Grzegorz Murzynowski

The gmverb Package[*]

For documentation please refer to the file(s)
gmverb.{sty,pdf}.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{gmverb}
    [2011/10/14 v0.98 After shortvrb (FM) but my way (GM)]
```

⟨*master⟩

(A handful of meta-settings skipped)

⟨/master⟩

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 their 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|`.

---

[*] This file has version number dated .

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

`\nobreakbslash` `\nobreaklbrace`

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

`\VerbHyphen`

The default 'hyphen' is `%` since it's the default comment char. If you wish another char to appear at the line break, 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` `\dekclubs`

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 get `\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 TEX book*'s format, **gmverb** provides a macro `\dekclubs` as a shorthand for `\MakeShortVerb(*)\|`.

`\DeleteShortVerb`

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

One more difference between **gmverb** and **shortvrb** is that the chars `\activeated` by `\MakeShortVerb` in the math mode behave as if they were 'other', so you may type e.g., `$2|0$` to get $2|0$ and + `\activeated` this way is in the math mode typeset properly etc.

However, if you don't like such a conditional behaviour, you may specify some `$$$$`'s (any nonempty seQuence) mixed with star(s) if you wish as an optional argument to keep the short-verbatim char behave as short-verbatim also in math mode.

`\OldMakeShortVerb`

For compatibility with **gmdoc** and for convenience there is a shorthand for that provided, `\OldMakeShortVerb`.

`\dekclubs` `\dekclubs*` `\olddekclubs`

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

`\edverbs`

There's one more declaration, `\edverbs` that makes `\[` checks if the next token is an active char and opens an `\hbox` if so. That is done so that you can write (in `\edverbs`' and `\dekclubs`' scope)

```
\[|<verbatim stuff>|\]
```

instead of

```
\[\hbox{|<verbatim stuff>|}\]
```

to get a displayed shortverb.

Both versions of `\dekclubs` OCSR.

The `verbatim` environment inserts `\topsep` before and after itself, just as in standard version (as if it was a `list`).

`\VisSpacesGrey` `visspacesgrey`

In August 2008 Will Robertson suggested grey visible spaces for **gmdoc**. I added a respective option to **gmdoc** but I find them so nice that I want to make them available for all verbatim environments so I bring here the declaration `\VisSpacesGrey`. It redefines only the visible spaces so affects `\verb*` and `verbatim*` and not the unstarred versions. The colour of the visible spaces is named `visspacesgrey` and you can redefine it **xcolor** way.

`\verbatimspecials`

We also provide the `\verbatimspecials` declaration that takes six arguments:

`#1` `m` a char for verbatim escape char (for catcode 0), has to be unbraced[1],

`#2` `m` a char for group starter (for catcode 1), has to be unbraced,

`#3` `m` a char for group ender (for catcode 2), has to be unbraced,

`[#4]` (optional) a char for verbatim math shift (for catcode 3); it has to be in square brackets if present. If absent, nothing is set for the verbatim math shift,

`[#5]` (optional) a char for the shorthand for `\metachar`; it has to be in square brackets if present. If provided, e.g., `›` as I suggest in `\GMverbatimspecials`, then it itself becomes an active char let-equal to `\meta`, and a CS made of it, `\›` in this example, becomes `\string`.

`{#6}` `b` optional in curly braces, additional stuff (commands) to be executed in a verbatim.

All the specials defined this way, except the meta char, if preceded with the escape char, will be typeset verbatim.

For example, after telling TEX

```
\verbatimspecials ⁄ « » [ ¿ ] [ › ] {\def\|{$\vert$}}
```

(the slash is Unicode Fractional Slash, spaces are ignored) you can write

```
|\macro⁄arg«arg. ¿n+1¿»\›[No›]Value›(T⁄|F›)|
```

to get

```
\macro{⟨arg. n + 1⟩}\[No]Value(T|F)
```

Note also that `⁄|` is a control sequence so it doesn't delimit the short verbatim `|`'s argument.

The `\verbatimspecials` declaration OCSR. Subsequent uses of it override the previous settings. If you specified the optionals at first and then specify `\verbatimspecials` without optionals, the previous optional settings are forgotten.

`\noverbatimspecials`

To turn the 'verbatim specials' off write `\noverbatimspecials`, which OCSR too.

`\sups`
`\subs`

Note that although we don't provide a 'verbatim superscript' nor 'verbatim subscript', you have the `\sups` and `\subs` CS'es defined by **gmutils**.

`\(\)`

The 4th argument for the math shift is optional because you can use LATEX's `\(` and `\)`.

`alltt`
`semiverbatim`

The `\verbatimspecials` declaration goes a step further than LATEX's `alltt` and Til Tantau's **beamer**'s `semiverbatim` environments. To get their effect, declare

```
\verbatimspecials\{}
```

`\scanverb`

There is something for verbatims in arguments of commands: `\scanverb`[`*`]`{`*⟨text⟩*`}`. However there are some limitations: if `%` is the comment char (which is usual situation), then you cannot use `%` in *⟨text⟩*, or rather, `%` will act as comment char anyway. Moreover, spaces are ignored. This last limitation may be worked around if you declare `\verbatimspecials`, say `⁄` (fraction slash) as the escape char. Then you can use `⁄␣` to put a space which will be typeset blank in the unstarred version and visible with star.

`\verbDiscretionaryHyphen`

Not so long ago I started to use the 'broken bar' (U+00A6, ¦) character as a hyphen in hyperlinks, because it seems not to occur in hyperlinks at all unlike hyphen. I suggest the same char for verbatims, but if you don't like it, there's the `\verbDiscretionaryHyphen`declaration that takes two arguments. Broken bar is declared as

```
\verbDiscretionaryHyphen{"A6}{¦}
```

---

[1] To be precise, the arguments cannot be wrapped in curly braces because those are recatcoded to 'other'. But if you make some other pair of chars category 1 and 2 that are not on the `\dospecials` list, then you can wrap the arguments in those chars. But what for?

Since version 0.95 (August 2010) this package also provides the `\VerbatimPitch` declaration that modifies the `verbatim` environments and derivatives so that the environment contents (recatcoded, i.e. "sanitised") are wrapped in the macro `\VerbatimContents` and therefore available after environment's end (after `\endverbatim` to be precise, so also in the end-def of a derivative verbatim).

This may be useful for TEXnical examples: you can rescan the contents of a verbatim with `\scantokens` and execute/typeset it. Such a thing is done in the **gmdoc** package, see the `verbatim@p` environment.

The package options

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

Installation

Unpack the **\jobname-tds.zip** archive (this is an archive that conforms the TDS standard, see **CTAN/tds/tds.pdf**) in some texmf directory or just put the **gmutils.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 **gmutils** package consists of the following three files and a TDS-compliant archive.

gmverb.sty
README
gmverb.pdf
gmverb.tds.zip

Compiling of 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 **gmverb.sty** file twice (`xelatex gmverb.sty` in the directory you wish the documentation to be in, you don't have copy the **.\gmdExt** file there, TEX will find it), then MakeIndex on the **\jobname.idx** file, and then LATEX on **\jobname.\gmdExt** once more.

MakeIndex shell commands:

```
makeindex -r gmverb
makeindex -r -s gmglo.ist -o gmverb.gls gmverb.glo
```

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**, the **gmutils** bundle, **gmiflink.sty** and also some standard packages: **hyperref.sty**, **color.sty**, **geometry.sty**, **multicol.sty**, **lmodern.sty**, **fontenc.sty** that should be installed on your computer by default.

Moreover, you should put the **gmglo.ist** file, a MakeIndex style for the changes' history, into some **texmf/makeindex** (sub)directory.

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

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

```
466 \RequirePackage{gmcommand}[2011/10/12]
```

For `\firstofone`, `\afterfi`, `\gmobeyspaces`, `\@ifnextcat`, `\foone` and `\noexpand`'s and `\expandafter`'s shorthands `\@nx` and `\@xa` resp. and `\DeclareCommand`.

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   480 \let\verbhyphen\xiipercent
```

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

```
\VerbHyphen   486 \def\VerbHyphen#1{%
              487   {\escapechar\m@ne
              488     \@xa\gdef\@xa\verbhyphen\@xa{\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'.

```
\breaklbrace     496 \def\breaklbrace{%
                 497   \discretionary{\type@lbrace\verbhyphen}{}{\type@lbrace}%
                 498   \yeshy}
```

```
                 500 \foone{\catcode`\[=1 \catcode`\{=\active \catcode`\]=2 }%
                 501 [%
\dobreaklbrace   502    \def\dobreaklbrace[\catcode`\{=\active
                 503    \def{%
\breaklbrace     504      [\breaklbrace\gm@lbracehook]]%
                 505 ]
```

Now we only initialise the hook. Real use of it will be made in **gmdoc**.

```
509 \relaxen\gm@lbracehook
```

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

```
            514 \foone{\catcode`\!=0 \@makeother\\}%
            515 {%
\bslash     516   !def!bslash{\}%
            517   }% of \foone.
```

```
\breakbslash  520 \def\breakbslash{%
              521   \discretionary{\verbhyphen}%
              522   {\type@bslash}{\type@bslash}\yeshy% it seems that we allow hyphenation after
                        backslash but hyphenation will be allowed iff \hyphenchar\font is nonnega-
                        tive.
              525 }% of \breakbslash.
```

Sometimes line breaking at a backslash may be unwelcome. The basic case, when the first CS in a verbatim breaks at the line end leaving there `%`, is covered by line 1006. For the others let's give the user a counter-crank:

\nobreakbslash
\breakbslash

```
531 \pdef\nobreakbslash{\def\breakbslash{\type@bslash\yeshy}}% 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 1006.
```

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

\nobreaklbrace
\breaklbrace
\dobreakbslash
\breakbslash

```
538 \pdef\nobreaklbrace{\def\breaklbrace{\type@lbrace\yeshy}}

541 \foone{\catcode`\!=0 \catcode`\\=\active}%
542 {%
543   !def!dobreakbslash{!catcode`!\=!active !def\{!breakbslash}}%
544 }
```

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

\breakablevisspace

```
551 \def\breakablevisspace{\discretionary{\visiblespace}{}{\visiblespace}}
```

The `\visiblespace` macro is `\let` in **gmutils** to `\xiispace` or to `\xxt@visiblespace` of **xltxtra** if available.

\dobreakvisiblespace
\breakablevisspace
\dobreakblankspace
\xiiclub

```
555 \foone\obeyspaces% it's just re\catcode'ing.
556 {%
557 \newcommand*\dobreakvisiblespace{\def {%
        \breakablevisspace}\obeyspaces}% \defing it caused a stack overflow
        disaster with gmdoc.
559 \newcommand*\dobreakblankspace{\let =\space\obeyspaces}%
560 }

563 \foone{\@makeother\|}{%
564   \def\xiiclub{|}}
```

Almost-Knuthian `\ttverbatim`

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

\ttverbatim

```
573 \pdef\ttverbatim{%
574   \let\do=\do@noligs \verbatim@nolig@list
575   \let\do=\@makeother \dospecials
576   \dobreaklbrace\dobreakbslash
577   \dobreakspace
578   \makeatletter
582   \ifhmode
583     \setspaceskip
584   \fi
585   \verbatimfont
586   \xdef\gmv@storedhyphenchar{\the\hyphenchar\font}%
```

Assignment of the hyphenchar is always global so let the above edefinition be also such.

```
589   \hyphenchar\font=\gmv@hyphenchar
590   \ttverbatim@hook
591 }% of \ttverbatim
```

(2010/08/14, v0.993:) rigid `\tt` in `\ttverbatim` changed to redefinable `\verbatim¦font` due to absurd problems with bad fontifying of **gmdoc**

\verbatimfont

```
597 \def\verbatimfont{\tt}
```

While typesetting stuff in the QX fontencoding I noticed there were no spaces in verbatims. That was because the QX encoding doesn't have any reasonable char at position 32. So we provide a hook in the very core of the verbatim making macros to set proper fontencoding for instance.

```
\@emptify\ttverbatim@hook
```

\VerbT1
\VerbT
\ttverbatim@hook

```
\def\VerbT1{\def\ttverbatim@hook{\fontencoding{T1}\selectfont}}
```

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
\@shortvrbdef
\@shortvrbdef

```
\DeclareCommand \MakeShortVerb{
  #1  Q {*$} %
  #2  m % the char to be made shortverb. Has to be provided as \⟨char⟩ (backslashed).
}{%
  \gmu@ifstrany {*}{#1}
  {\def\@shortvrbdef{\verb*}}%
  {\def\@shortvrbdef{\verb}}
  \gmu@ifstrany {$}{#1}%
```

Depending on whether `$` was specified as optional argument to `\MakeShortVerb`, we define the CS `\gmV@mm\`*⟨the char⟩* as either sth. special or sth. usual in the math mode:

```
  {\gmV@MakeShortVerbAlsoInMath #2 }% extracted to a macro because we reassign it while typesetting LaTeX 2ε Sources.
  {\gmV@MakeShortVerbNotInMath  #2}%
  \@MakeShortVerb #2%
}% of \MakeShortVerb.
```

V@MakeShortVerbAlsoInMath

```
\pdef\gmV@MakeShortVerbAlsoInMath #1{% If some dollars were specified (paid ;-) ) in #1, then we direct the shortverb char to behave in the math mode the same as in normal text.
  \Name\glet {gmV@mm\string#1}\@firstofone
}%
```

mV@MakeShortVerbNotInMath

```
\pdef\gmV@MakeShortVerbNotInMath #1{% If no dollar was paid ;-) in #1, we get vicious and define shortverb to act in math mode as normal (other) char.
  \edef\gmV@tempA {\strip@bslash {#1}}%
  \@XA {%
    \Name\gdef {gmV@mm\string#1}}%
  \@xa{\@xa % these two expandafters serve to get “other” version of (unbackslashed) #1.
    \ifmmode \gmV@tempA
```

Two subsequent expandafters become part of the definition and serve to gobble or execute the short-verbatim macro.

```
      \@xa \@gobble
    \else
      \@xa \@firstofone
    \fi
  }%
```

```
}% of if to be normal in math mode
```

We store the original catcode of a shortverb char in a CS `\gmV@cc\`⟨*the char*⟩. Moreover, if the char is originally active (of catcode 13), then we create another CS to store its original meaning, `\gmV@ac\`⟨*the char*⟩.

\@MakeShortVerb

```
\def\@MakeShortVerb#1{%
  \@xa\ifx\csname gmV@cc\string#1\endcsname\relax
    \@shortvrbinfo{Made }{#1}\@shortvrbdef
    \add@special{#1}%
    \AddtoPrivateOthers#1% a macro to be really defined in gmdoc.
    \@xa
    \xdef\csname gmV@cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
    \catcode`\~\active \lccode`\~=`#1%
    \lowercase{%
      \Name\glet {ac\string#1}~%
      \@xa\gdef\@xa~\@xa{%
        \csname \strip@bslash{\gmV@mm}\string#1\endcsname
        {\@shortvrbdef~}%
      }% of active #1 def
    }% of lowercase
    \endgroup
    \global\catcode`#1\active
  \else
    \@shortvrbinfo\@empty{#1 already}{\@empty\verb(*)}%
  \fi
}% of \@MakeShortVerb.
```

\DeleteShortVerb

```
\def\DeleteShortVerb#1{%
  \@xa\ifx\csname gmV@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 gmV@cc\string#1\endcsname
    \global \@xa\let \csname gmV@cc\string#1\endcsname \relax
    \ifnum\catcode`#1=\active
      \begingroup
      \catcode`\~\active \lccode`\~`#1%
      \lowercase{%
        \global\@xa\let\@xa~%
        \csname gmV@ac\string#1\endcsname}%
      \endgroup
    \fi
  \fi
}
```

My little addition

\gmv@packname
\gmv@packname

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

\@shortvrbinfo

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

\add@special

```
731 \def\add@special#1{%
732   \rem@special{#1}%
733   \@xa\gdef\@xa\dospecials\@xa
734   {\dospecials \do #1}%
735   \@xa\gdef\@xa\@sanitize\@xa
736   {\@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 CSes `\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

```
747 \def\rem@special#1{%
748   \def\do##1{%
749     \ifnum`#1=`##1 \else \@nx\do\@nx##1\fi
750   }%
751   \xdef\dospecials{\dospecials}%
752   \begingroup
753   \def\@makeother##1{%
754     \ifnum`#1=`##1 \else \@nx\@makeother\@nx##1\fi}%
755   \xdef\@sanitize{\@sanitize}%
756   \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 begin definition of your environment and then necessarily `\endverbatim` in its end definition. Of course (or maybe *surprisingly*), the commands written in the begin definition after `\verbatim` will also be executed at `\begin{`*⟨environment⟩*`}`.

verbatim
\verbatim

```
769 \def\verbatim{%
770   \edef\gmv@hyphenpe{\the\hyphenpenalty}%
771   \edef\gmv@exhyphenpe{\the\exhyphenpenalty}%
772   \@beginparpenalty \predisplaypenalty \@verbatim
773   \frenchspacing \gmobeyspaces \@xverbatim
774   \hyphenpenalty=\gmv@hyphenpe\relax
775   \exhyphenpenalty=\gmv@exhyphenpe
776   \hyphenchar\font=\m@ne
```

The line below serves as the delimiter for `\verbatim@PitchContents`, to discard the stuff before it (see l. 850).

verbatim*

```
782   \@gobble\verbatim
783 }% in the LaTeX version there's \@vobeyspaces instead of \gmobeyspaces.
788 \@namedef{verbatim*}{\@beginparpenalty \predisplaypenalty \@verbatim
789   \@sxverbatim % it's the same as \@xverbatim and defines the verbatim end
          (a macro delimited with \end{⟨curr.envir.⟩}).
791 }
```

\endverbatim

```
793 \def\endverbatim{\@@par
```

`%%    \hyphenchar\font=\gmv@storedhyphenchar %` hyphenchar assignments are
`%%    %` always global. And for an entire paragraph works the one last in it so we hide it.

```
  \ifdim\lastskip >\z@
    \@tempskipa\lastskip \vskip -\lastskip
    \advance\@tempskipa\parskip \advance\@tempskipa -\@outerparskip
    \vskip\@tempskipa
  \fi
  \addvspace\@topsepadd
  \@endparenv
}
```

*

```
\n@melet{endverbatim*}{endverbatim}
```

```
\begingroup \catcode `!=0 %
\catcode `[= 1 \catcode`]=2 %
\catcode`\{=\active
\@makeother\}%
\catcode`\\=\active%
```

\@xverbatim

```
!gdef!@xverbatim[%
  [!endlinechar!m@ne !everyeof[!@nx]%
    !edef!verbatim@currenvir[%
      !@xa!scantokens!@xa[!@currenvir]%
    ]%
```

of `\verbatim@currenvir`. This macro is defined as the meaning of `\@currenvir` rescanned. It's done specially for the active star in my verbatims. `%` `\@currenvir` is fully expanded but my active star is `\protected`.

```
    !@xa]%
```

and here a little trick with groups:

```
    !@xa!def!@xa!verbatim@currenvir
  !@xa[!verbatim@currenvir]%
  !edef!verbatim@edef[%
  !def!@nx!verbatim@end%
  ####1!noexpand\end!@nx{%
    !@xa!unexpanded!@xa[!verbatim@currenvir]%
  }[%
    !@nx!verbatim@PitchContents ####1%
    !@nx!verbatim@PitchContents@Delim% added 2010/8/16
    ####1!@nx!end[!@currenvir]]]%
  !verbatim@edef
  !verbatim@end]%
!endgroup
```

\@sxverbatim

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

rbatim@PitchContents@Left
\verbatim@PitchContents

```
\def\verbatim@PitchContents@Left{%
  \long\def\verbatim@PitchContents
  ##1\@gobble\verbatim ##2\verbatim@PitchContents@Delim
}
```

By default we make `\verbatim@PitchContents` a gobbler.

```
\verbatim@PitchContents@Left {}
```

\VerbatimPitch

```
\pdef\VerbatimPitch{%
```

But in this declaration scope we make `\@xverbatim` pitch the contents of `verbatim`s in a macro. We use that in **gmdoc** not to repeat examples' code.

```
  \verbatim@PitchContents@Left {%
    \gdef\VerbatimContents{##2}%
  }%
}% of \VerbatimThrow.
```

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%
  \advance\@totalleftmargin\verbatimleftskip
  \parskip\verbatimparskip % added 2010/6/2
  \raggedright
  \leftskip\@totalleftmargin% so many assignments to preserve the list think-
         ing 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
  \verbatim@specials
```

```
}
```

\stanzaskip

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

\verbatimleftskip

```
\newskip\verbatimleftskip
\verbatimleftskip\leftmargini
```

\verbatimhangindent

```
\newskip\verbatimhangindent
\verbatimhangindent=3em
```

\verbatimparskip

```
\newskip\verbatimparskip
\verbatimparskip\z@skip
```

\check@percent

```
\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.

\AddtoPrivateOthers

```
\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
  \verbatim@specials
  \gm@verb@eol
  \gmu@ifstar
  {\verb@lasthook\@sverb@chbsl}%
  {\gmobeyspaces\frenchspacing\verb@lasthook\@sverb@chbsl}}% in the
        LaTeX version there's \@vobeyspaces instead of \gmobeyspaces.
```

```
\emptify\verb@lasthook
```

\@sverb@chbsl

```
\def\@sverb@chbsl#1{\@sverb#1\check@bslash}
```

\@def@breakbslash

```
\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.

\DefineTypeChar

```
\DeclareCommand\DefineTypeChar{mmo}{%
```

```
%  #1  m the char as a CS,
%  #2  m short name of the char.
% [#3] o the cs of the char in 'other' catcode.
```

```
  \@namedef{gmu@#2wd@name}{#2 wd of
    \@xa\meaning\the\font\space at \detoken@xa\f@size pt}
```

```
  \@namedef{gmu@measure#2}{%
    \unless\ifcsname\csname gmu@#2wd@name\endcsname\endcsname
    \gmu@measurewd{#1}% \edefs \gmu@tempa as the width of the char and \gmu@tempb
          as the width of the char among 20 copies of itself.
    \@xa\gn@melet\csname gmu@#2wd@name\endcsname{gmu@tempb}% here we let
          the CS with the name contained in \gmu@⟨char-name⟩wd@name to the expanded
          value of width of the char measured among copies of it.
    \fi
  }% of \gmu@measure⟨char-name⟩.

  \@nameedef{type@#2}{%
    \@nx\leavevmode
    \@xanxcs{gmu@measure#2}%
    \hbox to \@nx\csname
    \@xanxcs{gmu@#2wd@name}\@nx\endcsname
    {\gmuIfValueTF{#3}{\@nx#3}{\@xanxcs{#2}}%
      \@nx\hss}%
  }% of \type@⟨char-name⟩,
}% of \DefineTypeChar.

\DefineTypeChar\\{bslash}% this defines \type@bslash and its aides \gmu@measurebslash
      and \gmu@bslashwd@name.

\DefineTypeChar\{{lbrace}[\xiilbrace]% this defines \type@lbrace and its auxilia
      analogous to the above.
```

\check@bslash

```
\def\check@bslash{%
  \@ifnextchar\@def@breakbslash
  {\type@bslash\yeshy\@gobble}% note we allow hyphenation but actually this will
        have effect if \hyphenchar\font allows hyphenation (when it's not > 0 ).
  {}}

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

\verb@egroup

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

`%%   \hyphenchar\font=\gmv@storedhyphenchar %` `\hyphenchar` behaves like `\hyphenpenalty` etc.: the last one in a paragraph is in charge for entire.

```
  \egroup
}
```

\gm@verb@eol

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

The latter is a LaTeX 2$_\varepsilon$ kernel macro that `\activeates` 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.

\check@percent

```
\begingroup
\obeylines\obeyspaces%
\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}% here we make the delimiter to be the macro
        closing the verbatim group.
}
```

\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
\olddekclubs

```
\def\dekclubs{\gmu@ifstar{\MakeShortVerb*\|}{\MakeShortVerb\|}}
\def\olddekclubs{\OldMakeShortVerb\|}
```

But even if a shortverb is unconditional, the spaces in the math mode are not printed. So,

\edverbs

```
\newcommand*\edverbs{%
  \let\gmv@dismath\[%
  \let\gmv@edismath\]%
  \def\[{%
    \@ifnextac\gmv@disverb\gmv@dismath}%
  \relaxen\edverbs}%
```

\gmv@disverb

```
\def\gmv@disverb{%
  \gmv@dismath%
  \hbox\bgroup\def\]{\egroup\gmv@edismath}%
}
```

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:

\OldMakeShortVerb

```
\pdef\OldMakeShortVerb{\MakeShortVerb $$}%
```

Grey visible spaces

In August 2008 Will Robertson suggested grey spaces for **gmdoc**. I added a respective option to that package but I like the grey spaces so much that I want provide them for any verbatim environments, so I bring the definition here. The declaration, if put in the preamble, postpones redefinition of `\visiblespace` till `\begin{document}` to recognise possible redefinition of it when **xltxtra** is loaded.

```
\let\gmd@preambleABD\AtBeginDocument
\AtBeginDocument{\let\gmd@preambleABD\firstofone}

\RequirePackage{xcolor}% for \providecolor
```

\VisSpacesGrey

```
\def\VisSpacesGrey{%
  \providecolor{visspacesgrey}{gray}{0.5}%
  \gmd@preambleABD{%
    \edef\visiblespace{%
      \hbox{\@nx\textcolor{visspacesgrey}%
        {\@xa\unexpanded\@xa{\visiblespace}}}}%
  }}
```

Verbatim specials—CSes in verbatims

\verbatimspecials

`\pdef\verbatimspecials{%` This declaration only defines a bearer of the ‘verbatim specials’.

```
% #1  m char for verbatim escape char (for catcode 0), has to be unbraced,
% #2  m char for verbatim group begin (for catcode 1), has to be unbraced,
% #3  m char for verbatim group end (for catcode 2), has to be unbraced,
% [#4] o char for verbatim math shift (for catcode 3),
% [#5] o char for a shorthand for \metachar.
% {#6} b (optional braced) additional stuff (commands) to be executed at the beginning of the verbatims.
```

```
  \@bsphack
  \begingroup
  \let\do\@makeother
  \dospecials
  \catcode`\ =10
  \verbatim@specials@iii}
```

\verbatim@specials@iii

`\pdef\verbatim@specials@iii#1#2#3{%` as you see, we take only first three arguments in a despecialized group. It’s to avoid `\futurelet` of the optionals’ parser to touch (and thus spoil) subsequent token. Yes, we could handle the case of a space or single line end but handling the case of a backslash would be somewhat difficult.

\verbatim@specials@list

```
  \endgroup
  \def\verbatim@specials@list{#1#2#3}%
  \@ifnextchar[%
  {\begingroup\let\do\@makeother\dospecials
    \catcode`\ =9
    \verbatim@specials@iv}%
  {\addtomacro\verbatim@specials@list{\NoValue\NoValue}%
    \verbatim@specials@vi}%
}% of \verbatim@specials@iii.
```

\verbatim@specials@iv

```
\pdef\verbatim@specials@iv[#1]{%
  \endgroup
  \addtomacro\verbatim@specials@list{#1}%
  \@ifnextchar[%
  {\begingroup\let\do\@makeother\dospecials
    \catcode`\ =9
    \verbatim@specials@v}%
  {\addtomacro\verbatim@specials@list{\NoValue}%
  \verbatim@specials@vi}%
}% of \verbatim@specials@iv.
```

\verbatim@specials@v

```
\pdef\verbatim@specials@v[#1]{%
  \endgroup
  \addtomacro\verbatim@specials@list{#1}%
  \verbatim@specials@vi
}% of \verbatim@specials@v.
```

\verbatim@specials@vi

```
\DeclareCommand\verbatim@specials@vi\long{b}
{\addtomacro\verbatim@specials@list{{#1}}%
\@esphack}
```

\verbatim@specials

```
\def\verbatim@specials{% this is the macro that actually sets the chars given in
      % \verbatim@specials@list as the escape char, group begin and group end.
  \ifdefined\verbatim@specials@list
  \@xa\verbatim@specials@\verbatim@specials@list
  \fi
}% of \verbatim@specials.
```

\verbatim@specials@

```
\long\def\verbatim@specials@#1#2#3#4#5#6{%
  \catcode`#1=0
  \protected\@namedef{#1}{#1}%
  \catcode`#2=1
  \protected\@namedef{#2}{#2}%
  \catcode`#3=2
  \protected\@namedef{#3}{#3}%
  \edef\gmu@tempa{\the\endlinechar}%
  \endlinechar\m@ne % we have to suppress adding of a line end by \scantokens
        since it would turn into an active char ^^M and raise an error (which actually
        did happen).
  \scantokens{%
    #1let#1bgroup=#2%
    #1let#1egroup=#3%
    #1catcode#1backquote#1h=6#1relax%
    #1pdef#1<h1>#2#1meta#2h1#3#3%
    #1catcode#1backquote#1h=11#1relax%
  }%
  \endlinechar\gmu@tempa\relax
  \gmuIfValueT{#4}{%
    \catcode`#4=3
    \protected\@namedef{#4}{#4}}%
  \gmuIfValueT{#5}{%
    \begingroup
    \lccode`\~=`#5\lowercase{\endgroup\let~\metachar}%
    \protected\@namedef{#5}{#5}%
    \catcode`#5=\active
  }% of if value #5.
  \gmuPutIfValue{#6}%
}
```

\noverbatimspecials

```
\pdef\noverbatimspecials{\let\verbatim@specials@list\@undefined}
```

\GMverbatimspecials

```
\def\GMverbatimspecials{%
  \gmu@ifCSdefined {\XeTeXversion}%
  {\verbatimspecials
    ⁄% escape
    «»% \bgroup and \egroup
    [¿]% math shift
    [›]% meta-char
    {\def\|{\metachar{$\vert$}}%
      \makestarlow
      \relaxen\'% to provide a CS separator (space is not 10 in verbatims).
      \let\›\string
    }% of #6.
  }%
  {}%
}% of \GMverbatimspecials.
```

Partial \verb in arguments

Now command for partial verbatims in arguments of commands:

```
\let\gmu@tempa\all@stars
\@xa\addtomacro\@xa\gmu@tempa\@xa{\all@unders}
```

`\gmv@hashhalfing`
`\xiihash`

```
\foone{\catcode`#=\active}
{\def\gmv@hashhalfing{%
    \def#{\xiihash\@ifnextchar#\gobble{}}%
    \catcode`#\active}%
}
```

```
\foone{\@makeother\^^R}{%
  \@xa\DeclareCommand\@xa\scanverb\@xa{%
    \@xa Q\@xa{\gmu@tempa}>Pm}{%
          %  #1  Q{⋆_}
          %  #2  m the stuff to be rescanned and typeset verbatim. Note that % will
                be executed during first scan so at best will disappear.
```

Spaces are ignored (because of detokenizers that add a space after a CS) but if you declare some \verbatimspecials, then you can use ⟋ where ⟋ denotes the escape char in verbatim.

```
    \begingroup
    \gmu@septify
    \endlinechar=\m@ne
    \@xa\IfIntersect\@xa{\all@stars}{#1}%
    {\def\ {\breakablevisspace}}%
    {\let\ =\space}%
    \@xa\IfIntersect\@xa{\all@unders}{#1}%
    {}{% We make spaces ignored only if there was no underscore in #1 and if #2
          doesn't contain \ .
      \gmu@ifxany\ {#2}%
      {}{\addtomacro\verb@lasthook{\catcode`\ =9 }}}%
    \addtomacro\verb@lasthook{\gmv@hashhalfing }%
    \@makeother\^^R%
    \edef\gmu@tempa{%
      \@nx\scantokens{%
        \bslash verb%
        ^^R\detokenize{#2}^^R% we delimit the \verb's argument with 'other' ^^R
            assuming this char to be used very seldom if at all.
      }% of \scantokens,
    }\gmu@tempa
    \endgroup
```

```
  }% of \scanverb,
}% of \foone.
```

\verbDiscretionaryHyphen
\gmv@hyphenchar
\gmv@hyphen

```
\def\verbDiscretionaryHyphen#1#2{%
  \def\gmv@hyphenchar{\numexpr#1\relax}%
  \def\gmv@hyphen{#2}%
}
```

```
\verbDiscretionaryHyphen{"A6}{¦}
```

```
\ifgmuXeTeX
```

(2010/06/28, v0.94:) due to Will Robertson's remark that recatcoding long (no-ASCII) dashes works only under XeTeX and LuaTeX, I embrace them in a XeTeX conditional

\gmu@tempa
\verbLongDashes

```
\def\gmu@tempa{%
  \DeclareCommand\verbLongDashes{
    >iT{–}% to memorise which dash we set
    B{1.41} % expansion of en-dash
    >iT{—} % as above
    B{2} % expansion of em-dash
  }%
}
```

\gmu@tempb

```
\def\gmu@tempb{\catcode`–\active \catcode`—\active}
```

```
\foone{\catcode`–\active\catcode`—\active}
{%
  \edef\gmu@tempa{\@xau\gmu@tempa
    {%
      \@nx\addtomacro\@nx\ttverbatim@hook{%
        \@xau\gmu@tempb
        \def\@nx–{\@nx\scalebox{##1}[1]{\string–}}%
        \def\@nx—{\@nx\scalebox{##2}[1]{\string—}}%
      }%
    }%
  }%
}%
\gmu@tempa
```

```
\fi % of if XeTeX.
```

Note that we have two "hyphens": one for places where a line may be broken with a comment char and another, provided as \hyphenchar, for discretionary hyphens at points where correct TeX code cannot be broken, such as CS names.

```
\endinput
```

End of file 'gmverb.sty'.

□

Change History

gmverb

General:

CheckSum 1040 , 83

CheckSum 997 , 83

gmverb v0.79

\edverbs:

    - added, 1068

gmverb v0.80
- `\edverbs`:
    - debugged, i.e. `\hbox` added back and redefinition of `\[`, 1068
- `\xiiclub`:
    - `\ttverbatim@hook` added, 564

gmverb v0.81
- General:
    - `\afterfi` made two-argument (first undelimited, the stuff to be put after `\fi`, and the other, delimited with `\fi`, to be discarded, 1331

gmverb v0.82
- General:
    - CheckSum 663 , 83

gmverb v0.83
- General:
    - added a hook in the active left brace definition intended for **gmdoc** automatic detection of definitions (in line 504), 1331
    - CheckSum 666 , 83

gmverb v0.84
- General:
    - CheckSum 658 , 83

gmverb v0.85
- General:
    - added restoring of `\hyphenpenalty` and `\exhyphenpenalty` and setting `\hyphenchar=-1`, 1331
    - CheckSum 673 , 83

gmverb v0.87
- General:
    - CheckSum 661 , 83
    - visible space tidied and taken from **xltxtra** if available. **gmutils** required. The `\xii…` CSes moved to **gmutils**. The documentation driver moved into the **.sty** file, 1331

gmverb v0.88
- General:
    - CheckSum 682 , 83
- `\VisSpacesGrey`:
    - added, or rather moved here from **gmdoc**, 1116

gmverb v0.89
- General:
    - `\dekclubs`, `\dekclubs*` and `\olddekclubs` made more consistent, shorthands for `\MakeShortVerb\|`, `\MakeShortVerb*\|` and `\OldMakeShortVerb\|` respectively., 1331
    - CheckSum 686 , 83

gmverb v0.90
- General:
    - CheckSum 684 , 83
    - some `\(b|e)group` changed to `\(begin|end)group`, 1331

gmverb v0.91
- General:
    - CheckSum 686 , 83
    - put to CTAN on 2008/11/21, 83
- `\verbatimleftskip`:
    - added, 913

gmverb v0.92
- General:
    - CheckSum 979 because of `\verbatimspecials`, hyphenation in verbatims, low star in verbatims, kerning of backslash in shrunk fonts, 83
- `\breakbslash`:
    - renamed from `\fixbslash`, 531
- `\breaklbrace`:
    - renamed from `\fixlbrace`, 538
- `\ttverbatim`:
    - added `\makeatletter` to sound with the ‘verbatim specials’, namely to allow control sequences containing `@`, 578

gmverb v0.93
- General:
    - CheckSum 1035 because of a bug fix in `\scanverb`(halfing the hashes), 83
    - put to CTAN on 2010/03/04, 83
- `\gmv@hashhalfing`:
    - cut out as separate macro, 1249
- `\verbDiscretionaryHyphen`:
    - added to synchronise hyphen chars in **gmdoc**’s documentation, 1292
- `\xiihash`:
    - mandatory argument made long (a bug fix), 1256

gmverb v0.94
- General:
    - CheckSum 979 because of wrapping the UTF-8 dashes’ setting in a XƎTEX’s conditional, 83
    - put to CTAN on 2010/07/07, 83
- `\gmv@hyphen`:
    - due to Will Robertson’s remark that recatcoding long (no-ASCII) dashes works only under XƎTEX and LuaTEX, I embrace them in a XƎTEX conditional, 1301

gmverb v0.95
- General:
    - put to CTAN on 2011/02/02, 83

gmverb v0.98
- General:
    - put to CTAN on 2011/10/14, 83

gmverb v0.993
- `\ttverbatim`:
    - rigid `\tt` in `\ttverbatim` changed to redefinable `\verbatimfont` due to absurd problems with bad fontifying of **gmdoc**, 591

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.