13}$. Since we'll know a bit of where these macros are used, we know we have also to (re)define such an active space to be some space.

```
                \begin{catcode}`\␣\active
\gmobeyspaces   \gdef\gmobeyspaces{\catcode`\␣\active\let␣\␣}
                \end{catcode}
```

(The above three preliminary definitions are present also in gmutils.)

```
                \bgroup
                \@makeother\%
                \firstofone{\egroup
\twelvepercent    \def\twelvepercent{%}}
```

Someone may want to use another char for comment, but we assume here 'orthodoxy'. Other assumptions in gmdoc are made. The 'knowledge' what char is the comment char is used to put proper 'hyphen' when a `verbatim` line is broken.

```
\verbhyphen     \let\verbhyphen\twelvepercent
```

Provide a declaration for easy changing it. Its argument should be of \⟨*char*⟩ form (of course, a ⟨*char*⟩$_{12}$is also allowed).

```
\VerbHyphen     \def\VerbHyphen#1{%
                  {\escapechar\m@ne
                    \expandafter\gdef\expandafter\verbhyphen\expandafter{\string#1}}}
```

As you see, it's always global.

The Breakables

Let's define a `\discretionary` left brace such that if it breaks, it turns `{%` at the end of line. We'll use it in almost Knuthian `\ttverbatim`—it's part of this 'almost'.

```
                \bgroup␣\catcode`\<=1␣\@makeother\{␣\catcode`\>=2␣%
                \firstofone<\egroup
\breaklbrace      \def\breaklbrace<\discretionary<{\verbhyphen><><{>>%
```

```
\twelvelbrace   20     \def\twelvelbrace<{>%
                21  >% of \firstofone
```

```
22 \bgroup␣\catcode`\<=1␣\catcode`\{=\active␣\catcode`\>=2
23 \firstofone<\egroup
24     \def\dobreaklbrace<\catcode`\{=\active␣\def{<\breaklbrace>>%
25 >% end of \firstofone.
```

The `\bslash` macro defined below I use also in more 'normal' TEXing, e.g., to `\typeout` some `\outer` macro's name.

```
                26  {\catcode`\!=0␣\@makeother\\%
\bslash         27  !gdef!bslash{\}%
\breakbslash    28  !gdef!breakbslash{!discretionary{!verbhyphen}{\}{\}}%
                29  }
```

Sometimes linebreaking at a backslash may be unwelcome. The basic case, when the first CS in a verbatim breaks at the lineend leaving there `%`, is covered by line 183. For the others let's give the user a countercrank:

```
\fixbslash      30  \newcommand*\fixbslash{\let\breakbslash=\bslash}% to use due to the common
                        scoping rules. But for the special case of a backslash opening a verbatim scope,
                        we deal specially in the line 183.
```

Analogously, let's provide a possibility of 'fixing' the left brace:

```
\fixlbrace      31  \newcommand*\fixlbrace{\let\breaklbrace=\twelvelbrace}
```

```
32 {\catcode`\!=0␣%
33 !catcode`!\=!active
34 !gdef!dobreakbslash{!catcode`!\=!active␣!def\{!breakbslash}}%
35 }
```

The macros defined below, `\visiblebreakspaces` and `\twelveclub` we'll use in the almost Knuthian macro making verbatim. This 'almost' makes a difference.

```
                36  \bgroup\catcode`\␣=12␣%
                37  \firstofone{\egroup%
\twelvespace    38  \def\twelvespace{␣}%
                39  \def\breakabletwelvespace{\discretionary{␣}{}{␣}}}
```

```
                40  \bgroup\obeyspaces% it's just re\catcode'ing.
                41  \firstofone{\egroup%
\activespace    42  \newcommand*\activespace{␣}%
                43  \newcommand*\dobreakvisiblespace{\let␣=\breakabletwelvespace\obeyspaces}%
                        %\defing it caused a stack overflow disaster with gmdoc.
                44  \newcommand*\dobreakblankspace{\let␣=\space\obeyspaces}%
                45  }
```

```
                46  \bgroup\@makeother\|
\twelveclub     47  \firstofone{\egroup\def\twelveclub{|}}
```

Almost-Knuthian `\ttverbatim`

`\ttverbatim` comes from The TEXbook too, but I add into it a LATEX macro changing the `\catcode`s and make spaces visible and breakable and left braces too.

```
\ttverbatim     48  \newcommand*\ttverbatim{%
                49    \let\do=\do@noligs␣\verbatim@nolig@list
```

```
  \let\do=\@makeother␣\dospecials
  \dobreaklbrace\dobreakbslash
  \dobreakspace
  \tt}
```

We wish the visible spaces to be the default.

```
\let\dobreakspace=\dobreakvisiblespace
```

The Core: From shortvrb

The below is copied verbatim ;-) from doc.pdf and then is added my slight changes.

\MakeShortVerb*
\MakeShortVerb

```
\def\MakeShortVerb{%
  \@ifstar
  {\def\@shortvrbdef{\verb*}\@MakeShortVerb}%
  {\def\@shortvrbdef{\verb}\@MakeShortVerb}}
```

\@MakeShortVerb

```
\def\@MakeShortVerb#1{%
  \expandafter\ifx\csname␣cc\string#1\endcsname\relax
  \@shortvrbinfo{Made␣}{#1}\@shortvrbdef
  \add@special{#1}%
  \AddtoPrivateOthers#1% a macro to be really defined in gmdoc.
  \expandafter
  \xdef\csname␣cc\string#1\endcsname{\the\catcode`#1}%
  \begingroup
  \catcode`\~\active␣\lccode`\~`#1%
  \lowercase{%
    \global\expandafter\let
    \csname␣ac\string#1\endcsname~%
    \expandafter\gdef\expandafter~\expandafter{%
      \expandafter\ifmmode\expandafter\string\expandafter~%
      \expandafter\else\expandafter\afterfi\@shortvrbdef~\fi}}% This terrible number of \expandafters is to make the shortverb char just other in the math mode (my addition).
  \endgroup
  \global\catcode`#1\active
 \else
 \@shortvrbinfo\@empty{#1␣already}{\@empty\verb(*)}%
 \fi}
```

\DeleteShortVerb

```
\def\DeleteShortVerb#1{%
  \expandafter\ifx\csname␣cc\string#1\endcsname\relax
  \@shortvrbinfo\@empty{#1␣not}{\@empty\verb(*)}%
  \else
  \@shortvrbinfo{Deleted␣}{#1␣as}{\@empty\verb(*)}%
  \rem@special{#1}%
  \global\catcode`#1\csname␣cc\string#1\endcsname
  \global␣\expandafter\let␣\csname␣cc\string#1\endcsname␣\relax
  \ifnum\catcode`#1=\active
  \begingroup
  \catcode`\~\active␣\lccode`\~`#1%
  \lowercase{%
    \global\expandafter\let\expandafter~%
    \csname␣ac\string#1\endcsname}%
  \endgroup␣\fi␣\fi}
```

My little addition

```
\@ifpackageloaded{gmdoc}{%
  \def\gmv@packname{gmdoc}}{%
  \def\gmv@packname{gmverb}}
```

\@shortvrbinfo

```
\def\@shortvrbinfo#1#2#3{%
  \PackageInfo{\gmv@packname}{%
   ^^J\@empty␣#1\expandafter\@gobble\string#2␣a␣short␣reference
    for␣\expandafter\string#3}}
```

\add@special

```
\def\add@special#1{%
  \rem@special{#1}%
  \expandafter\gdef\expandafter\dospecials\expandafter
  {\dospecials␣\do␣#1}%
  \expandafter\gdef\expandafter\@sanitize\expandafter
  {\@sanitize␣\@makeother␣#1}}
```

For the commentary on the below macro see the doc package's documentation. Here let's only say it's just amazing: so tricky and wicked use of `\do`. The internal macro `\rem@special` defines `\do` to expand to nothing if the `\do`'s argument is the one to be removed and to unexpandable CSs `\do` and ⟨*`\do`'s argument*⟩ otherwise. With `\do` defined this way the entire list is just globally expanded itself. Analogous hack is done to the `\@sanitize` list.

\rem@special

```
\def\rem@special#1{%
  \def\do##1{%
    \ifnum`#1=`##1␣\else␣\noexpand\do\noexpand##1\fi}%
  \xdef\dospecials{\dospecials}%
  \begingroup
  \def\@makeother##1{%
    \ifnum`#1=`##1␣\else␣\noexpand\@makeother\noexpand##1\fi}%
  \xdef\@sanitize{\@sanitize}%
  \endgroup}
```

And now the definition of `verbatim` itself. As you'll see (I hope), the internal macros of it look for the name of the current environment (i.e., `\@currenvir`'s meaning) to set their expectation of the environment's `\end` properly. This is done to allow the user to define his/her own environments with `\verbatim` inside them. I.e., as with the verbatim package, you may write `\verbatim` in the begdef of your environment and then necessarily `\endverbatim` in its enddef. Of course (or maybe surprisingly), the commands written in the begdef after `\verbatim` will also be executed at `\begin{`⟨*environment*⟩`}`.

verbatim
\verbatim

```
\def\verbatim{\@beginparpenalty␣\predisplaypenalty␣\@verbatim
  \frenchspacing␣\gmobeyspaces␣\@xverbatim}% in the LaTeX version there's %%
          \@vobeyspaces instead of \gmobeyspaces.
```

verbatim*

```
\@namedef{verbatim*}{\@beginparpenalty␣\predisplaypenalty␣\@verbatim
  \@sxverbatim}
```

\endverbatim

```
\def\endverbatim{\@@par
  \ifdim\lastskip␣>\z@
    \@tempskipa\lastskip␣\vskip␣-\lastskip
    \advance\@tempskipa\parskip␣\advance\@tempskipa␣-\@outerparskip
    \vskip\@tempskipa
  \fi
  \addvspace\@topsepadd
  \@endparenv}
```

```
\expandafter\let\csname␣endverbatim*\endcsname␣=\endverbatim
```

\@xverbatim

```
\begingroup␣\catcode␣`!=0␣%
\catcode␣`[=␣1␣\catcode`]=2␣%
\catcode`\{=\active
\@makeother\}%
\catcode`\\=\active%
!gdef!@xverbatim[%
  !edef!verbatim@edef[%
    !def!noexpand!verbatim@end%
      ####1!noexpand\end!noexpand{!@currenvir}[%
      ####1!noexpand!end[!@currenvir]]]%
  !verbatim@edef
  !verbatim@end]%
!endgroup
```

\@sxverbatim

```
\let\@sxverbatim=\@xverbatim
```

F. Mittelbach says the below is copied almost verbatim from LaTeX source, modulo `\check@percent`.

\@verbatim

```
\def\@verbatim{%
```

Originally here was just `\trivlist␣\item[]`, but it worked badly in my document(s), so let's take just highlights of if.

```
  \parsep\parskip
```

From `\@trivlist`:

```
  \if@noskipsec␣\leavevmode␣\fi
  \@topsepadd␣\topsep
  \ifvmode
    \advance\@topsepadd␣\partopsep
  \else
    \unskip␣\par
  \fi
  \@topsep␣\@topsepadd
  \advance\@topsep␣\parskip
  \@outerparskip␣\parskip
```

(End of `\trivlistlist` and `\@trivlist` highlights.)

```
  \@@par\addvspace\@topsep
  \if@minipage\else\vskip\parskip\fi
  \leftmargin\parindent% please notify me if it's a bad idea.
  \advance\@totalleftmargin\leftmargin
  \raggedright
  \leftskip\@totalleftmargin% so many assignments to preserve the list thinking for possible future changes. However, we may be sure no internal list shall use \@totalleftmargin as far as no inner environments are possible in verbatim(*).
  \@@par% most probably redundant.
  \@tempswafalse
  \def\par{% but I don't want the terribly ugly empty lines when a blank line is met.
        Let's make them gmdoc-like i.e., let a vertical space be added as in between
        stanzas of poetry. Originally \if@tempswa\hbox{}\fi, in my version will
        be
    \ifvmode\if@tempswa\addvspace\stanzaskip\@tempswafalse\fi\fi
    \@@par
    \penalty\interlinepenalty␣\check@percent}%
  \everypar{\@tempswatrue\hangindent\verbatimhangindent\hangafter\@ne}%
        since several chars are breakable, there's a possibility of breaking some lines.
        We wish them to be hanging indented.
  \obeylines
  \ttverbatim}
```

```
\@ifundefined{stanzaskip}{\newlength\stanzaskip}{}
\stanzaskip=\medskipamount
```

\verbatimhangindent

```
\newlength\verbatimhangindent
\verbatimhangindent=3em
```

```
\providecommand*\check@percent{}
```

In the gmdoc package shall it be defined to check if the next line begins with a comment char.

Similarly, the next macro shall in gmdoc be defined to update a list useful to that package. For now let it just gobble its argument.

```
\providecommand*\AddtoPrivateOthers[1]{}
```

Both of the above are `\provide`d to allow the user to load gmverb after gmdoc (which would be redundant since gmdoc loads this package on its own, but anyway should be harmless).

Let's define the 'short' verbatim command.

\verb*
\verb

```
\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
  \bgroup
  \ttverbatim
  \gm@verb@eol
  \@ifstar{\@sverb@chbsl}{\gmobeyspaces\frenchspacing\@sverb@chbsl}}% in
        the LaTeX version there's \@vobeyspaces instead of \gmobeyspaces.
\def\@sverb@chbsl#1{\@sverb#1\check@bslash}
```

```
\def\@def@breakbslash{\breakbslash}% because \ is \defined as \breakbslash
      not \let.
```

For the special case of a backslash opening a (short) verbatim, in which it shouldn't be breakable, we define the checking macro.

\check@bslash

```
\def\check@bslash{\@ifnextchar{\@def@breakbslash}{\bslash\@gobble}{}}
```

```
\let\verb@balance@group\@empty
```

\verb@egroup

```
\def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}
```

\gm@verb@eol

```
\let\gm@verb@eol\verb@eol@error
```

The latter is a LaTeX 2$_\varepsilon$ kernel macro that `\active`ates line end and defines it to close the verb group and to issue an error message. We use a separate CS 'cause we are not quite positive to the forbidden line ends idea. (Although the allowed line ends with a forgotten closing shortverb char caused funny disasters at my work a few times.) Another reason is that gmdoc wishes to redefine it for its own queer purpose.

However, let's leave my former 'permissive' definition under the \verb@eol name.

```
                    \begingroup
                    \obeylines\obeyspaces%
\verb@eolOK         \gdef\verb@eolOK{\obeylines%
                    \def^^M{␣\check@percent}%
                    }%
                    \endgroup
```

The \check@percent macro here is \provided to be \@empty but in gmdoc employed shall it be.

Let us leave (give?) a user freedom of choice:

```
\verbeolOK          \def\verbeolOK{\let\gm@verb@eol\verb@eolOK}
```

And back to the main matter,

```
                    \def\@sverb#1{%
                      \catcode`#1\active␣\lccode`\~`#1%
                      \gdef\verb@balance@group{\verb@egroup
                        \@latex@error{Illegal␣use␣of␣\bslash␣verb␣command}\@ehc}%
                      \aftergroup\verb@balance@group
                      \lowercase{\let~\verb@egroup}}
\verbatim@nolig@list \def\verbatim@nolig@list{\do\`\do\<\do\>\do\,\do\'\do\-}
\do@noligs          \def\do@noligs#1{%
                      \catcode`#1\active
                      \begingroup
                      \lccode`\~=`#1\relax
                      \lowercase{\endgroup\def~{\leavevmode\kern\z@\char`#1}}}
```

And finally, what I thought to be so smart and clever, now is just one of many possible uses of a general almost Rainer Schöpf's macro:

```
\dekclubs           \def\dekclubs{\MakeShortVerb*\|}
```

doc- And shortvrb-Compatibility

One of minor errors while TEXing doc.dtx was caused by my understanding of a 'shortverb' char: at my settings, in the math mode an active 'shortverb' char expands to itself's 'other' version thanks to \string. doc/shortvrb's concept is different, there a 'shortverb' char should work as usual in the math mode. So let it may be as they wish:

```
\old@MakeShortVerb  \def\old@MakeShortVerb#1{%
                      \expandafter\ifx\csname␣cc\string#1\endcsname\relax
                      \@shortvrbinfo{Made␣}{#1}\@shortvrbdef
                      \add@special{#1}%
                      \AddtoPrivateOthers#1% a macro to be really defined in gmdoc.
                      \expandafter
                      \xdef\csname␣cc\string#1\endcsname{\the\catcode`#1}%
                      \begingroup
                      \catcode`\~\active␣\lccode`\~`#1%
                      \lowercase{%
                        \global\expandafter\let
```

```
                         218      \csname␣ac\string#1\endcsname~%
                         219      \expandafter\gdef\expandafter~\expandafter{%
                         220        \@shortvrbdef~}}%
                         221    \endgroup
                         222    \global\catcode`#1\active
                         223   \else
                         224   \@shortvrbinfo\@empty{#1␣already}{\@empty\verb(*)}%
                         225   \fi}
   \OldMakeShortVerb     226 \def\OldMakeShortVerb{\begingroup
                         227   \let\@MakeShortVerb=\old@MakeShortVerb
                         228   \@ifstar{\eg@MakeShortVerbStar}{\eg@MakeShortVerb}}
\eg@MakeShortVerbStar    229 \def\eg@MakeShortVerbStar#1{\MakeShortVerb*#1\endgroup}
    \eg@MakeShortVerb    230 \def\eg@MakeShortVerb#1{\MakeShortVerb#1\endgroup}
                         231 \endinput% for the Tradition.
```

Index

Numbers written in italic refer to the code lines 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. The numbers preceded with 'p.' are page numbers. All the numbers are hyperlinks.