niceverb.sty

—

Minimizing Markup for Documenting LaTeX packages

Uwe Lück
http://contact-ednotes.sty.de.vu

April 16, 2009

Abstract

niceverb.sty provides very decent syntax for describing LaTeX packages and the syntax of macros conforming to LaTeX syntax conventions.

Contents

1 Presenting niceverb

1.1 Purpose

The niceverb package provides "minimal" markup for documenting LaTeX packages, reducing the number of keystrokes/visible characters needed (kind of poor man's WYSIWYG).[1] It conveniently handles command names in arguments of macros such as `\footnote` or even of sectioning commands. If you use makedoc.sty additionally, commands for typesetting a package's code are inserted automatically (just using TeX). As opposed to tools that are rather common on UNIX/Linux, this operation should work at any TeX installation, irrespective of platform.

Both packages may at least be useful while working at a very new package and may suffice with small, simple packages. After having edited your package's code (typically in a `.sty` file), you just "`latex`" the manual file (maybe some `.tex` file) and get instantly the corresponding refreshed documentation.

niceverb and makedoc may also help to generate without much effort documentations of nowadays commonly expected typographical quality for packages that so far only had plain text documentations.

1.2 Acknowledgement/Basic Ideas

Three ideas of Stephan I. Böttcher's in documenting his lineno.sty inspired the present work:

1. The markup and its definitions are short and simple, markup commands are placed at the right "margin" of the ASCII file, so you hardly see them in reading the source file, you rather just read the text that will be printed.

2. An awk script removes the `%`s starting *documentation* lines and inserts the commands for typesetting the package's *code* (you don't see them in the source).

3. An active character ('`|`') issues a `\string` *and* switches to typewriter typeface for typesetting a command verbatim—so this works without changing category codes (which is the usual idea of typesetting code), therefore it works even in macro arguments.

1.3 The Commands and Features of niceverb

Single quotes `‘`, `’`, "less than" `<` (accompanied with `>`), the hash mark `#`, ampersand `&`, and in an extended "auto mode" even backslash `\` become `\active` characters with "special effects." '`|`...`|`' should highlight descriptions of user commands and their syntax.

[1] "What you see is what you get." Novices are always warned that WYSIWYG is essentially impossible with LaTeX.

The package mainly aims at typesetting commands and descriptions of their syntax *if the latter is "standard LaTeX-like"*, using "meta-variables." A string to be typeset "verbatim" thus is assumed to start with a single command like `\foo`, maybe followed by stars ('`*`') and pairs of square brackets (`[...]`) or curly braces (`{...}`), where those pairs contain strings indicating the typical kinds of contents for the respective arguments of that command. A typical example is this:

`\foo*[`⟨*opt-arg*⟩`]{`⟨*mand-arg*⟩`}`

This was achieved by typing

```
&\foo*[<opt-arg>]{<mand-arg>}
```

In "auto mode" of the package, even typing

```
\foo*[<opt-arg>]{<mand-arg>}
```

would have sufficed—WYSIWYG! (I call such mixtures of *verbatim* and "meta-variables" *'meta-code'*.)

Now for the details:

"Meta-variables:" The package supports the "angle brackets" style of "meta-variables" (as with ⟨*meta-variable*⟩). You just type '`<foo>`' to get '⟨*foo*⟩'.

This works due to a sloppy variant `\SimpleVerb` of `\verb` which doesn't care about possible ligatures and definitions of active characters. Instead, it assumes that the "verbatim" font doesn't contain ligatures anyway. '`\verb|<foo>|`', by contrast, just yields '`<foo>`'.

Almost the same feature is offered by ltxguide.cls which formats the basic guides from the LaTeX Project Team. The present feature, however, also works in plain text outside verbatim mode.

Single quotes (left/right) for "short verb:" The package "assumes" that *quoting* refers to *code*, therefore '`‘foo’`' is typeset as '`foo`'. This somewhat resembles the `\MakeShortVerb` feature of doc.sty.

It will typically fail when you try to typeset commands (and their syntax) in *macro arguments*—e.g.,

```
\footnote{‘\bar’ is a celebrated fake example!}
```

will try to *execute* `\bar` instead of typesetting it, giving an "undefined" error so. `\verb` fails in the same situation, for the same reason. '`&`' (`\footnote{&\bar`⟨*remaining*⟩`}`) or "auto mode" (see below) may then work better.

Double quotes and apostrophes should still work the usual way; otherwise you could control the parsing mechanisms using curly braces (outside and inside don't interact). To get usual single quotes, you can use their standard substitutes `\lq` and `\rq`. You can "abuse" this "single quotes" feature

just to get typewriter typeface.[2] For difficult cases, you can still use the standard `\verb` command from LaTeX.

Single right quotes for `\textsf`: Package names are (by some unwritten convention!?) typeset with `\textsf`; it was natural to use a remaining case of using single quotes for abbreviating `\textsf{`⟨*text*⟩`}` by `'`⟨*text*⟩`'`.[3]

This idea of switching fonts continues font switching of wiki.sty which uses the syntax for editing Wikipedia pages.

Ampersand '`&`' typesets command syntax even in arguments: e.g., type '`&\foo{<arg>}`' to get '`\foo{`⟨*arg*⟩`}`'. This may be even more convenient for typing than the single quotes method, although looking somewhat strange. However, `&` may terminate *verbatim* unexpectedly, being designed for displaying "LaTeX-like command syntax" in the first instance.[4][5]

This choice of `&` rests on the assumption that there won't be many tables in the documenation. You can restore the usual meaning of `&` by `\MakeAlign\&` and turn the present special meaning on again by

`\MakeActiveLet\&\CmdSyntaxVerb`.

You could also redefine (`\renewcommand`) `\descriptionlabel` using `\CmdSyntaxVerb` (the "normal command" equivalent to `&`) so `\item[\foo]` works as wanted.

"Auto mode" typesets commands verbatim unless ... In "auto mode", the backslash '`\`' is an active character that builds a command name from the ensuing letters and typesets the command (and its syntax, allowing meta-variables) verbatim. However, there are some exceptions, which are collected in a macro `\niceverbNoVerbList`. `\begin`, `\end`, and `\item` belong to this list, you can redefine (`\renewcommand`) it. There is also a command `\NormalCommand{`⟨*letters*⟩`}` *issuing* the command `\`⟨*letters*⟩ instead of typesetting it. Since auto mode is somewhat dangerous, you have to start it explicitly by `\AutoCmdSyntaxVerb`. You can end it by `\EndAutoCmdSyntaxVerb`.

Auto mode is motivated by the observation that there are package files containing their documentation as pure (well-readable) ASCII text—containing the names of the new commands without any kind of quotation marks or verbatim commands. Auto mode should typeset such documentation just from the same ASCII text.

Hash mark `#` comes verbatim. No macro definitions are expected in the `document` environment.[6] Rather, '`#`' is an active character for taking the

[2] In macro arguments this requires that the right single quote `'` is `\active`.

[3] Font switching by sequences of single quotes is a feature of the syntax for editing Wikipedia pages and of wiki.sty.

[4] Moreover, `&` currently has a limited xspace functionality only.

[5] You can even use `&` for referring to active characters like `&` in footnotes etc.!

[6] This idea recently appeared on the latex-l mailing list. It may be wrong, not sure at the moment, think of latexa.ltx ...

next character (assuming it is a digit) to form a reference to a *macro parameter*—'`#1`' becomes '`#1`' (WYSIWYG indeed!).

1.4 Examples

The local configuration file mdcorr.cfg documents itself using niceverb syntax. Its code and the documentation that is typeset from it are at the end of the 'examples' section of makedoc.pdf. (2009/04/15)

Moreover, the documentation niceverb.pdf of niceverb.sty was typeset from niceverb.tex and niceverb.sty using niceverb syntax, likewise fifinddo.pdf and makedoc.pdf. The example of niceverb shows the most frequent use of the `&` feature.

It seems to me that I could type so many pages on fifinddo and makedoc in little more than a week only due to the "minimal" *verbatim* and syntax-display syntax.

1.5 What is Wrong with the Present Version

1. niceverb.sty should be an extension of wiki.sty; yet their font selection mechanisms are currently not compatible.

2. Font switching or horizontal spacing may fail in certain situations (parentheses, titles, footnotes; you can correct spacing by '\␣').

3. The "vertical" character '`|`' was planned to be active and provide a version of the `decl` environment of ltxguide.cls. Currently makedoc.cfg installs inline, coloured/framed boxes instead (2009/04/09). They have their merits! See fifinddo.pdf and makedoc.pdf. However, they badly deal with long command names and many arguments ... (They could also issue an `\index` command!)—Doubled verticals could make the difference, calling a `decl` table indeed.

4. "Auto mode" has not been tested on a serious application yet.

5. niceverb's font switching tricks sometimes turn against their inventor (and other users?). There must be some switching "off" (and "on" again).[7] Also, there might better help with weird errors, some syntax checks might intercept earlier.

   Similarly, some choices reflect a personal style and should be modifiable, especially by package options.[8]

[7] fifinddo/makedoc allow inserting such commands from a driver script, invisible in the file that contains the "contentual" documentation.

[8] Please sponsor the project or support it otherwise!

2 Implementation of the Markup Syntax

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{niceverb}[2009/04/15 v0.2
                           minimize \string\verb\space code (UL)]

%% Copyright (C) 2009 Uwe Lueck,
%% http://www.contact-ednotes.sty.de.vu
%% -- author-maintained in the sense of LPPL below --
%%
%% This file can be redistributed and/or modified under
%% the terms of the LaTeX Project Public License; either
%% version 1.3a of the License, or any later version.
%% The latest version of this license is in
%%     http://www.latex-project.org/lppl.txt
%% We did our best to help you, but there is NO WARRANTY.
%%
%% Please report bugs, problems, and suggestions via
%%
%%   http://www.contact-ednotes.sty.de.vu
%%
```

2.1 Switching category codes

```
\providecommand{\CatCode}{\catcode`}
% \providecommand*{\MakeActive}[1]{\CatCode#1\active}
\providecommand*{\MakeAlign} [1]{\CatCode#14\relax}
\providecommand*{\MakeLetter}[1]{\CatCode#111\relax}
\providecommand*{\MakeOther} [1]{\CatCode#112\relax}
\providecommand*{\MakeSub}   [1]{\CatCode#18\relax}

\newcommand*{\MakeActiveLet}[2]{%% cf. \@sverb & \do@noligs
  \CatCode#1\active
  \begingroup
    \lccode`\~`#1\relax \lowercase{\endgroup \let~#2}}

\MakeLetter\_

% \newcommand*{\make_iii_other}{\MakeOther\\\MakeOther\{\MakeOther\}}
  %% <- replaced 2009/04/05
```

2.2 Sloppy variant of \verb

```
\newcommand*{\begin_min_verb}{%
  \relax \ifmmode \hbox \else \leavevmode\null \fi
    %% <- standard, for $$...$$
  \bgroup \tt %%%\let\do\MakeOther \dospecials
    \MakeLetter\_\MakeLetter\@}
\newcommand*{\SimpleVerb}[1]{%
```

Mainly avoid \verb's noligs list which overrides definitions of some active characters, while cmtt doesn't have any ligatures anyway.

```
    \ifx\protect\@typeset@protect
      \begin_min_verb %%%\make_iii_other %% much usual 2009/04/05
        \let\do\MakeOther \dospecials    %% 2009/04/06
        \MakeOther\|%% probably 'niceverb'
        \MakeActiveLet#1\niceverb_egroup %% 2009/04/11
        \verb@eol@error %% TODO change message 2009/04/09
    \else \string\SimpleVerb \string#1\fi}
```

2009/04/11: about etc.

```
  \newcommand*{\niceverb_egroup}{\egroup\ifmmode\else\@\fi}
```

2.3 Single quotes typeset meta-code

```
  \newcommand*{\lq_verb}{%
    \ifx\protect\@typeset@protect
      \expandafter \lq_double_test
    \else \lq \fi}
  %     \ifcat\noexpand'\noexpand~%
  %       \expandafter\expandafter\expandafter
  %         \protect_corresp_quotes
  %     \else
  %       \rq_verb
  %     \fi
  %   \fi}
  \newcommand*{\lq_double_test}{%
    %% test settles next catcode, better switch to ``other''
    %% in advance:
    \begingroup \let\do\MakeOther \dospecials
      \futurelet\let_token \lq_double_decide}
  \newcommand*{\lq_double_decide}{%
    \ifx\let_token\lq_verb
        \endgroup
      \lq\lq \expandafter \@gobble
```

Corresponding right quotes will become "other" due to having no space at the left.

```
    %%TODO to be changed with 'wiki.sty'.
    \else
        \endgroup
  %     \expandafter \rq_verb
      \expandafter \SimpleVerb \expandafter \'%
    \fi}
  % \newcommand*{\rq_verb}{\SimpleVerb\'}
  % \AtBeginDocument{\MakeActiveLet\`\rq_verb}
  \AtBeginDocument{\MakeActiveLet\`\lq_verb}
    %% Strings referred to will be code
    %% TODO to be changed with wiki.sty
```

2.4 Ampersand typesets meta-code

```
\newcommand*{\CmdSyntaxVerb}{%
  \ifx\protect\@typeset@protect
    \expandafter \cmd_syntax_verb
  \else %% thinking of .aux only
    \string\CmdSyntaxVerb \expandafter \string
  \fi}
\newcommand*{\cmd_syntax_verb}[1]{%
  \begin_min_verb \string#1\futurelet\let_token \after_cs}

\AtBeginDocument{\MakeActiveLet\&\CmdSyntaxVerb}
  %% not needed with \Auto... OTHERWISE useful in args!
  %% TODO \MakeAmpCmdSyntax
  %% TODO \let\endcell& (wie \endline, \endgraf) \MakeEndCell
\newcommand*{\after_cs}{%
  \ifcat\noexpand\let_token a\egroup \space
  \else \expandafter \decide_verb \fi}
\newcommand*{\test_more_verb}{\futurelet\let_token \decide_verb}
\newcommand*{\decide_verb}{%
%  \show\let_token
    \jumpteg_on_with\bgroup\braces_verb
    \jumpteg_on_with[\brackets_verb
    \jumpteg_on_with*\star_verb
  \egroup}
    %% CAUTION/TODO wrong before (... if cmd without arg
    %%         use \ then or choose usual verb...
    %%         or \MakeLetter\( etc. ... or \xspace
\newcommand*{\jumpteg_on_with}[2]{%
  \ifx\let_token#1\do_jumpteg_with#2\fi}
 %% TODO cf. xfor, xspace (break@loop);
 %% \DoOrBranch#1...#1 or so
\def\do_jumpteg_with#1#2\egroup{\fi#1}
\def\braces_verb#1{\string{#1\string}\test_more_verb}
\def\brackets_verb[#1]{[#1]\test_more_verb}
\def\star_verb*{*\test_more_verb}
```

2.5 Escape character typesets meta-code

```
\DeclareRobustCommand*{\BuildCsSyntax}{%
  \futurelet\let_token \build_cs_syntax_sp}
\newcommand*{\build_cs_syntax_sp}{%
  \ifx\let_token\@sptoken \else %% TODO ^^M!?
    \expandafter \start_build_cs_syntax
  \fi}
\newcommand*{\start_build_cs_syntax}[1]{%
  \edef\string_built{\string#1}%% #1 may be active
  \MakeLetter\_\MakeLetter\@%% CAUTION, cf. ...
  \test_more_cs}
\newcommand*{\test_more_cs}{%
```

```
126     \futurelet\let_token \decide_more_cs}
127   \newcommand*{\decide_more_cs}{%
128     \ifcat\noexpand\let_token a\expandafter \add_to_cs
129     \else
130       \MakeSub\_\MakeOther\@%
131       \expandafter \in@ \expandafter
132         {\csname \string_built \expandafter \endcsname
133           \expandafter}\expandafter{\niceverbNoVerbList}%
134       \ifin@
135         \csname \string_built
136           \expandafter\expandafter\expandafter \endcsname
137       \else
138         \begin_min_verb \@backslashchar\string_built
139           \expandafter\expandafter\expandafter \test_more_verb
140       \fi
141     \fi}
142     %% TODO such \if nestings with ifthen!?
143    %% cf.:
144   %  \let\let_token,\typeout{\meaning\let_token}
145     %% TEST TODO fuer xspace!? (\ifin@)
146   \newcommand*{\add_to_cs}[1]{%
147     \edef\string_built{\string_built#1}\test_more_cs}
148   \newcommand*{\AutoCmdSyntaxVerb}{\MakeActiveLet\\\BuildCsSyntax}
149     %% TODO or \niceverbswitch...
150
151   \newcommand*{\EndAutoCmdSyntaxVerb}{\CatCode\\\z@}
152   \newcommand*{\NormalCommand}{} \let\NormalCommand\@nameuse
153     %% Were tests:
154   %  \futurelet\LetToken\relax \relax
155   %  \show\LetToken \typeout{\ifcat\noexpand\LetToken aa\else x\fi}
156
157   \newcommand*{\niceverbNoVerbList}{%
158     \begin\end\item\verb\EndAutoCmdSyntaxVerb\NormalCommand
159     \section\subsection\subsubsection}%% TODO!?
160   \providecommand*{\AddToMacro}[1]{%
161     \expandafter \def \expandafter #1\expandafter}
```

2.6 Meta-variables

```
162   \def\niceverb_meta#1>{%
163     \mbox{\normalfont\itshape $\langle$#1\/$\rangle$}}
164     %% TODO offer without angles as well
165   \AtBeginDocument{\MakeActiveLet\<\niceverb_meta}
166     %% difference to ltxguide.cls: also outside verbatim
```

2.7 Hash mark is code

```
167   \newcommand*{\param_verb}[1]{{\tt\##1}}
168   \AtBeginDocument{\MakeActiveLet\#\param_verb}
```

2.8 Single right quote for \textsf

```
\newcommand*{\niceverb_rq_sf}{%
%   \relax %% in case of \hskip1sp 2009/04/08
            %% but spoils ligature  2009/04/09
  \ifx\protect\@typeset@protect
    \expandafter \niceverb_rq_sf_test
  \else \rq \fi}
```

Another macro just to avoid more sequences of \expandafter:

```
\newcommand*{\niceverb_rq_sf_test}{%
  \ifhmode
    \ifdim\lastskip>\z@
      \expandafter\expandafter\expandafter \niceverb_rq_sf_exec
    \else
      \ifnum\spacefactor=1001 %% in parentheses 2009/04/10
        \expandafter\expandafter\expandafter\expandafter
          \expandafter\expandafter\expandafter
            \niceverb_rq_sf_exec
      \else \rq \fi
%       \rq
    \fi
  \else \ifvmode
    \expandafter\expandafter\expandafter \niceverb_rq_sf_exec
    \else \rq \fi
  \fi}
\sfcode`\(=1001 %% enable in parentheses 2009/04/10
{\CatCode\'\active \gdef\niceverb_rq_sf_exec#1'{\textsf{#1}}}
  %% TODO to be changed with wiki.sty:
\AtBeginDocument{\MakeActiveLet\'\niceverb_rq_sf}

  %% TODO \niceverbswitch{<mode>}{<on/off>} (bzw. Doku ohne {})
```

2.9 Leave package mode

```
\MakeSub\_

\endinput
```

2.10 VERSION HISTORY

```
v0.1   2009/02/21   very first, sent to CTAN
v0.2   2009/04/04   ...NoVerbList: \subsubsection, \AddToMacro,
       2009/04/05   \SimpleVerb makes more other than iii
       2009/04/06   just uses \dospecials
       2009/04/08   debugging code for rq/sf, +\relax
       2009/04/09   +\verb@eol@error, prepared for new doc method,
                    removed spurious \makeat..., -\relax (ligature),
       2009/04/10   ('-trick
```

```
208         2009/04/11   \@ after \SimpleVerb
209         2009/04/14   noted TODO below
210         2009/04/15   change v0.1 to 2009/02/21
211
212   TODO: choose expansions of active characters by options 2009/04/10
213   TODO: &\@tempa and &\_tempa fail 2009/04/14
214
```