The newverbs Package

Martin Scharrer
martin@scharrer-online.de

CTAN: http://www.ctan.org/pkg/newverbs

Version v1.3a – 2012/05/08

Abstract

This package allows the definition of `\verb` variants which add TeX code before and after the verbatim text. When used together with the `shortvrb` package it allows the definition of short verbatim characters which use this variants instead of the normal `\verb`.

1 Usage

1.1 Defining new variants of \verb

```
\newverbcommand{⟨\macro⟩}{⟨code before⟩}{⟨code after⟩}
\renewverbcommand{⟨\macro⟩}{⟨code before⟩}{⟨code after⟩}
\provideverbcommand{⟨\macro⟩}{⟨code before⟩}{⟨code after⟩}
```

This macros allow the definition of `\verb` variants. The verbatim content is processed using `\verb`, but the variants can add TeX code before and after it. The three definition macros use `\newcommand*`, `\renewcommand*` and `\providecommand*` internally to define *⟨macro⟩*, respectively. Afterwards *⟨macro⟩* can be used like `\verb`. The star version of *⟨macro⟩* will use `\verb*`.

See the implementation of `\qverb` in section 3.2 for an example.

1.2 Provided \verb variants

Two `\verb` variants are provided (i.e. with `\provideverbcommand`) by default.

```
\qverb⟨char⟩⟨verbatim material⟩⟨char⟩
```

This macro adds quote characters around the verbatim material. Two macros are used to insert the quotes: `\qverbbeginquote` (‘) and `\qverbendquote` (’). They can be redefined by the user if required. If the `csquotes` package was loaded beforehand the above macros use its macros `\openinnerquote` and `\closeinnerquote` to take advantage of the language dependent quotation marks. See the manual of `csquotes` for more details.

Using `\qverb`*⟨char⟩⟨verbatim material⟩⟨char⟩* is equal to `\qverbbeginquote\verb`*⟨char⟩⟨verbatim material⟩⟨char⟩*`\qverbendquote`, or

‘\verb⟨*char*⟩⟨*verbatim material*⟩⟨*char*⟩’ when the default definition of the quote macros is used.

```
\fverb⟨char⟩⟨verbatim material⟩⟨char⟩
```

This macro adds a frame (`\fbox{}`) around the verbatim text (`\fverb+$&^_%$+` → $\boxed{\texttt{\$\&\^{}\_\$}}$). A TEX box is used to store the content first, then the box is framed. The user can define similar command using the following code:

```
      \newverbcommand{\myverb}{\begin{lrbox}{\verbbox}}
                        {\end{lrbox}\mycommand{\usebox{\verbbox}}}
```

The temporary box `\verbbox` is only provided inside a `\...verbcommand`.

1.3 Using `\verb` variants with short verbatim character

```
\MakeSpecialShortVerb{⟨\macro⟩}{\⟨char⟩}
\MakeSpecialShortVerb*{⟨\macro⟩}{\⟨char⟩}
```

This package also defines a special version of the `\MakeShortVerb` macro from the `shortvrb` package. The orignal command `\MakeShortVerb*{\`⟨*char*⟩`}` changes the meaning of ⟨*char*⟩ so that ⟨*char*⟩⟨*verbatim material*⟩⟨*char*⟩ is a shorter alternative to `\verb*`⟨*char*⟩⟨*verbatim material*⟩⟨*char*⟩.

The new macro `\MakeSpecialShortVerb*{\`⟨*verb variant*⟩`}{\`⟨*char*⟩`}` does the same, but instead of `\verb*` it uses a `\`⟨*verb variant*⟩`*` which needs to be defined using `\newverbcommand`. The package `shortvrb` must be loaded by the user in order to make this macro work. It is not loaded automatically by `newverbs`.

The special meaning of ⟨*char*⟩ can be removed using `shortvrb`’s `\DeleteShortVerb`, i.e. the same way as for characters defined with the normal `\MakeShortVerb`. If a character was already made a short verbatim character it must be “deleted” before it can be redefined by `\MakeShortVerb` or `\MakeSpecialShortVerb`.

Examples:

`\MakeSpecialShortVerb{\qverb}{\"}` will make ‘`"`’ a short, quoting verbatim character: `"$^&$"` → ‘`$^&$`’.

`\DeleteShortVerb{\"}\MakeSpecialShortVerb{\fverb}{\"}` will change it definition to use `\fverb`: $\boxed{\texttt{\$\^{}\&\$}}$.

```
\collectverb{⟨code⟩}⟨char⟩⟨verbatim material⟩⟨char⟩
\collectverb*{⟨code⟩}⟨char⟩⟨verbatim material⟩⟨char⟩
\collectverb{⟨code⟩}{⟨verbatim material⟩}
\collectverb*{⟨code⟩}{⟨verbatim material⟩}
```

This macro is supposed to be used with its `{`⟨*code*⟩`}` argument at the end of user or package macro which want to typeset verbatim material. It will collect everything between the following ⟨*char*⟩ and its next occurrence as verbatim material. An exception is if the following ⟨*char*⟩ is ‘`{`’, then ‘`}`’ is taken as the end ⟨*char*⟩ to simulate a normal argument to increase user friendliness. Afterwards ⟨*code*⟩ is expanded with `{`⟨*verbatim material*⟩`}` direct behind it. The macro ensures proper font settings to typeset the verbatim material. For this, a group is opened before the material is collected and closed directly after the given code is processed. Therefore all changes done by the ⟨*code*⟩ are local and the material should be typeset directly. (In special cases when the group is disruptive, ⟨*code*⟩ can be a macro which reads both the verbatim material and the `\endgroup` as two arguments. However, then special care must be taken to use the correct font and some of the special characters may be active but have lost their definition.) The starred version will make spaces appear as '␣' instead of displaying them as normal spaces.

```
\Collectverb{⟨code⟩}⟨char⟩⟨verbatim material⟩⟨char⟩
\Collectverb*{⟨code⟩}⟨char⟩⟨verbatim material⟩⟨char⟩
\Collectverb{⟨code⟩}{⟨verbatim material⟩}
\Collectverb*{⟨code⟩}{⟨verbatim material⟩}
```

This macro is supposed to be used with its `{`⟨*code*⟩`}` argument at the end of user or package macro which want to collect plain verbatim material suitable to be written in auxiliary files or log messages. It will collect everything between the following ⟨*char*⟩ and its next occurrence as verbatim material without adjusting the font or defining any characters in a special way (besides being verbatim). The starred version will make spaces appear as '␣' when typeset but still be written to auxiliary files as normal spaces. An exception is if the following ⟨*char*⟩ is '`{`', then '`}`' is taken as the end ⟨*char*⟩ to simulate a normal argument to increase user friendliness. Afterwards ⟨*code*⟩ is expanded with `{`⟨*verbatim material*⟩`}` direct behind it. This macro does not add any group around the code. Should the code be typeset after all a proper font (e.g. `\ttfamily` or `\verbatim@font`) must be enabled manually.

```
\Verbdef⟨\macro⟩⟨char⟩⟨verbatim material⟩⟨char⟩
\Verbdef*⟨\macro⟩⟨char⟩⟨verbatim material⟩⟨char⟩
\Verbdef⟨\macro⟩{⟨verbatim material⟩}
\Verbdef*⟨\macro⟩{⟨verbatim material⟩}
```

This macro uses `\Collectverb` internally to define ⟨*\macro*⟩ as the plain ⟨*verbatim material*⟩. This can be used to define macros for special characters, so these can be used in error or warning messages or be written into auxiliary files. A corresponding `\verbdef` macro which defines macros with typesetable verbatim material is already provided by the `verbdef` package.

2 Compatibility with other verbatim packages

The compatibility with other verbatim packages is not tested yet. This package relies on the normal internal definition of `\verb` and `\MakeShortVerb`. Any package which changes these might break this package. Users which encounter incompatibilities should not hesitate to contact the package author (with details!).

Since v1.2 from 2011/02/16 the new verbatim macros and their short versions can be used inside `tabularx` environments. This package patches an internal macro of `tabularx` to achieve this compatibility.

3 Implementation

```
%<!COPYRIGHT>
\ProvidesPackage{newverbs}[%
%<!DATE>
%<!VERSION>
%<*DRIVER>
    2099/01/01 develop
%</DRIVER>
    Define new 'verb' commands and short verb. characters]
```

3.1 Verb Definition Commands

\newverbcommand

\renewverbcommand

\provideverbcommand

This macro calls the real macro with the to be used definition macro.

```
\newcommand*\newverbcommand{\new@verbcommand\newcommand}
\newcommand*\renewverbcommand{\new@verbcommand\renewcommand}
\newcommand*\provideverbcommand{\new@verbcommand\providecommand}
```

\new@verbcommand

#1: underlying definition macro
#2: macro to define
#3: code before
#4: code after

The trailing code is inserted by patching `\verb@egroup` which is called by `\verb` after the verbatim content.

```
\def\new@verbcommand#1#2#3#4{%
  #1*#2{%
    \begingroup
    \newverbcommand@settings
    \def\verb@egroup{\verb@orig@egroup#4\endgroup}%
    \def\newverbs@txend{#4\endgroup}%
    #3\verb
  }%
}
```

`\newverbs@tabularxsupport`

Enables support for the new verbatim macros inside `tabularx` environments. This environment defines its own almost-verbatim form of `\verb`which lacks the end-macro we patch above. The following code inserts such an end-macro.

```
\def\newverbs@tabularxsupport{%
    \begingroup
    \def\origa@TX@vb##1{\def\@tempa####1##1{\toks@{####1}\edef\@tempa{\the\toks@}%
        \expandafter\TX@v\meaning\@tempa\\ \\\ifnum0=`{\fi}}\@tempa!}
    \def\origb@TX@vb##1{\def\@tempa####1##1{\toks@{####1}\edef\@tempa{\the\toks@}%
        \expandafter\TX@v\meaning\@tempa\\\\\ifnum0=`{\fi}}\@tempa!}%
    \ifcase0%
        \ifx\TX@vb\origa@TX@vb 1\else
        \ifx\TX@vb\origb@TX@vb 1\fi\fi
    \relax
        \endgroup
        \PackageWarning{newverbs}{Couldn't patch 'TX@vb' macro of the 'tabularx' package. Definition unknown.}%
    \else
        \endgroup
        \PackageInfo{newverbs}{Patching 'TX@vb' macro of the 'tabularx' package.}%
        \def\TX@vb##1{\def\@tempa####1##1{\toks@{####1}\edef\@tempa{\the\toks@}%
            \expandafter\TX@v\meaning\@tempa\\\\\ifnum0=`{\fi}\newverbs@txend}\@tempa!}%
    \fi
    \let\newverbs@tabularxsupport\relax
}
```

The end-macro is initially empty and is set for every call of a new verb macro.

```
\def\newverbs@txend{}
```

The support is activated either now or at the begin of the document if the `tabularx` is loaded.

```
\@ifpackageloaded{tabularx}{%
    \newverbs@tabularxsupport
}{%
    \AtBeginDocument{\@ifpackageloaded{tabularx}{\newverbs@tabularxsupport}{}}%
}
```

`\newverbcommand@settings`

Some settings required for all new `\verb`-like commands. The original end group macro from `\verb` is saved away. Also the ‘temp box a’ is provided with a user friendly name.

```
\def\newverbcommand@settings{%
  \let\verb@orig@egroup\verb@egroup
  \let\verbbox\@tempboxa
}
```

3.2 Provided New Verb Commands

`\qverb`

Quoting version of `\verb`. Places a quote character before and after the verbatim content: ‘`verb`’.

```
\provideverbcommand{\qverb}{\qverbbeginquote}{\qverbendquote}
```

`\qverbbeginquote`

`\qverbendquote`

This macros insert the actual quotes. They can be redefined by the user to contain the required quotes. If available the quoting macros of `csquotes` are used.

```
\@ifundefined{openinnerquote}{%
  \def\qverbbeginquote{`}%
  \def\qverbendquote{'}%
}{%
  \def\qverbbeginquote{\openinnerquote}%
  \def\qverbendquote{\closeinnerquote}%
}
```

`\fverb`

A framed version of `\verb`. Stores the verbatim content first into a box. Then the box content is framed.

```
\newverbcommand{\fverb}
  {\setbox\verbbox\hbox\bgroup\color@setgroup}
  {\color@endgroup\egroup\fbox{\box\verbbox}}
```

3.3 Make Special Short Verbatim Characters

`\MakeSpecialShortVerb`

#1: verbatim macro
Uses the definition of `\MakeShortVerb` from `shortvrb` except with `\verb` replaced with the first argument. The second argument is then read by `\@MakeShortVerb`.

```
\newcommand*\MakeSpecialShortVerb[1]{%
  \@ifstar
    {\def\@shortvrbdef{#1*}\@MakeShortVerb}%
    {\def\@shortvrbdef{#1}\@MakeShortVerb}%
}
```

3.4 Collect verbatim argument

`\collectverb`

Collects verbatim text which can be typeset. Checks for an existing star.

```
\newcommand*\collectverb{%
    \begingroup
    \verbatim@font
    \@ifstar
        \@scollectverb
        \@collectverb
}
```

`\@collectverb`

#1: <code>
Changes catcodes and ensures that spaces are displayed normally.

```
\def\@collectverb#1{%
    \verb@eol@error
    \let\do\@makeother
    \dospecials
    \@vobeyspaces
    \frenchspacing
    \@noligs
    \@@collectverb{#1}%
}
```

`\@scollectverb`

#1: <code>
Changes catcodes.

```
\def\@scollectverb#1{%
    \verb@eol@error
    \let\do\@makeother
    \dospecials
    \@noligs
    \@@collectverb{#1}%
}
```

\@@collectverb

#1: <code>
#2: <char>

Defines \@@@collectverb to read everything to the next occurrence of ⟨*char*⟩ and feed it to the given ⟨*code*⟩.

```
\def\@@collectverb#1#2{%
    \ifnum`#2=`\{%
        \catcode`\}\active
    \else
        \catcode`#2\active
    \fi
    \begingroup
    \ifnum`#2=`\{%
        \lccode`\~`\}%
    \else
        \lccode`\~`#2%
    \fi
    \lowercase{\endgroup
        \def\@@@collectverb##1~}{#1{##1}\endgroup}%
    \@@@collectverb
}
```

\collectverbenv

Collects verbatim text which can be typeset. Checks for an existing star.

```
\newcommand*\collectverbenv{%
    \begingroup
    \verbatim@font
    \@ifstar
        \@scollectverbenv
        \@collectverbenv
}
```

\@collectverbenv

#1: <code>

Changes catcodes and ensures that spaces are displayed normally.

```
\def\@collectverbenv#1{%
    \newverb@catcodes
    \@vobeyspaces
    \frenchspacing
    \@noligs
    \expandafter\@@collectverbenv\expandafter{\@currenvir}{#1}%
}
```

\@scollectverbenv

#1: <code>
Changes catcodes.

```
\def\@scollectverbenv#1{%
    \newverb@catcodes
    \@noligs
    \expandafter\@@collectverbenv\expandafter{\@currenvir}{#1}%
}
```

\@@collectverbenv

#1: <envname>
#2: <code>

```
\begingroup
\catcode`\|=0
\catcode`\(=1
\catcode`\)=2
\@makeother\{
\@makeother\}
\@makeother\\
|catcode`|^^M=|active%
|gdef|@@collectverbenv#1#2(%
    |long|def|@@@collectverb##1^^M##2^^M\end{#1}(#2(##2)|endgroup|end(#1))%
    |@@@collectverb%
)%
|endgroup%
```

\Collectverb

Collects argument as plain verbatim and feeds it to the given code. The text is suitable to be printed to auxiliary files.

```
\newcommand*\Collectverb{%
    \begingroup
    \@ifstar
        \@sCollectverb
        \@Collectverb
}
```

\@Collectverb

```
\def\@Collectverb#1{%
    \verb@eol@error
    \let\do\@makeother
    \dospecials
    \obeyspaces
    \@@Collectverb{#1}%
}
```

\@sCollectverb

```
\def\@sCollectverb#1{%
    \verb@eol@error
    \let\do\@makeother
    \dospecials
    \@@Collectverb{#1}%
}
```

\@@Collectverb

```
\def\@@Collectverb#1#2{%
    \ifnum`#2=`\{%
        \catcode`\}\active
    \else
        \catcode`#2\active
    \fi
    \begingroup
    \ifnum`#2=`\{%
        \lccode`\~`\}%
    \else
        \lccode`\~`#2%
    \fi
    \lowercase{\endgroup
        \def\@@@Collectverb##1~}{\endgroup#1{##1}}%
    \@@@Collectverb
}
```

\Collectverbenv

Collects environment content as plain verbatim and feeds it to the given code. The text is suitable to be printed to auxiliary files.

```
\newcommand*\Collectverbenv{%
    \begingroup
    \@ifstar
        \@sCollectverbenv
        \@Collectverbenv
}
```

\@Collectverbenv

```
\def\@Collectverbenv#1{%
    \newverb@catcodes
    \obeyspaces
    \expandafter\@@Collectverbenv\expandafter{\@currenvir}{#1}%
}
```

\newverb@catcodes

```
\begingroup
\catcode‘\^^I=\active
\gdef\newverb@catcodes{%
    \let\do\@makeother
    \dospecials
    \obeylines
    \endlinechar=13
    \catcode‘\^^I=\active
    \def^^I{\newverb@tab}%
}
\gdef^^I{\newverb@tab}%
\endgroup
```

\newverb@tab

```
\edef\newverb@tab{\space}%\space\space\space}
```

\@sCollectverbenv

```
\def\@sCollectverbenv#1{%
    \newverb@catcodes
    \expandafter\@@Collectverbenv\expandafter{\@currenvir}{#1}%
}
```

\@@Collectverbenv

```
\begingroup
\catcode`\|=0
\catcode`\(=1
\catcode`\)=2
\@makeother\{
\@makeother\}
\@makeother\\
|catcode`|^^M=|active%
|gdef|@@Collectverbenv#1#2(%
    |long|def|@@@Collectverb##1^^M##2^^M\end{#1}(|endgroup#2(##2)|end(#1))%
    |@@@Collectverb%
)%
|gdef|misj(|def^^M(^^J))%
%|gdef|misj(|def^^M##1(|ifx##1|endmarker|else|noexpand^^M|expandafter##1|fi))%
|endgroup%
```

\newverbenvironment

```
\newcommand\newverbenvironment{}
\long\def\newverbenvironment#1#2#{%
    \@newverbenvironment{#1}{#2}%
}
\long\def\@newverbenvironment#1#2#3#4{%
    \newenvironment{#1}#2{%
        \begingroup
        \newverb@catcodes
        \obeyspaces
        #3%
        \expandafter\@@Collectverbenv\expandafter{\@currenvir}{\long\def\BODY}%
    }{#4}%
}
```

\newverbsfont

```
\newcommand\newverbsfont{%
    \verbatim@font
    \frenchspacing
}
```

\Verbdef

```
\newcommand*\Verbdef{%
    \@ifstar
        {\@Verbdef*}%
        {\@Verbdef{}}%
}
```

\@Verbdef

```
\newcommand*\@Verbdef[2]{%
    \Collectverb#1{\def#2}%
}
```