Menu.sty: Typesetting menus

Knut Lickert

January 28, 2009

Abstract

Menu.sty define a command `\menu{Programm!!Menu1!Command}` and expand it to something like `Programm⇒Menu1→Command`. You can use it for documentations of programs.

1 Where to use the style

If you write a documentation of a program you have to define things like "Push button [F1] or go to the Menu to `Help→Help` and you get help". This style help you to write the menu entry.

1.1 Other packages

I have not found a package to typeset menus (that's why this style was written). If you find one, please contact me (knut@lickert.net).

keystroke A LaTeX package which provides macros for the graphical representation of the keys on a computer keyboard.

Example: [Ctrl] , [Shift ⇑] [AltGr]

2 How to use the style

2.1 Inline-Menu

\menu `\menu` has one parameter. The macro is used to typeset a path in a menu. Like `\index` different levels are separated with a "!" inside the parameter. Instead a "!" you can use two "!!" and so you can define bigger steps in the menu. I use programs, where a menu is starting another menu (e.g. in a popup). So instead of saying "Goto `Menu1→Entry1→Menu2` and then from `Menu2→Menu3→Entry3`" I write "Goto `Menu1→Entry1→Menu2⇒Menu3→Entry3`.

Example: `\menu{File!Save!!Filename}`: `File→Save⇒Filename`

This is only to show one path in a menu. This command may not be used to show a complete menu. I recommand to make a snapshot of your programm and include the picture via `includegraphics` (Package graphicx)

2.2 Menu-Figure

menufolder   Number of parameter: 2

\menuitem   Number of parameter: 2

\menuitemactive   Number of parameter: 2

\menuseparator   Add a separator line. An optional length can be defined.

Example:

```
%\settowidth{\menuitemlength}{Close and d\underline on't save}
%\begin{menufolder}{\underline Menu}%
%    \menuitem{Close and d\underline on't save}
%    \menuitem{Close and sa\underline ve}
%    \menuitemactive{I\underline tem}
%\end{menufolder}
%}
%
```

M̲enu

     Close and d̲o̲n't save

     Close and sa̲v̲e

     I̲t̲em   ⟸

2.2.1 Options

There are different options to modify the look of menus.

hand The hand point to the active menu item. Requires bbding.

framed The menu entries are framed

grey framed and background values.

3 Changing the look

\menutext   A step of a menu is written with this macro, default is \texttt. You can change it with \renewcommand, e.g. \renewcommand{\menutext}[1]{\emph{#1}}

3.1 Inline-Menus

\menumathsymbols   If you want to replace the symbols, you can use \menusymbols. Parameter one is the flag for "!", Parameter for "!!" (Default is → and ⇒).

\menusymbols   Like \menumathsymbols with two differences:

- There is no math-environment (if you need it, you can guarantee it with \ensuremath)

- There are two optional parameters, defining a start and end command of a menu.

This command gives you a flexible possibility to change the look of menus.

menusymbols Define a style with the parameters:

[**Start**] How should a menu start.

> A \par is not allowed, but you can use a \\. If a paragraph ends before the \menu, you must use ~\\ instead.

{} The look/actions for a "!" in a menu.

{} The look/actions for a "!!" in a menu.

[**End**] How should a menu end.

3.1.1   Examples

Predefined look

Coding	Result
\menu{Menu!Menu}	Menu→Menu
\menu{Menu!!Menu}	Menu⇒Menu
\menu{Menu!!Next Menu!Entry}	Menu⇒Next Menu→Entry

Changed symbols   After \menumathsymbols{>}{\gg} you get

Coding	Result
\menu{Menu!Menu}	Menu>Menu
\menu{Menu!!Menu}	Menu≫Menu
\menu{Menu!!Next Menu!Entry}	Menu≫Next Menu>Entry

Attention: If you want to change it local, you must define a block.

Define your own styles   You can define your own "styles", \menu{Menu!!Next Menu!Entry} makes an entry like:

```
%\newlength{\menusep}            Example for a new style
%\menusymbols[                   Menu ⟨
%   \setlength{\menusep}{0em}~\\      Next Menu
% ]{                                      Entry ⇐
%   \\
%   \addtolength{\menusep}{1em}
%   \hspace*{\menusep}
% }{
%   $\wr$\\
%   \addtolength{\menusep}{2em}
%   \hspace*{\menusep}
% }[
% \ensuremath{\Leftarrow}
% ]
%
```

3.2 Layout of Menu-figures

The macros can be replaced. See the implementation for examples.

\menufolderentry   Defines the look of the entry of a menu. Parameters are length (optional) and text.

\menuitem   Defines the look of one item in the menu. Parameters are length (optional) and text.

\menuitemactive   Defines the look of an active item in the menu. Parameters are length (optional) and text.

\menuitemactivesymbol   This symbol is used to show to the active item.

3.2.1 Examples

4 Ideas for the future

- Create Index entries for menus (optional)

  If I do it, it would be an additional index (style multind). I think this would be wrong in the normal index.

- An optional parameter: \menu[Programm]{Menu!...}

  Does anybody need it? At the moment I think there is no need. When there is the possibility of index entries, then it could be sensefull (Index is sorted by Programm, Menu1, Menu2...).

- A command \popup or \rightmouse to define a click on the right mouse button. \click[Feld]{menu!entry}. This function could be a feature of the package keystroke.

Any comments? Up to now I don't need this features, so I will not implement them. If you need it, or think you have a better idea, send me a mail to knut@lickert.net.

If you built your own "fancy" style, or have another idea to use "!!" and "!", please send me your idea.

Implementation

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{menu}[
3     \filedate\space\fileversion\space menu-Package]
4 \typeout{menu.sty: Support of typeout menus}
5 \RequirePackage{xspace}
6 %\RequirePackage{keystroke}[2003/08/15 v1.5]
```

5 Commands

\menu   Take the parameter and send it with an endflag ("<") to \@menu.

```
7 \newcommand{\menu}[1]{\@menusymbolStart\@menu #1!<\@menusymbolEnd}
```

\menutext   How to write one step of the menu?

```
8 \newcommand{\menutext}[1]{\texttt{#1}}
```

\@menu   Get two parameters, separated with "!". Parameter one can be:

```
9 \def\@menu#1!#2<{%
10 \def\@menuParOne{#1}%
11 \def\@menuParTwo{#2}%
12 \def\gt{!}%
```

Check if the first parameter has content.

```
13 \ifx\@empty\@menuParOne%
14 {\ifx\@menuParTwo\gt%
15  \else\expandafter\@menuTo\@menuParTwo <\@menu#2<%
16  \fi}%
17 \else%
```

First parameter has content, so let's print it.

```
18 \menutext{#1}%
```

Add a \@menuto and the second part, but only if there is a second part.

```
19  \ifx\@menuParTwo\@empty%
20  \else%
21   \expandafter\@menuto\@menuParTwo <\@menu#2!<%
22  \fi%
23 \fi%
24 }
```

\@menuto

```
25 \def\@menuto#1#2<{\ifx!#1\else\@menusymbolOne\allowbreak\fi}
```

\@menuTo

```
26 \def\@menuTo#1#2<{\ifx!#1\else\@menusymbolTwo\allowbreak\fi}
```

6   The different symbols

\@menusymbolStart   The symbol, printed on start of \menu.

```
27 \newcommand*\@menusymbolStart{}
28 \newcommand*\@menusymbolEnd{}
```

\@menusymbolOne   The symbol, printed when a "!" occurs in \menu.

```
29 \newcommand*\@menusymbolOne{\ensuremath{\rightarrow}}
```

\@menusymbolTwo   The symbol, printed when a "!!" occurs in \menu.

```
30 \newcommand*\@menusymbolTwo{\ensuremath{\Rightarrow}}
```

<dl>
<dt>\menusymbol</dt>
<dd>

Possibility to redefine the symbols for "!" and "!!".

```
31 \newcommand*\menusymbols[3][]{%
32 \renewcommand*\@menusymbolStart{#1}%
33 \renewcommand*\@menusymbolOne{#2}%
34 \renewcommand*\@menusymbolTwo{#3}%
35 \@menusymbols%
36 }
```
</dd>

<dt>\@menusymbol</dt>
<dd>

Define the End-Makro for a menu.

```
37 \newcommand*\@menusymbols[1][]{%
38 \renewcommand*\@menusymbolEnd{#1}%
39 }
```
</dd>

<dt>\menumathsymbols</dt>
<dd>

Possibility to redefine the symbols for "!" and "!!".

```
40 \newcommand*\menumathsymbols[3][]{
41 \renewcommand*\@menusymbolStart{\ensuremath{#1}}
42 \renewcommand*\@menusymbolOne{\ensuremath{#2}}
43 \renewcommand*\@menusymbolTwo{\ensuremath{#3}}
44 }
```
</dd>
</dl>

7 The Predefined Layouts

7.1 Standard

```
45 \menumathsymbols{\rightarrow}{\Rightarrow}
```

8 Menu-Drawings

```
46 \newlength{\menuitemlength}
47 \newlength{\menusep}
48 \setlength{\menuitemlength}{6em}
49 \setlength{\menusep}{3em}
```

<dl>
<dt>\menuitemactivesymbol</dt>
<dd>

```
50 \newcommand\menuitemactivesymbol{\ensuremath{\Leftarrow}}%
```
</dd>
</dl>

FiXme: *parameter 1 must be a length (2cm result* `\menufolder`*)*

A negative parskip results in an overlapping menu.

```
51 \newenvironment{menufolder}[2][\menuitemlength]{%
52     \begin{minipage}{2#1}%
53     \setlength{\parskip}{0pt}%
54     \menufolderentry[#1]{#2}\par%
55     \addtolength{\leftskip}{\menusep}%
56     }{%
57     \end{minipage}%
58 }
```

<dl>
<dt>\menufolderentry</dt>
<dd>

```
59 \newcommand\menufolderentry[2][\menuitemlength]{%
60     \makebox[#1][l]{\menutext{#2}\hfill}%
61 }
```
</dd>
</dl>

```
62 \newcommand\menuitem[2][\menuitemlength]{%
63     \makebox[#1][l]{\menutext{#2}\hfill}\par%
64 }
```

```
65 \newcommand\menuitemactive[2][\menuitemlength]{%
66     \makebox[#1][l]{\menutext{\emph{#2}}\hfill\menuitemactivesymbol}\par%
67 }
```

```
68 \newcommand\menuseparator[1][\menuitemlength]{%
69     \makebox[#1][l]{~\hrulefill~}\par%
70 }
```

8.1   Options

```
71 \newif\ifmenuoptionhand
72 \DeclareOption{hand}{
73 \menuoptionhandtrue
74 }

75 \newif\ifmenuoptionframed
76 \DeclareOption{framed}{
77 \menuoptionframedtrue
78 }

79 \newif\ifmenuoptiongrey
80 \DeclareOption{grey}{
81 \menuoptiongreytrue
82 }
```

Process the options to set the flags.

```
83 \ProcessOptions\relax
```

8.1.1   Option hand

Activate the hand-option.

```
84 \ifmenuoptionhand
85 \RequirePackage{bbding}%fuer  \HandLeftUp
```

```
86 \renewcommand\menuitemactivesymbol{\HandLeftUp}%

87 \fi%ifmenuoptionhand
```

8.1.2   Option framed

Activate the framed-option.

```
88 \ifmenuoptionframed
89 \RequirePackage{fancybox}%fuer shadowbox
```

\menufolderentry

```
90 \renewcommand\menufolderentry[2][\menuitemlength]{%
91     \framebox[#1][l]{\menutext{#2}\hfill}%
92 }
```

\menuitem

```
93 \renewcommand\menuitem[2][\menuitemlength]{%
94     \framebox[#1][l]{\menutext{#2}\hfill}\par%
95 }
```

\menuitemactive

```
96 \renewcommand\menuitemactive[2][\menuitemlength]{%
97     \shadowbox to #1{\menutext{\emph{#2}}\hfill\menuitemactivesymbol}\par%
98 }
```

```
99 \fi%Option framed
```

8.1.3 Option grey

Activate the grey-option.

```
100 \ifmenuoptiongrey
101 \RequirePackage{fancybox}%fuer shadowbox
102 \RequirePackage{color}%fuer farbige Boxen
103 \definecolor{menugrey}{rgb}{0.9,0.9,0.9}
```

\menufolderentry

```
104 \renewcommand\menufolderentry[2][\menuitemlength]{%
105     \fcolorbox{white}{black}{\makebox[#1]{\menutext{\textcolor{white}{#2}}}}\par%
106 }
```

\menuitem

```
107 \renewcommand\menuitem[2][\menuitemlength]{%
108     \fcolorbox{black}{menugrey}{\makebox[#1]{\menutext{\textcolor{white}{#2}\hfill}}}\par%
109 }
```

FiXme: *offset 1ex*
\menuitemactive

```
110 \renewcommand\menuitemactive[2][\menuitemlength]{%
111     \textsf{\hspace*{1ex}\shadowbox to #1 {\menutext{#2}\hfill\menuitemactivesymbol}}\par%fixme
112 }
```

FiXme: *verbessern*
\menuseparator

```
113 \renewcommand\menuseparator[1][\menuitemlength]{%
114     \makebox[#1][l]{~\hrulefill~}\par%
115 }
```

```
116 \fi%Option grey
```