niceverb.sty

—

Minimizing Markup for Documenting LaTeX packages*

Uwe Lück†

November 9, 2015

Abstract

niceverb.sty provides very decent syntax (through active characters) for describing LaTeX packages and the syntax of macros conforming to LaTeX syntax conventions.

Keywords: literate programming, syntactic sugar, .txt to .tex enhancement, macro programming

Contents

*This document describes version v0.61 of niceverb.sty as of 2015/11/09.

†http://contact-ednotes.sty.de.vu

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—⟨*jobname*⟩`.sty`), you just "`latex`" the manual file (maybe some `.tex` file—⟨*jobname*⟩`.tex`) and get instantly the corresponding updated 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

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

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

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 these commands in the source).[2]

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.

4. '`<meta-variable>`' produces '⟨*meta-variable*⟩'. ('`\lt`' stores the original '<'.)

1.3 The Commands and Features of niceverb

Actually, it is the main purpose of niceverb to save you from "commands" ...

Single quotes `‘`, `’`, "less than" `<` (accompanied with `>`), the "vertical" `|`, the hash mark `#`, ampersand `&`, and in an extended "auto mode" even backslash `\` become `\active` characters with "special effects."

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 ('`[`⟨*opt-arg*⟩`]`') or curly braces ('`{`⟨*mand-arg*⟩`}`'), 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'*.

Outside macro arguments, you obtain the same by typing

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

Details:

[2] The corresponding part of the "present work" is makedoc.sty.

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

This works due to a sloppy variant `\NVerb` 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.[3] '`\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`', or (generally) `‘`⟨*content*⟩`’` turns ⟨*content*⟩ into meta-code with the meta-variable feature as above. This somewhat resembles the `\MakeShortVerb` feature of doc.sty. You can "abuse" our feature just to get typewriter typeface.

Problems with this feature will typically arise 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 or so. `\verb` fails in the same situation, for the same reason. '`&`' (`\footnote{&\bar`⟨*remaining*⟩`}`) or "auto mode" (see below) may then work better.[4] More generally, the quoting feature still works in macro arguments in the sense that you then have to mark difficult characters with `&` (simply as short for `\string`). However, it still won't work with curly braces that don't follow a command name (such *pairs* of braces will simply get lost, *single* braces will give errors or so).

Double quotes and apostrophes should still work the usual way. For difficult cases, you can still use the standard `\verb` command from LaTeX. To get *usual* single quotes, you can use their standard substitutes `\lq` and `\rq`, or for pairs of them, `\qtd{`⟨*text*⟩`}` in place of `\lq`␣⟨*text*⟩`\rq`—or even `\lq`␣⟨*text*⟩`\rq\`␣. To get single quotes around some verbatim ⟨*verb*⟩, often `\qtd{&`⟨*verb*⟩`}` works. It is for this reason that I have refrained from different solutions as in newverbs (so far).

v0.44 provides `\AddQuotes` after which single quotes *both* turn their content into metacode *and print* single quotes around them *automatically.* This can be turned off again by `\DontAddQuotes`.

Single right quotes for `\textsf`: Package names are (by some convention I often yet not always see working) typeset with `\textsf`; it was natural to use a remaining case of using single quotes for abbreviating

`\textsf{`⟨*text*⟩`}`

[3] On the other hand, `\NVerb` is more *careful* with niceverb's special characters.
[4] `\bar` indeed!

by `'⟨text⟩'`. This idea of switching fonts continues font switching of wiki.sty which uses the syntax for editing *Wikipedia* pages (font switching by sequences of right single quotes).

Verticals for setting-off command descriptions: `|⟨code⟩|` works like ``⟨code⟩'' except putting the result into a *framed box* (just as all around here)—or something else that you can achieve using some *hooks* described with the implementation. There are variants like `\cmdboxitem|⟨code⟩|`.

Ampersand shows command syntax &c. 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, in macro arguments this does not work with *private letters* (`@` and `_` here), for this case, use `\cs{⟨characters⟩}` or `\cstx{⟨characters⟩}⟨parameters⟩`.[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 `\MakeNormal\&` and turn the present special meaning on again by

`\MakeActive\&` or `\MakeActiveLet\&\CmdSyntaxVerb`

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

Another feature of niceverb's `&` is getting (some of the) special characters (as listed in the standard macro `\dospecials`) verbatim in arguments (where `\verb` and the like fail). It just acts similarly as TEX's primitive `\string` (which it actually invokes—cf. discussion on the left quote feature above).

"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, or add ⟨*macros*⟩ to it by `\AddToNoVerbList}{⟨macros⟩}` 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`. `\AutoCmdInput{⟨file⟩}` is probably most important.

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

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

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 next character (assuming it is a digit) to form a reference to a *macro parameter*—'`#1`' becomes '`#1`'–WYSIWYG indeed! (So the general syntax is `#`⟨*digit*⟩.)

Escaping from niceverb (generally). To get rid of the functionality of some active character ⟨*char*⟩ ('`&`', single quote, ampersand, hash mark—not "auto mode," see above) here, use `\MakeNormal\`⟨*char*⟩—may be within a group. To revive it again, use `\MakeActive\`⟨*char*⟩. This may fail when a different package overtook the active ⟨*char*⟩ (but I expect more failures then), in this case `\MakeActiveLet\`⟨*char*⟩`\`⟨*perm-alias*⟩ revives the niceverb meaning of ⟨*char*⟩ where `\`⟨*perm-alias*⟩ is the "permanent alias" for that active ⟨*char*⟩ according to the documentation below. E.g., `\LQverb` is the "permanent alias" for active single left quote, niceverb activates it by `\MakeActiveLet\'\LQverb`.—You can turn off niceverb syntax *altogether* by `\noNiceVerb` and revive it by `\useNiceVerb` (without "auto mode").

Right Quotes: Disabling/reviving replacement of `\textsf` by single right quotes requires

`\nvRightQuoteNormal` or `\nvRightQuoteSansSerif`

respectively.—The feature fails in certain occasions because a single right quote must not always be interpreted as `\textsf`, and deciding this by macros became quite laborious for me and is most likely still not perfect. There is a command `\nvAllRightQuotesSansSerif` to be used with care that interpretes *all* single right quotes as `\textsf`, which, e.g., means that you must use `\rq` for apostrophes.

"Moving" arguments: `\NiceVerbMove{`⟨*text*⟩`}` with v0.6 is for "moving" arguments so that niceverb syntax operates *locally* at the destination (table of contents or page headings). It is automatically used by niceverb's variant of LaTeX's sectioning commands; while with `\markboth`, `\markright`, `\addcontentsline` etc. you must it include yourself (currently, TODO?).

1.4 Examples

The file mdoccorr.cfg providing some `.txt`→LaTeX functionality—i.e., typographical corrections—documents itself using niceverb syntax. Its code and the documentation that is typeset from it are in the 'examples' section

[6] This idea appeared 2009 on the LATEX-L mailing list. It may be wrong, as I have sometimes experienced ...

of makedoc.pdf.—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.

nicetext bundle release v0.4 contains a file substr.tex that should typeset the documentation of the version of Harald Harders' substr.sty[7] that your TEX finds first, as well as arseneau.tex typesetting a few packages by Donald Arseneau. The outcomes (with me) are substr.pdf and arseneau.pdf. These are the first applications of niceverb's "auto mode" to (unmodified) third-party package files. (I also made a more ambitious documentation of Donald Arseneau's import.sty v3.0 before I found that CTAN already has a nicely typeset documentation of import.sty v5.2.)

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. Especially, the feature of

   ''⟨*text*⟩''

   replacing `\textit{`⟨*text*⟩`}` or `\emph{`⟨*text*⟩`}` may be considered missing.

2. Font switching or horizontal spacing may fail in certain situations. You can correct spacing by '\␣'.

3. The "vertical" character '|' produces inline boxes only at present. It might as well provide a version of the `decl` tabular environment of ltxguide.cls. The inline boxes badly deal with long command names and many arguments. Doubled verticals could ensure the `decl` mode. Moreover, such a box might issue an *index* entry.

4. One may have *opposite* ideas about using quotes—maybe rather "⟨*code*⟩" should typeset ⟨*code*⟩ *verbatim*. There might be a package option for this. If ordinary ''⟨*text*⟩"' still should work, awful tricks as now with the right quote feature would be needed.

5. "auto mode" seems not to work in section titles. (2011/01/26)

6. Certain difficulties with typesetting code in macro arguments may be overcome easily using ε-TEX features, I need to find out ...

[7] `http://ctan.org/pkg/substr`

2 The Package File

2.1 Preliminaries

2.1.1 File Header

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{niceverb}[2015/11/09 v0.61
                           minimize doc markup (UL)]

%% Copyright (C) 2009-2012, 2014 2015 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
%%
```

2.1.2 \newlet

`\newlet`⟨*cmd*⟩⟨*cnd*⟩ counters the risk of mistyping ⟨*cmd*⟩ with `\@ifdefinable`, and even saves some code lines:

```
\providecommand*{\newlet}[2]{\@ifdefinable#1{\let#1#2}}
\@onlypreamble\newlet
```

2.1.3 Switching Category Codes

Underscore as a "private letter," using stacklet with v0.5:

```
\RequirePackage{stacklet} \PushCatMakeLetter\_          %% 2012/08/27
```

v0.3 introduced `\AssignCatCodeTo` and `\MakeNormal`. v0.5 abolishes the former again and uses actcodes for some part of `\catcode` switching:

```
\RequirePackage{actcodes}
```

`\CatCode{\`⟨*character*⟩`}` (or simply `\CatCode\`⟨*character*⟩) saves one token per use and works when the category code of ‘‘’ ("single left quote") has changed. As of v0.5, it may be defined by a different package:

```
\providecommand*{\CatCode}{\catcode`}     %% \provi... 2012/08/27
% \newcommand*{\CatCode}[1]{\catcode`#1 } %% no better 2010/02/27
```

`\CatCode` is near to be moved into the catcodes bundle, and basic commands from stacklet and actcodes may be reimplemented using it (manycats; allcats for loading entire catcodes in good order).

`\AssignCatCodeTo{`⟨*number*⟩`}{\`⟨*char*⟩`}` no longer is considered useful (counted tokens in `memory.tex`) and replaced by `\CatCode`.

```
% \newcommand*{\AssignCatCodeTo}[2]{\catcode`#2=#1\relax}
```

`\MakeLetter\`⟨*char*⟩ is replaced by the stacklet package—I thought, but *here* it is also needed to declare the "private letters" of the package that is documented. This should be "variable." OK, the new (v0.5) `\private_letters` is a step towards this:

```
\newcommand*{\private_letters}{\CatCode\@11\CatCode\_11\relax}
```

`\MakeOther\`⟨*char*⟩ and `\MakeActive\`⟨*char*⟩ were implemented here before v0.5, now they are in actcodes ...

```
% \def        \MakeOther {\AssignCatCodeTo{12}}
```

`\MakeActiveLet\`⟨*char*⟩`\`⟨*macro-name*⟩ likewise is in actcodes. niceverb takes a copy `\MakeActiveLetHere` of it for dealing with hyperref (see Sec. 2.3.3). hyperref-compatibility of mere `\MakeActive` is not provided any longer:

```
\newlet\MakeActiveLetHere\MakeActiveLet
```

For restoring the usual category codes of TEX's special characters later, we store them now. (I.e., these characters are listed in the macro `\dospecials` that expands to

```
\do\ \do\\\do\{\do\}\do\$\do\&\do\#\do\^\do\_\do\%\do\~
```

their category codes are 10, 0, 1, 2, 3, 4, 6, 7, 8, 14, 13 respectively; "end of line", "ignored", "letter", "other", and "invalid" are missing—cf. *TEXbook* Chap. 7.)

```
\def\do#1{\expandafter
%   \chardef \csname normal_catcode_\string#1\expandafter \endcsname
```

← v0.6 2014/03/22: First I thought "too few `\expandafter`s"; actually the original `\expandafter` has no effect →

```
  \chardef \csname normal_catcode_\string#1\endcsname
    \CatCode#1\relax}
\dospecials
```

Tests: "normal category code" of `\` is 0, "normal category code" of `$` is 3; "normal category code" of `&` is 4.[8]

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

[8] LATEX's `\nfss@catcodes` is similar, but it makes space-like characters ignored. Also cf. ltfinal.dtx. TODO: `\RestoreNormalCatcodes`.

`\MakeNormal\⟨char⟩` saves you from remembering ...

```
\newcommand*{\MakeNormal}[1]{%
  \@ifundefined{\norm_catc_str#1}%
              {\MakeOther#1}%
              {\CatCode#1\csname\norm_catc_str#1\endcsname\relax}}
\newcommand*{\norm_catc_str}{normal_catcode_\string}
  %% TODO add ^^I and ^^M
  %% TODO save char tokens  %% 2012/08/27
```

We take a copy `\MakeNormalHere` of `\MakeNormal` as with `\MakeActive`.

```
\newlet\MakeNormalHere\MakeNormal
```

2.1.4 Robustness by `\IfTypesetting` or So

It seems we need some own ways of robustifying (as opposed to LaTeX's `\protect` and `\DeclareRobustCommand`—sometimes, especially for certain active characters) to achieve various compatibilities—using

`\IfTypesetting{⟨if⟩}{⟨unless⟩}`

It also saves some `\expandafter`s.

```
\providecommand*{\IfTypesetting}{%
%     \relax
```

← This `\relax` suppressed ligatures of single right quotes!

```
    \ifx \protect\@typeset@protect
         \expandafter \@firstoftwo
    \else \expandafter \@secondoftwo \fi}
```

`\nvSelfProtect{⟨cmd-char⟩}{⟨typeset⟩}`

is another idea. In "typesetting mode," ⟨*typeset*⟩ is run. Otherwise a single unexpanded token ⟨*cmd-char*⟩ remains. TODO bad at `\shipout`. No `\protect` appears, and as opposed to LaTeX's protection mechanism, running ⟨*typeset*⟩ does not require a second macro name. The idea is that `\nvSelfProtect{#1}{#2}` is the *definition* (substitution text—on token level) of ⟨*cmd-char*⟩.[9]

```
\newcommand*{\nvSelfProtect}[2]{%
    \ifx \protect\@typeset@protect
           \nv_expand_else{#2}%                %% braces 2014/03/26
%   \else  \protect#1\fi}
    \else  \noexpand#1\fi}                     %%  works 2014/03/28
\def\nv_expand_else#1\else#2\fi{\fi#1}
```

[9] This may go into a separate package under a different name later.

Sometimes "control sequences" get definitions with `\svSelfProtect` below whose first argument then is an active character—the "control sequence" then is the "permanent alias" of the active character. This is a somewhat "indirect self"-protection. At other places, the "self"-protection is more direct. Then `\NewSelfProtectedCommand{`⟨*cmd*⟩`}{`⟨*def*⟩`}` avoids mistakes from mistyping ⟨*cmd*⟩ and saves some code. It works like `\newcommand*`, provides the `\svSelfProtect`, and your definition ⟨*def*⟩ needs to contain the second argument of `\svSelfProtect` only. *Arguments* are not supported currently (TODO—well, 3 applications 2014/03/27):

```
\newcommand*{\NewSelfProtectedCommand}[2]{%
    \newcommand*#1{\nvSelfProtect#1{#2}}}
\@onlypreamble\NewSelfProtectedCommand
```

TODO 3 applications for the permanent alias case, saving catcode changes ...

Testing:

```
\newcommand*{\nvShowProtectedEdef}[1]{%
    \protected@edef\@tempa{#1}\show\@tempa}
```

2.1.5 Shared Shorthand Macros

`\begin_min_verb` is a beginning shared by some macros here. It begins like LaTeX's `\verb`, apart from the final `\tt`. `\bgroup` is needed for `\hbox` and must be balanced by an `\egroup` counterpart later.

```
\newcommand*{\begin_min_verb}{%
  \relax \ifmmode \hbox \else \leavevmode\null \fi
  \bgroup \tt}
```

For typographical additions ("decorations") to the verbatim material, we collect it in a box register addressed by `\niceverb_savebox`:

```
\newsavebox\niceverb_savebox
```

`\SetNiceVerbSaveBox` starts reading the ("meta-")verbatim material:

```
\newcommand*{\SetNiceVerbSaveBox}{%
    \setbox\niceverb_savebox\hbox\bgroup}
```

`\NVerb`, `\HardNVerb`, or `\NiceMaybeMetaVerb` with an optional argument about as `[\`⟨*id*⟩`_egroup]` should follow, cf. Sec. 2.2.1.—There have been two applications only up to now (2014/03/19), but this may change soon.[10]

`\TheNiceVerbSaveBox` allows referring to the verbatim material collected, in order to place it a *single time*—and no surrounding braces are needed:

```
\newcommand*{\TheNiceVerbSaveBox}{\box\niceverb_savebox}
```

TODO:

`\NewNiceVerbDecoration{\`⟨*deco*⟩`}{\`⟨*end-name*⟩`}{`⟨*start*⟩`}{`⟨*end-code*⟩`}`

might save from typing `\`⟨*end-name*⟩ twice and from typing the two `\egroup`s.

TODO left quote verb moving: braces get lost.

[10] Thought of `\qtdnverb`, but this doesn't need this kind of treatment.

2.2 Implementation of the "Nice" Syntax

2.2.1 \NVerb

Discovered mistakes in this section 2014/03/19, with respect to robustness. (i) `\NVerb` not really was meant to be a user command, to appear in documentation code (rather to be internal). (ii) The attempt to make it robust was incomplete. (iii) The code for "not typesetting" was strange. (iv) It is difficult to imagine that somebody attempts to use "verbatim" code, e.g., in a section title (while with our `&`/`\string` it's ok). *At least handy replacement for* `\textsf`. Well, let's see. May be I once find a useful application. So I repair the code—`\protect` before `\_...._false`. `\NVerb`⟨*char*⟩⟨*code*⟩⟨*char*⟩:

```
\newcommand*{\NVerb}{%
    \protect\_no_nice_meta_verb_false \NiceMaybeMetaVerb}
```

`\HardNVerb`⟨*char*⟩⟨*code*⟩⟨*char*⟩ does not recognize meta-variables:

```
\newcommand*{\HardNVerb}{%
    \protect\_no_nice_meta_verb_true \NiceMaybeMetaVerb}
\newif\if_no_nice_meta_verb_
```

v0.6 equips both `\NVerb[`⟨*end-cmd*⟩`]` and `\HardNVerb[`⟨*end-cmd*⟩`]` with an optional argument for a single parameter-less macro for what to do after reading verbatim text—for boxing or quoting etc.[11] `\niceverb_egroup` then is useless and removed. Macros that were assigned to it before v0.6 move into the new optional arguments.—Actually, the next macro `\NiceMaybeMetaVerb` shared by `\NVerb` and `\HardNVerb` gets the optional argument: TODO!?

```
% \newcommand*{\nice_maybe_meta_verb}[1]{%
```

```
\newcommand*{\NiceMaybeMetaVerb}[2][\niceverb_normal_egroup]{%
```

`\newcommand` with v0.6 must suffice for robustness, so removing 2014/03/20:

```
% \IfTypesetting{%
```

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

```
    \begin_min_verb
      \let\do\MakeOther \dospecials
```

Turn off niceverb specials:

[11] The goal resembles that of `\collectverb` in Martin Scharrer's newverbs. A difference in implementation is that the character delimiting the verbatim text is used as a parameter delimiter for a new/temporary macro. So the verbatim characters are fixed. Our approach will be collecting the verbatim material in a box, if we need something more complex than `\niceverb_mormal_egroup`. This allows changing category codes with `\MetaVar` again, although there hasn't been a need for this so far. It might be useful for allowing shorthand macros in `\MetaVar`'s argument.

```
78          \MakeOther\|\MakeOther\`\MakeOther\'%
79          \if_no_nice_meta_verb_ \MakeOther\<%
80          %%% \else    \MakeActiveLet\<\MetaVar     %% 2010/12/31
81          \else        \MakeActiveLetHere\<\MetaVar %% 2011/06/20
82          \fi
83    %     \MakeActiveLetHere #2\niceverb_egroup
84          \MakeActiveLetHere #2#1%                  %% 2014/03/18
```

After the previous line has worked, we use `\def` instead of `\let`, so there is no longer a need to choose a command name for the verbatim delimiter – well, **no**, don't define the same macro several times. Also, the same "end" macro might be used for different purposes, e.g., when a macro in an eventual expansion of the "end" macro is modified.

```
85    %     \MakeActiveDef     #2{#1}%                %% 2014/03/18
86          \verb@eol@error %% TODO change message 2009/04/09
87    %   }{\string\NVerb \string#1}}
```

← both `\string` very strange (second one finds ⟨*char*⟩—maybe it's active—but then its next occurrence delimiting the verbatim code will harm too!), also redirecting to `\NVerb`. (May have been ok for entries to auxiliary files.) New difficulties come from the optional argument, which needs protection as well.—Ok, the optional argument is not protected, and active characters ⟨*char*⟩ must "protect themselves," so use of `\IfTypesetting` changes, cf. Sec. 2.1.4.

```
88    }
```

[2014/03/19 removing/hiding remarks from 2009f. that I don't understand anymore . . .]

```
89    \newcommand*{\niceverb_normal_egroup}{%
90        \egroup
91        \niceverb_maybe_rq                    %% 2011/09/09 for \AddQuotes
92        \ifmmode\else\@\fi}
93    % \@ifdefinable\niceverb_egroup                        %% rm 2014/03/18
94    %     {\let\niceverb_egroup\niceverb_normal_egroup}
```

2.2.2 Single Quotes Typeset Meta-Code

`\LQverb` will be a "permanent alias" for the active left single quote.

The verbatim feature must not act when another single left quote is ahead—we assume a double quote is intended then, and we typeset it (thus the left quote feature does not allow to typeset something verbatim that starts with a single left quote). In page headers, a `\protect` could be in the way before v0.6. (A hook for `\relax`ing certain things in `\markboth` and `\markright` would have been an alternative. TODO)

```
95    \MakeActive\`
96      \newcommand*{\LQverb}{%
97    %   \IfTypesetting{\lq_double_test}{\protect`}}
```

New approach v0.6:

```
      \nvSelfProtect`\lq_double_test}
%   \IfTypesetting{\lq_double_test}{\noexpand`}}
\MakeOther\`
\newcommand*{\lq_double_test}{%
```

This test settles the next catcode, so better switch to "other" in advance (won't harm if left quote isn't next): TODO switch what?

```
  \begingroup
    \let\do\MakeOther \dospecials
    \MakeOther\|%% 2010/03/09!
    \futurelet\let_token \lq_double_decide}
\newcommand*{\lq_double_decide}{%
  \ifx\let_token\LQverb
    \endgroup
    ``\expandafter \@gobble
```

... alternative ...

```
%     \expandafter `%
```

does not recognize next left quote—why? TODO—Corresponding right quotes will become "other" due to having no space at the left. TODO to be changed with wiki.sty.

```
  \else
%     \ifx\let_token\protect                      %% rm. 2014/03/28
% %     \show\let_token           %% indeed before v0.6, 2014/03/24
%       \expandafter\expandafter\expandafter \lq_double_decide_ii
%     \else
      \endgroup
      \niceverb_maybe_qs                              %% 2011/09/09
      \expandafter\expandafter\expandafter \NVerb
        \expandafter\expandafter\expandafter \'%
%     \fi
  \fi}
```

`\lq_double_decide_ii` continues test behind `\protect`.

```
\newcommand*{\lq_double_decide_ii}[1]{%
    \futurelet\let_token \lq_double_decide}
```

2.2.3 Ampersand (or \cstx) Typesets Meta-Code

`\CmdSyntaxVerb` will be a permanent alias for the active `&`.

```
\MakeActive\&
  \newcommand*{\CmdSyntaxVerb}{%
    \IfTypesetting{%
      \begin_min_verb
```

v0.3 moves the previous line from `\cmd_syntax_verb` where it is too late to establish private letters according to next line which was in `\begin_min_verb` earlier—an important bug fix!

```
128         \private_letters                                    %% v0.5
129         \cmd_syntax_verb
130   %     }{\protect&\string}}
131   %     }{\noexpand&\string}}
```

... with `\string`, in an `\edef`, the following command cannot be properly typeset, so

```
132       }{\noexpand&\noexpand}}                               %% 2014/03/26
```

TODO actually test non-typesetting, maybe introduce macros that perform tests anywhere ...

```
133   \MakeNormal\&
134   \newcommand*{\cmd_syntax_verb}[1]{%
135     \string#1\futurelet\let_token \after_cs}
```

However, `&` (or `\CmdSyntaxVerb`) may fail with private letters, especially in *macro arguments*[12] and with hyperref in titles of *sections bearing* `\label`*s*, so we provide something like `\cs{`⟨*characters*⟩`}` from tugboat.sty.

```
136   \DeclareRobustCommand*{\cs}[1]{%
137   %   \begin_min_verb \backslash_verb #1\egroup}
```

... fails with `_` in footnote today (2014/03/19) so:

```
138     \begin_min_verb \withcsname\string#1\endcsname\egroup} %% v0.6
139   \newcommand*{\backslash_verb}{\char`\\}
```

Moreover, typing `&\par` in "short" *macro arguments* fails, you better type `\cs{par}` then. Likewise, `\cs{if`⟨*letters*⟩`}` and `\cs{fi}` is safer in case you want to skip some part of the documentation (e.g., a package option skips commented code) by `\if`⟨*letters*⟩`\fi`. Finally, there will be PDF bookmarks support for `\cs` rather than for a real `&` or `\CmdSyntaxVerb` analogue like `\cstx{`⟨*characters*⟩`}*[`⟨*opt*⟩`]{`⟨*mand*⟩`}` as follows.

```
140   \DeclareRobustCommand*{\cstx}[1]{%              %% corr. 2010/03/17
141   %   \begin_min_verb \backslash_verb #1%
```

v0.6 like above:

```
142       \begin_min_verb \withcsname\string#1\endcsname
143       \futurelet\let_token \after_cs}
144   \newcommand*{\after_cs}{%
145     \ifcat\noexpand\let_token a\egroup \space
146     \else \expandafter \decide_verb \fi}
147   \newcommand*{\test_more_verb}{\futurelet\let_token \decide_verb}
```

[12] TODO: vfoot2e.sty – see notes.

```
\newcommand*{\decide_verb}{%
    \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}
  %% not needed with \Auto... OTHERWISE useful in args!
```

As latex.ltx has `\endgraf` as a permanent alias for the primitive version of `\par` and `\endline` for `\cr`, we offer `\endcell` as a replacement for the original `&`:

```
\let\endcell&
```

2.2.4 Escape Character Typesets Meta-Code

`\BuildCsSyntax` will be a permanent alias for the active escape character.

```
\DeclareRobustCommand*{\BuildCsSyntax}{%
  \futurelet\let_token \build_cs_syntax_sp}
\newcommand*{\build_cs_syntax_sp}{%
  \ifx\let_token\@sptoken
    \@%                                  %% 2010/12/30
  \else %% TODO ^^M!?
    \expandafter \start_build_cs_syntax
  \fi}
\newcommand*{\start_build_cs_syntax}[1]{%
  \edef\string_built{\string#1}%
```

`#1` may be active.—With Donald Arseneau's import.sty (e.g.), '_' may be needed to be `\active` with the meaning of `\textunderscore`, therefore restoring its category code needs some more care than with v0.32 and earlier:

```
  \edef\before_build_cs_sub{\the\CatCode\_}%
  \private_letters                                  %% v0.5
  \test_more_cs}
\newcommand*{\test_more_cs}{%
  \futurelet\let_token \decide_more_cs}
\newcommand*{\decide_more_cs}{%
  \ifcat\noexpand\let_token a\expandafter \add_to_cs
  \else
%     \MakeNormalHere\_
```

Restoring '_' more carefully with v0.4 (\begingroup ... \endgroup!?):

```
      \CatCode\_\before_build_cs_sub
      \MakeOther\@%
%     \expandafter \in@ \expandafter
%       {\csname \string_built \expandafter \endcsname
%         \expandafter}\expandafter{\niceverbNoVerbList}%
      %% <- useless braces 2014/07/17 ->
      \expandafter \in@ \csname \string_built \expandafter
          \endcsname \expandafter {\niceverbNoVerbList}%
      \ifin@
        \csname \string_built
          \expandafter\expandafter\expandafter \endcsname
      \else
        \begin_min_verb \backslash_verb\string_built
          \expandafter\expandafter\expandafter \test_more_verb
      \fi
    \fi}
    %% TODO such \if nestings with ifthen!?
   %% cf.:
%  \let\let_token,\typeout{\meaning\let_token}
    %% TEST TODO fuer xspace!? (\ifin@)
\newcommand*{\add_to_cs}[1]{%
    \edef\string_built{\string_built#1}\test_more_cs}
```

\AutoCmdSyntaxVerb starts, \EndAutoCmdSyntaxVerb *ends* "auto mode."

```
\newcommand*{\AutoCmdSyntaxVerb}{%
    \MakeActiveLetHere\\\BuildCsSyntax}
\newcommand*{\EndAutoCmdSyntaxVerb}{\CatCode\\\z@}
```

\NormalCommand{⟨*characters*⟩} executes \⟨*characters*⟩ in "auto mode."

```
\newcommand*{\NormalCommand}{} \let\NormalCommand\@nameuse
```

Once I may want to use this feature in *Wikipedia*-like section titles as supported by makedoc, yet I cannot really apply the present feature soon, so this must wait ... (There is a special problem with \newlabel and hyperref ...)

Former tests:

```
%  \futurelet\LetToken\relax \relax
%  \show\LetToken \typeout{\ifcat\noexpand\LetToken aa\else x\fi}
```

\niceverbNoVerbList is the list of macros that will be *executed* instead of being typeset.

```
\newcommand*{\niceverbNoVerbList}{%
  \begin\end\item\verb\EndAutoCmdSyntaxVerb\NormalCommand
  \section\subsection\subsubsection} %% TODO!?
```

`\AddToMacro{\niceverbNoVerbList}{`⟨*macros*⟩`}` can be used to add ⟨*macros*⟩ to that list.

```
\providecommand*{\AddToMacro}[2]{%   %% TODO move to ... 2010/03/05
  \expandafter \def \expandafter #1\expandafter {#1#2}}
  %% <- was very wrong 2010/03/18
```

Hey, or just `\AddToNoVerbList{`⟨*macros*⟩`}`:

```
\newcommand*{\AddToNoVerbList}{\AddToMacro\niceverbNoVerbList}
```

"Auto mode" probably ain't mean a thing if it ain't invoked using

`\AutoCmdInput{`⟨*file*⟩`}`

for typesetting ⟨*file*⟩ in "auto mode:"

```
\newcommand*{\AutoCmdInput}[1]{%
    \begingroup
      \AddToMacro\niceverbNoVerbList{\ProvidesFile}%
      %% <- removed '\endinput', will be code! 2010/04/05
      \AutoCmdSyntaxVerb
      \input{#1}%
      \EndAutoCmdSyntaxVerb
    \endgroup
}
```

2.2.5 Meta-Variables

`\MetaVar<`⟨*var-id*⟩`>` will be a permanent alias for the active '<'. v0.6 simplifies `\pdfstringdefDisableCommands`.

```
% \def\MetaVar#1>{%
\MakeActive\<
\newcommand*{\MetaVar}{\nvSelfProtect>\nvMetaVar}
\MakeOther\<
\def\nvMetaVar#1>{%
    \mbox{\normalfont\itshape $\langle$#1\/$\rangle$}}
```

As opposed to ltxguide.cls, this works outside verbatim as well. TODO: offer without angles as well

2.2.6 Hash Mark is Code

`\HashVerb`⟨*digit*⟩ will be a permanent alias for the active hash mark.

```
\newcommand*{\HashVerb}[1]{{\tt\##1}}
```

2.2.7 Single Right Quotes for \textsf

\RQsansserif will be a permanent alias for the active single right quote.

One essential problem with the single right quote feature is that a single right quote may be meant to be an apostrophe. This is certainly the case at the right of a letter. On the other hand, we assume that it is *not* an apostrophe (i) in vertical mode (opening a new paragraph), (ii) after a horizontal skip.

Another problem is that with and LaTeX (as with Plain TeX—*The TeXbook* p. 357), the right single quote is needed for primes in math mode, and LaTeX enforces this in \@outputpage preparing \writes (why? TODO) as well as page headers.

```
\MakeActive\'
  \newcommand*{\niceverb_rq_choice}[1]{%                %% 2014/03/27
```

We make a deal with \active@math@prime: in math mode, the prime functionality acts; outside, the "right quote sansserif mode" acts. Test: a' now works with niceverb.—For page headers, in expanding without typesetting, the expansion of \RQsansserif must contain another active single right quote.

```
    \nvSelfProtect'{\ifmmode
                      \expandafter\active@math@prime
                    \else
                      \expandafter#1%
                    \fi}}
```

The following \do_rq_sansserif is what \DoRQsansserif below was before v0.6. This, too, must be changed for \active@math@char, and earlier use of \DoRQsansserif in \niceverb_rq_sf_test must be replaced.

```
  \@ifdefinable\do_rq_sansserif
    {\def\do_rq_sansserif#1'{\textsf{#1}}}
  \newcommand*{\RQsansserif}{%
%   \IfTypesetting{\niceverb_rq_sf_test}{\protect'}}
    \niceverb_rq_choice\niceverb_rq_sf_test}
\MakeOther\'
```

Another macro just to avoid more sequences of \expandafter:

```
\newcommand*{\niceverb_rq_sf_test}{%
  \ifhmode
    \ifdim\lastskip>\z@
%     \expandafter\expandafter\expandafter \DoRQsansserif
      \expandafter\expandafter\expandafter \do_rq_sansserif
    \else
      \ifnum\niceverb_spacefactor
        \expandafter\expandafter\expandafter\expandafter
          \expandafter\expandafter\expandafter
            \do_rq_sansserif %%% \DoRQsansserif
      \else '\fi
    \fi
```

```
    \else \ifvmode
%   \expandafter\expandafter\expandafter \DoRQsansserif
    \expandafter\expandafter\expandafter \do_rq_sansserif
    \else '\fi
  \fi}
% \nvShowProtectedEdef{'niceverb'}
\MakeOther\'
```

`\DoRQsansserif` is *another* (possible) alias for the active single right quote, below.

```
\newcommand*{\DoRQsansserif}{%
    \niceverb_rq_choice\do_rq_sansserif}                %% 2014/03/27
```

The following cases are typical and cannot be decided by the previous criteria: (i) parenthesis, (ii) footnotes and after "horizontal" environments like `\[`⟨*math*⟩`\]`, (iii) section titles, (iv) `\noindent`. We introduce some dangerous tricks—redefinitions of LaTeX's internal `\@sect` and of TeX's primitives `\noindent` and `\ignorespaces` as well as by a signal `\spacefactor` value of 1001.

`\nvAllowRQSS` becomes more powerful with v0.6, for Sec. 2.3.5:

```
\NewSelfProtectedCommand{\nvAllowRQSS}{%
    \MakeActiveLetHere\'\RQsansserif
    \niceverb_rqsf                      %% 2014/03/27
    \niceverb_ignore}                   %% 2010/03/16
```

These and the entire right quote functionality are activated by

`\nvRightQuoteSansSerif` and disabled by `\nvRightQuoteNormal`

—at `\begin{document}`—where we collect previous settings—or later:

```
\AtBeginDocument{%
    \edef\before_niceverb_parenthesis{\the\sfcode`\(}%
    \newlet \before_niceverb_ignore   \ignorespaces     %% 2010/03/16
    \newlet \before_niceverb_sect     \@sect    %% \newlet 2014/03/25
    \newlet \before_niceverb_noindent \noindent}        %% 2010/03/08
```

We assume that `\@sect` has the same parameters there as in LaTeX (even if redefined by another package, like hyperref).

```
\def\niceverb_sect#1#2#3#4#5#6[#7]#8{%
    \before_niceverb_sect{#1}{#2}{#3}{#4}{#5}{#6}%
%                       [{\protect\nvAllowRQSS #7}]%
%                        {\protect\nvAllowRQSS #8}}
```

With v0.6, a more general `\NiceVerbMove{`⟨*text*⟩`}` is introduced, defined in Sec. 2.3.5:

```
                        [\NiceVerbMove{#7}]%
                        {\NiceVerbMove{#8}}}
```

2010/03/20:

```
\newcommand*{\niceverb_spacefactor}{\spacefactor=1001\relax}
\newcommand*{\niceverb_noindent}{%
    \before_niceverb_noindent \niceverb_spacefactor}
\newcommand*{\niceverb_ignore}{%
    \ifhmode \niceverb_spacefactor \fi \before_niceverb_ignore}
```

Here are the main switches. With v0.6, `\nvRightQuoteSansSerif` is divided into two parts, for Sec. 2.3.5:

```
\newcommand*{\niceverb_rqsf}{%                      %% 2014/03/27
%   \MakeActiveLet\'\RQsansserif
    \sfcode`\(=1001   %% enable in parentheses 2009/04/10
```

I also added `\sfcode`/=1001` in the preamble of makedoc.tex.

```
%   \let\@footnotetext\niceverb_footnotetext
    \let\ignorespaces\niceverb_ignore               %% 2010/03/16
%   \let\@sect\niceverb_sect
    \let\noindent\niceverb_noindent}                %% 2010/03/08
\newcommand*{\nvRightQuoteSansSerif}{%
    \niceverb_rqsf
    \MakeActiveLet\'\RQsansserif
    \let\@sect\niceverb_sect
    \def\niceverb_rqsf_kind{\nvAllowRQSS}}
```

← It really must be `\def` in order to transmit the choice to the table of contents.

With v0.6, in dealing with moving things in Sec. 2.3.5, section titles are handled in a more complex way. We divide the former `\nvRightQuoteNormal` into two parts:

```
\newcommand*{\niceverb_rq_normal}{%
%   \MakeNormal\'%                                  %% 2010/03/21
    \sfcode`\(=\before_niceverb_parenthesis\relax
    \let\ignorespaces\before_niceverb_ignore        %% 2010/03/16
    \let\noindent\before_niceverb_noindent}         %% 2010/03/08
\MakeActive\`
\newcommand*{\nvRightQuoteNormal}{%
    \MakeNormal\'%                                  %% 2010/03/21
    \niceverb_rq_normal
    \let\nv_rqsf_kind\@empty
    \ifnum\CatCode\`=\active                %% `=' missing 2015/11/09
      \ifx`\LQverb \else
        \let\@sect\before_niceverb_sect
      \fi
    \else
        \let\@sect\before_niceverb_sect
    \fi}
\MakeOther\`
```

`\nvAllRightQuotesSansSerif` (after `\begin{document}`!) forces the `\textsf` feature *without* testing for apostrophes. You then must be sure—DANGER! CARE!—to use '`\rq`' only for obtaining an apostrophe and the double quote character '`"`' for closing double quotes, or our `\dqtd{`⟨*text*⟩`}` for the entire quoting.

```
\newcommand*{\nvAllRightQuotesSansSerif}{%
    \niceverb_rq_normal %%% \nvRightQuoteNormal          %% 2014/03/27
```

That's one use of `\DoRQsansserif` with v0.6:

```
    \MakeActiveLet\'\DoRQsansserif
    \def\niceverb_rqsf_kind{\nvAllRQSS}}                 %% 2014/03/27
```

← must be `\def` for transmissions.

```
\NewSelfProtectedCommand{\nvAllRQSS}{%
        \niceverb_rq_normal
```

That's the other use of `\DoRQsansserif` with v0.6:

```
        \MakeActiveLetHere\'\DoRQsansserif}
```

[Hiding remarks from 2010f. (`\ctanpkgref`) 2014/03/23]

2.2.8 Boxes Highlighting Commands and Syntax

With v0.3, we include one kind of command syntax boxes whose ⟨*content*⟩ is (in niceverb syntax) delimited as `|`⟨*content*⟩`|`. `\GenCmdBox`⟨*char*⟩⟨*content*⟩⟨*char*⟩ works like `\NVerb`⟨*char*⟩⟨*content*⟩⟨*char*⟩ except putting the latter's result into a framed (or coloured or . . .) box.

```
\newcommand*{\GenCmdBox}  {\_no_nice_meta_verb_false \gen_cmd_box}
```

`\HardVerbBox` is a variant of `\GenCmdBox` with the meta-variable feature disabled (for the documentation of the present package).

```
\newcommand*{\HardVerbBox}{\_no_nice_meta_verb_true  \gen_cmd_box}
\newcommand*{\gen_cmd_box}{%
% \let\niceverb_egroup\nice_collect_verb_egroup %% rm 2014/03/18
% \setbox\niceverb_savebox \hbox\bgroup
```

← 2014/03/19 →

```
  \SetNiceVerbSaveBox
%     \if_no_nice_meta_verb_
%           \expandafter \HardNVerb
%     \else \expandafter \NVerb     \fi
```

← 2014/03/19 → [TODO use generalization]

```
334       \NiceMaybeMetaVerb[\nice_collect_verb_egroup]%
335   }
336   \newcommand*{\nice_collect_verb_egroup}{%
337       \egroup \egroup
338     \ifvmode \expandafter \VerticalCmdBox
339     \else    \ifmmode \hbox \fi
340              \expandafter \InlineCmdBox \fi
341   %               {\box\niceverb_savebox}%
```

← 2014/03/19 →

```
342                   \TheNiceVerbSaveBox
```

(Removing a remark that I don't understand 2014/03/19.)

```
343     \ifmmode\else\@\fi
344   % \let\niceverb_egroup\niceverb_normal_egroup %%     rm 2014/03/19
345   }
```

`\nvCmdBox` will be the permanent alias for '`|`'.

```
346   \newcommand*{\nvCmdBox}{\GenCmdBox\|}
```

`\VerticalCmdBox{`⟨*content*⟩`}` may eventually start a `decl` environment as in ltxguide.cls, looking ahead for another '`|`' in order to (perhaps) append another row. Another possibility is first to do some

```
\if@nobreak\else␣\pagebreak[2]\fi
```

etc. and then invoke `\InlineCmdBox`. The user can choose later by some `\renewcommand`. We do the perhaps most essential thing here (again cf. `\begin_min_verb`):

```
347   \newcommand*{\VerticalCmdBox}{%
```

v0.6 encourages a page break here according to the above idea, in order to avoid a page break after explaining subsequent code (TODO: that's a major functionality change):

```
348                 \if@nobreak\else \pagebreak[2]\fi
349                 \leavevmode\InlineCmdBox}
```

(2011/11/05 removing `\null`.) The command declaration boxes in the documentation of Nicola Talbot's datatool would be an especially nice realization of `\VerticalCmdBox`.[13]

`\InlineCmdBox{`⟨*content*⟩`}`, according to our idea, should not change baseline skip, even with some `\fboxsep` and `\fboxrule`. (However, it may be a good idea to increase the overall normal baseline skip.) We therefore replace actual height and depth of the content by the height and depth of math parentheses.

[13] I find the documentation of Martin Scharrer's newverbs package similarly impressive.

```
\newcommand*{\InlineCmdBox}[1]{%
  \bgroup
```

... needed in math mode with `\begin_min_verb`.

```
    \fboxsep 1pt
    \kern\SetOffInlineCmdBoxOuter
    \smash{\SetOffInlineCmdBox{\kern\SetOffInlineCmdBoxInner
                               \InlineCmdBoxArea{#1}%
                               \kern\SetOffInlineCmdBoxInner}}%
    \mathstrut
    \kern\SetOffInlineCmdBoxOuter
  \egroup
}
```

The default choice for `\SetOffInlineCmdBox` is `\fbox`:

```
\newlet\SetOffInlineCmdBox\fbox
```

You can `\renewcommand` it to change `\fboxsep`, `\fboxrule` etc. or to use a `\colorbox` with the color package, e.g., I used the following setting so far:

```
\RequirePackage{color}
\renewcommand*{\SetOffInlineCmdBox}
              {\colorbox[cmyk]{.1,0,.2,.05}}
```

`\SetOffInlineCmdBoxInner` enables controlling the inner horizontal space to the box margin independently of `\fboxsep`.

```
\newcommand*{\SetOffInlineCmdBoxInner}{-\fboxsep\thinspace}
```

This choice is inspired by `\cstok` for "boxed" things in Knuth's manmac.tex which formats *The TEXbook*.

`\SetOffInlineCmdBoxOuter` allows that the box hangs out into the margin horizontally. We set it to 0 pt as default (it is a macro only, for a while).

```
\newcommand*{\SetOffInlineCmdBoxOuter}{\z@}
```

The height and depth of the frame should be the same for all inline boxes, we think. The present choice `\InnerCmdBoxArea` for the spacing respects code characters rather than the height and depth of the angle brackets that surround meta-variable names.

```
\newcommand*{\InlineCmdBoxArea}[1]{%
    \smash{#1}\vphantom{gjpq\backslash_verb}}
```

`\cmdboxitem|`⟨*content*⟩`|` is another variant of `\GenCmdBox`. It should replace `\item[`⟨*content*⟩`]` in the `description` environment.

```
\newcommand*{\cmdboxitem}{%
% \bgroup
%   \let\niceverb_egroup\cmd_item_egroup
%   \global %% TODO!? 2010/03/15
%   \setbox\niceverb_savebox \hbox\bgroup
```

← 2014/03/19 →

```
371       \SetNiceVerbSaveBox
372   %     \NVerb}
```

← 2014/03/19 →

```
373         \NVerb[\cmd_item_egroup]}
374   \newcommand*{\cmd_item_egroup}{%
375         \egroup \egroup %%% \egroup                %% 1 less 2014/03/19
376     \item[\InlineCmdBox\TheNiceVerbSaveBox]}
```

Does it work?

`\foo{`⟨*arg*⟩`}` could be defined for a test.

`\bar{`⟨*arg*⟩`}` could be defined for a test as well.

2.3 When niceverb Gets Nasty

These things are new with v0.3.

2.3.1 Meta-Variables

This is even newer than v0.3.

In case you actually need < and > in math mode, `\lt` and `\gt` are "provided" as aliases:

```
377   \providecommand*{\gt}{>}
378   \providecommand*{\lt}{<}
```

2.3.2 Quotes

In order to get *real* single quotes, you could use `\lq`␣⟨*text*⟩`\rq`, maybe appending a `\`␣, but the code `\qtd{`⟨*text*⟩`}` may look better and be easier to type.

```
379   \providecommand*{\qtd}[1]{‘#1’}              %% provide 2012/11/27
```

However, here we get the problem that the left quote in `\qtd{‘`⟨*code*⟩`’}` will be unable to switch into verbatim mode entirely—then use `&`, e.g., '`\qtd{&&}`' typesets "`&`", i.e., the ampersand in single (non-verbatim) quotes.

```
380   % todo \qtdverb!? alternative meaning for \LQverb!? 2010/03/06
381   %      rather rare, & takes less space              2010/03/09
```

. . . see approaches below . . .

`\AddQuotes` automatically surrounds code with single quotes. I have so often felt that it was a design mistake to drop them (2011/09/09):

```
382   \newcommand*{\AddQuotes}{%
383       \let\niceverb_maybe_qs\niceverb_add_qs}
384   \newcommand*{\niceverb_add_qs}{%
```

In a math display, quotes are suppressed even with `\AddQuotes`:

```
385       \ifmmode\else
386         `\let\niceverb_maybe_rq\niceverb_rq
387       \fi}
388   \newlet\niceverb_maybe_rq\relax
389   \newcommand*{\niceverb_rq}{'\let\niceverb_maybe_rq\relax}
```

You can undo this by `\DontAddQuotes`:

```
390   \newcommand*{\DontAddQuotes}{\let\niceverb_maybe_qs\relax}
```

The default will be the behaviour that we had before:

```
391   \DontAddQuotes
```

With v0.6, `\qtdnverb`⟨*char*⟩⟨*m-verb*⟩⟨*char*⟩ encloses the "meta-verbatim" material with single quotes:

```
392   \newcommand*{\qtdnverb}{%
```

Don't ... after `\AddQuotes`:—TODO

```
393     \ifx\niceverb_maybe_qs\niceverb_add_qs
394       \expandafter\NVerb
395     \else
396       \lq
397   %     \expandafter\NVerb\expandafter[\expandafter\egroup\expandafter]
398     \fi}
```

`\dqtd{`⟨*text*⟩`}` can be used for enclosing in *double* quotes with the dangerous `\nvAllRightQuotesSansSerif` (see above).

```
399   \providecommand*{\dqtd}[1]{``#1"}                    %% 2012/11/27
```

2.3.3 hyperref

This is for/about compatibility with the hyperref package. (One preliminary thing: in doubt, don't load niceverb earlier than hyperref.)

We need some substitutions for PDF bookmarks with hyperref. We issue them at `\begin{document}` when we know if hyperref is at work.[14]

```
400   \AtBeginDocument{%
401     \@ifpackageloaded{hyperref}{%
402       \newcommand*{\PDFcstring}{%         %% moved here 2010/03/09
403         \134\expandafter\@gobble\string}% %% ASCII octal encoding
404       \pdfstringdefDisableCommands{%
405         \let\nvAllowRQSS\empty            %% not \relax 2010/03/12
406         \let\NiceVerbGeneral\empty        %% 2014/03/27
407         \let\nvAllRQSS\empty              %% 2014/03/27
408         \MakeActiveLetHere\<<%            %% 2014/03/28
```

[14] An alternative approach would be using afterpackage by Alex Rozhenko.

```
409         %% 2010/03/12
410         \MakeActiveLetHere\`\lq \MakeActiveLetHere\'\rq
411         \MakeActiveLetHere\&\PDFcstring
412         \def\cs{\134}%                    %% 2010/03/17, 2011/06/27
```

The typesetting version of `\BuildCsSyntax` (Sec. 2.2.4): 2014/07/16

```
413         \withcsname\def BuildCsSyntax \endcsname{\cs}%
```

... disables `\niceverbNoVerbList`; better switch off auto mode with section headings TODO (modify `\@startsection`)

```
414         \let\decide_more_cs\bookmark_more_cs
415       }%
```

Moreover, in order to avoid spurious `Label(s) may have changed` with hyperref, a single right quote must be *read* as active by a `\newlabel` if and only if it has been active when `\@currentlabelname` was formed.[15] as `\active`. We use `\protected@write` as this cares for `\nofiles`. `\@auxout` may be `\@partaux` for `\include`.

```
416       \newcommand*{\niceverb_aux_cat}[2]{%                %% 2010/03/14
417           \protected@write\@auxout{}{\string#1\string#2}}%
```

v0.5 restricts "activating" to `\MakeActiveLet`:

```
418   %     \renewcommand*{\MakeActive}[1]{%
419   %         \MakeActiveHere#1%
420   %         \niceverb_aux_cat\MakeActiveHere#1}%
421       \renewcommand*{\MakeActiveLet}[2]{%
422           \MakeActiveLetHere#1#2%
423   %         \niceverb_aux_cat\MakeActiveHere#1}%
424           \protected@write\@auxout{}{%
425               \string\MakeActiveLetHere\string#1\string#2}}%
426       \renewcommand*{\MakeNormal}[1]{%
427           \MakeNormalHere#1%
428           \niceverb_aux_cat\MakeNormalHere#1}%
429     }{}%
430   }
```

2.3.4 hyper-xr

With the hyper-xr package creating links into external documents, preceding `\externaldocument{`⟨*file*⟩`}` with `\MakeActiveLet\&\CmdSyntaxVerb` may be needed. I do not want to redefine something here right now as I have too little experience with this situation.

[15] This uses `\@onelevelsanitize`, therefore `\protect` doesn't change the behaviour of "active" characters.

2.3.5 Listings and Moving

Working on v0.6, in testing I discovered a problem with the *listing environments*. The present documentation uses code listings with makedoc, which build on the moreverb package and eventually call LaTeX's `\@noligs` macro.[16] The problem also appears with the `{verbatim}` environment from the LaTeX kernel (latex.ltx) as well as from the verbatim package—with anything that calls LaTeX's `\@noligs`. The latter assigns special meangings to the active characters listed in the `\verbatim@nolig@list`, three of them need a different meaning with niceverb. When a page break happens after such an environment has been entered (this may well be when the environment falls to the beginning of the next page), these settings are used in LaTeX's `\@outputpage` for running the `\write`s of the page as well as for page headers. And this happens quite often in a package documentation!

The problem was reported by Walter Schmidt with respect to math primes as `latex/3104` in 1999. I cannot reproduce it, and I see two reasons in recent `latex.ltx` code why it cannot happen anymore. However, one remedy in `latex.ltx` is activating `\active@math@prime` in `\@resetactivechars` of `\@outputpage`. But this is bad for niceverb's single right quote. We override the `\active@math@prime` functionality and verbatim `\@noligs` by appending some protection of the characters collected in `\verbatim@nolig@list` to `\@resetactivechars`. This solves the problem for `\write`s at `\shipout`. `\do_protect_noligs` is used for this purpose; actually it is applied in `\useNiceVerb` (Sec. 2.3.6):

```
\newcommand*{\do_protect_noligs}[1]{%                    %% 2014/03/28
    \MakeActiveLetHere#1\relax}          %% 'Here' missing 2015/11/09
```

`\nvResetPages` can be used to restore LaTeX's `\@resetactivechars`. I don't add it to `\noNiceVerb` because it could corrupt `\write`s, so should be used with care.

```
\AtBeginDocument{%
    \newlet\latex_reset_actives\@resetactivechars}
\newcommand*{\nvResetPages}{%
    \let\@resetactivechars\latex_reset_actives}
```

`\NiceVerbMove{`⟨*text*⟩`}` with v0.6 is for "moving" arguments so that niceverb syntax operates *locally* at the destination, I think of table of contents and page headers. It is automatically used by niceverb's variant of LaTeX's sectioning commands (Sec. 2.2.7); while with `\markboth`, `\markright`, `\addcontentsline` etc. you must it include yourself (currently, TODO?). This is meant as a remedy against LaTeX's and verbatim's `\@noligs` with respect to page headers. However, another purpose is that you could switch off the niceverb syntax at the beginning of your document (by `\noNiceVerb`), though certain entries to the table of contents can use niceverb syntax without affecting other entries (where

[16] moreverb is used, and its listing environments use `\@verbatim` from the verbatim package, then `\verbatim@font` calls `\@noligs` ...

some active characters may have different meanings, perhaps from a different package).

```
\newcommand*{\NiceVerbMove}[1]{%
```

What goes to `.aux` files must not have underscores:

```
    {\NiceVerbGeneral\niceverb_rqsf_kind#1}}
\NewSelfProtectedCommand{\NiceVerbGeneral}{%
% \newcommand*{\useNiceVerbHere}{%                    %% 2014/03/28
    \let\MakeActiveLet\MakeActiveLetHere \useNiceVerbI}
% \newcommand*{\NiceVerbGeneral}{%
%     \nvSelfProtect\NiceVerbGeneral\useNiceVerbHere}
```

2.3.6 Turning off and on altogether

These commands are new with v0.3.

`\noNiceVerb` *disables* all niceverb features.

```
\newcommand*{\noNiceVerb}  {\MakeNormal\`%
                            \MakeNormal\&%
                            \MakeNormal\<%
                            \MakeNormal\#%
                            \nvRightQuoteNormal
                            \MakeNormal\|%
                    \let\@sect\niceverb_before_sect}      %% 2014/03/27
```

`\useNiceVerb` *activates* all the niceverb features (apart from "auto mode"). With v0.6, it is divided into two parts for `\NiceVerbMove` in Sec. 2.3.5:

```
\newcommand*{\useNiceVerbI}{\MakeActiveLet\`\LQverb
```

TODO to be changed with wiki.sty v0.2

```
                            \MakeActiveLet\&\CmdSyntaxVerb
                            \MakeActiveLet\<\MetaVar
                            \MakeActiveLet\#\HashVerb
                            \nvRightQuoteSansSerif
                            \MakeActiveLet\|\nvCmdBox}
\newcommand*{\useNiceVerb} {\useNiceVerbI               %% 2014/03/27
                            \let\@sect\niceverb_sect
    \g@addto@macro\@resetactivechars{%
%                           \useNiceVerbHere            %% 2014/03/28
    \let\do\do_protect_noligs \verbatim@nolig@list      %% 2014/03/28
}}
```

2.4 Minor Final Things

2.4.1 Activating the niceverb Syntax

niceverb features are activated at `\begin{document}` so (some) other packages can be loaded *after* niceverb. For v0.3, we do this after possible settings for compatibility with hyperref.

```
\AtBeginDocument{\useNiceVerb}
```

2.4.2 Leave Package Mode

```
\PopLetterCat\_                                          %% 2012/08/27
\endinput
```

2.4.3 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
       2009/04/11   \@ after \SimpleVerb
       2009/04/14   noted TODO below
       2009/04/15   change v0.1 to 2009/02/21
v0.30  2010/02/27   short, more explained, \AssignCatCodeTo,
                    use \MakeActive for re-activating, \MakeNormal
       2010/02/28   fixed @ and _ with & by moving \begin_min_verb;
                    replaced \lq by `; Capitals in Titles
       2010/03/05   \SimpleVerb -> \NVerb;
                    use \MakeActive + \MakeNormal; \rq -> ';
                    renamed some sections; \lq_verb -> \LQverb,
                    \niceverb_meta -> \MetaVar,
                    \param_verb -> \HashVerb
       2010/03/06   removed \MakeAlign; removed @ and _ todo below;
                    \NVerb makes ` and ' other;
                    \nvAllowRQSF allows ' in column titles,
       2010/03/08   \LQverb and & work in column titles,
                    \RQverb works with \noindent;
                    bookmark substitutions
       2010/03/09   extended notes on 'hyperref' (in)compatibility;
                    \MakeLetter\@ in \CmdSyntaxVerb only;
                    |...| implemented as \prepareCmdBox etc.!
       2010/03/10   \colorbox example, \thinspace; ltxguide!;
                    removed todo; ..._exec -> \DoRQsansserif;
                    minor doc changes in ``Nasty"
       2010/03/11   doc changes in ``Escape Character ..." and
                    ``Ampersand"
       2010/03/12   \niceverb_aux_cat, \MakeActiveHere etc.,
                    \IfTypesetting, \noNiceVerb, \useNiceVerb,
                    corr. bracing mistake in \MakeNormal!
       2010/03/14   0.31 -> 0.3; \HardNVerb, \GenCmdBox,
                    \prepareCmdBox -> \nvCmdBox
       2010/03/15   \endcell; \cmdboxitem; remark on \sfcode`/
       2010/03/16   corr. -> \endline;
```

```
507                     advice on \cs{par}, \cs{if...}, \cs{fi};
508                     redefined \ignorespaces for RQ feature
509         2010/03/17  corr. `\fututelet', corr. \cs PDF substitution
510         2010/03/18  |\niceverbNoVerbList|, |\AddToMacro| etc.;
511                     corr. \AddToMacro;
512                     \lastskip-fix of \niceverb_ignore,
513                     another fix of \niceverb_noindent
514         2010/03/19  another fix of \niceverb_ignore: \spacefactor
515         2010/03/20  ... again: \niceverb_spacefactor
516
517   NOT DISTRIBUTED, just stored saved as separate version
518
519   v0.31  2010/03/20  right quote feauture: letters get \sfcode=1001
520                     `column title' -> `page headers', \ctanpkgref
521
522   NOT DISTRIBUTED, just stored as separate version
523
524   v0.32  2010/03/21  taking best things from v0.30 and v0.31
525          2010/03/23  removed \relax from \IfTypesetting
526   SENT TO CTAN
527
528   v0.4   2010/03/27  restoring `_' with "auto mode" safer
529          2010/03/28  \AddToNoVerbList
530          2010/03/29  note above, renamed v0.4
531   SENT TO CTAN
532
533   v0.41  2010/04/03  v0.33 -> v0.4
534          2010/04/05  corrected \AutoCmdInput list
535   SENT TO CTAN as part of NICETEXT release r0.41
536
537   v0.41a 2010/11/09  typo corrected
538   v0.42  2010/12/30  corr. `\ ' emulation in auto mode
539          2010/12/31  \MetaVar in ...maybe_meta...
540          2011/01/19  `...' fix
541          2011/01/24  \ctanpkgref moves to texlinks.sty
542          2011/01/26  update (C)
543   with nicetext RELEASE r0.42
544   v0.43  2011/05/09  \gt, \lt
545          2011/05/27  \cs uses \@backslashchar
546          2011/06/20  \MakeActiveLetHere in \nice_maybe_meta_verb !!!
547          2011/06/27  2011/05/27 undone
548          2011/08/20  `r0.42', `v0.43'
549   with nicetext RELEASE r0.43
550   v0.44  2011/09/09  \AddQuotes, \DontAddQuotes
551   with nicetext RELEASE r0.44
552   v0.45  2011/11/05  mod. \niceverb_collect_egroup/\VerticalCmdBox,
553                     tried \output problem without avail
554          2011/12/05  clarified "r0.44"
555   with nicetext RELEASE r0.5
556
```

```
v0.5   2012/08/27   using 'catcodes', \providecommand\CatCode,
                    rm. \AssignCatCodeTo, \private_letters
       2012/08/28   fixed \private_letters;
                    rewording for filling lines
       2012/09/27   corrections about \MakeActive...
with nicetext RELEASE r0.6

v0.6   2012/11/27   \[d]qtd only \provide'd
v0.61  2014/03/18   doc.: rm. TODO on private letters hook,
                    folding history tighter,
                    RM CODE COMMENTED OUT IN 2011;
                    \VerticalCmdBox gets \pagebreak[2]
       2014/03/19   doc.: strange replaced, restructured,
                    Command-Highlighting Boxes -> Boxes
                    Highlighting ...;
                    opt. arg. for \NVerb etc. replaces
                    \niceverb_egroup, \cs/\cstx enhanced,
                    reimpl.s with \SetNiceVerbSaveBox,
                    \nice_maybe_meta_verb -> \NiceMaybeMetaVerb
       2014/03/20   reworking robustness -- doc., ...; doc. on
                    'Shared ...', \qtdnverb
       2014/03/21   "debugging": \noexpand vs. \protect
       2014/03/22   ... continued; mod. \MakeNormal
       2014/03/23   ..., hiding ...
       2014/03/24   TODO on left quotes, doc. test there,
                    rm. babel-TODO
       2014/03/25   doc. about left quotes shorter, rm. earlier page
                    breaks, doc. problems with right quotes; \newlet;
                    dealing with \active@math@prime
       2014/03/26   corr. test for right single quote, more about
                    \active@math@prime, corr. \CmdSyntaxVerb
       2014/03/27   different treatment of \active@math@prime ...
                    main work for sec:listmv and independent
                    switching for rqsf, \NewSelfProtectedCommand
                    (3 applications); doc. corr., TODO;
                    \typeout test
       2014/03/28   test section; [TODO?]; \nvShowProtectedEdef,
                    \MetaVar: protection and hyperref version;
                    remark \NVerb; [\NiceVerbHere]; \protect test
                    with RQ removed; \do_protect_noligs fixed
                    and used
       2014/07/16   \BuildCsSyntax with hyperref
       2014/07/17   remarks on \BuildCsSyntax and removing useless
                    braces there (hours of trying better)
       2015/02/23   doc.: \PDFstring -> \pdfstring
       2015/04/07   doc.: page headings -> page headers
       2015/11/09   bugfixes \nvRightQuoteNormal and
                    \do_protect_noligs; doc. typo fix
```