Menu.sty: Typesetting menus

Knut Lickert

March 6, 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 Related 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).

There are related packages which may help to build program documentations:

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

Example: Ctrl , Shift ⇑ AltGr

More information at `http://www.ctan.org/tex-archive/help/Catalogue/entries/keystroke.html`

mouse A LaTeX package and a lot of pictures for the graphical representation of mice (with different buttons, pressed, not pressed, with wheel, ...)

More information at `http://tex.lickert.net/packages/mouse/index_en.html` or `http://tex.lickert.net/packages/mouse/index.html` (German).

This packages will go to CTAN, when the pictures are converted into a font (approx. at doomsday)

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 recommend to make a snapshot of your programm and include the picture via `includegraphics` (Package graphicx).

2.2 Menu-Figure

This command may be used to show a complete menu. I don't recommend to use this feature, instead I recommend to make a snapshot of your programm and include the picture via `\includegraphics` (Package graphicx).

`menufolder` Number of parameter: 2

1. (optional) width for the menu entry. Default `\menuitemlength`
2. Main menu topic
3. List of sub-entries

`\menuitem` Display a menu line inside a 'menufolder'.
Number of parameter: 2

1. (optional) width for the menu entry. Default `\menuitemlength`
2. Menu item text.

`\menuitemactive` Display an active menu line inside a 'menufolder'.
Number of parameter: 2

1. (optional) width for the menu entry. Default `\menuitemlength`
2. Menu item text.

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

`\menuitemlength` Default length for menu entries.
Recommendation: You can set it to the longest menu text:

```
%\settowidth{\menuitemlength}{longest menu text}
%
```

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}
%}
%
```

which results in:

Menu
Close and don't save
Close and save
Item ⇐

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

FiXme: *change option name* **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}
%\menusymbols[
%   \setlength{\menusep}{0em}~\\
% ]{
%   \\
%   \addtolength{\menusep}{1em}
%   \hspace*{\menusep}
% }{
%   $\wr$\\
%   \addtolength{\menusep}{2em}
%   \hspace*{\menusep}
% }[
% \ensuremath{\Leftarrow}
% ]
%
```

Example for a new style

```
Menu ≀
     Next Menu
       Entry ⇐
```

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 (op-

tional) and text.

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

3.2.1 Examples

Easy Menu

```
Menu
      Close and don't save
      Close and save
      ____________________
      Item                ⇐
```

Menu With Submenu

```
Menu
      Close and don't save
      Close and save
      Menu
            Close and don't save
            Close and save
            Item                ⇐
      Item
```

Example of the 'bug'

```
Menu
      Close and don't save
      Close and save
      Item                        ⇐
```

4 Ideas for the future

- Create Index entries for menus (optional)

  If I do it, it would be an additional index (style multind/spltidx. . .). 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.

  If you need something like this, take a look to my mouse.sty (`http://tex.lickert.net/packages/mouse/`).

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

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

5 Commands

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

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

`\menutext` How to write one step of the menu?

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

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

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

Check if the first parameter has content.

```
\ifx\@empty\@menuParOne%
{\ifx\@menuParTwo\gt%
 \else\expandafter\@menuTo\@menuParTwo <\@menu#2<%
 \fi}%
\else%
```

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

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

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

```
 \ifx\@menuParTwo\@empty%
 \else%
  \expandafter\@menuto\@menuParTwo <\@menu#2!<%
 \fi%
\fi%
}
```

`\@menuto`

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

`\@menuTo`

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

6 The different symbols

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

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

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

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

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

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

`\menusymbol` Possibility to redefine the symbols for "!" and "!!".

```
\newcommand*\menusymbols[3][]{%
\renewcommand*\@menusymbolStart{#1}%
\renewcommand*\@menusymbolOne{#2}%
\renewcommand*\@menusymbolTwo{#3}%
\@menusymbols%
}
```

`\@menusymbol` Define the End-Makro for a menu.

```
\newcommand*\@menusymbols[1][]{%
\renewcommand*\@menusymbolEnd{#1}%
}
```

`\menumathsymbols` Possibility to redefine the symbols for "!" and "!!".

```
\newcommand*\menumathsymbols[3][]{
\renewcommand*\@menusymbolStart{\ensuremath{#1}}
\renewcommand*\@menusymbolOne{\ensuremath{#2}}
\renewcommand*\@menusymbolTwo{\ensuremath{#3}}
}
```

7 The Predefined Layouts

7.1 Standard

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

8 Menu-Drawings

```
\newlength{\menuitemlength}
\newlength{\menusep}
\setlength{\menuitemlength}{6em}
\setlength{\menusep}{3em}
```

`\menuitemactivesymbol`

```
\newcommand\menuitemactivesymbol{\ensuremath{\Leftarrow}}%
```

FiXme: *parameter menufolder length (2cm results in 22cm)*

A negative parskip results in an overlapping menu.

Remark: the actual width of the menu is twice the width of the width given text length.

```
\newenvironment{menufolder}[2][\menuitemlength]{%
    \begin{minipage}{2#1}%
    \setlength{\parskip}{0pt}%
    \menufolderentry[#1]{#2}\par%
    \addtolength{\leftskip}{\menusep}%
    }{%
    \end{minipage}%
}
```

`\menufolderentry`

```
\newcommand\menufolderentry[2][\menuitemlength]{%
    \makebox[#1][l]{\menutext{#2}\hfill}%
}
```

`\menuitem`

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

`\menuitemactive`

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

FiXme: *besser positionieren (mischer?)*

`\menuseparator`

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

8.1 Options

```
\newif\ifmenuoptionhand
\DeclareOption{hand}{
\menuoptionhandtrue
}
```

```
\newif\ifmenuoptionframed
\DeclareOption{framed}{
\menuoptionframedtrue
}
```

```
\newif\ifmenuoptiongrey
\DeclareOption{grey}{
\menuoptiongreytrue
}
```

Process the options to set the flags.

```
\ProcessOptions\relax
```

8.1.1 Option hand

Activate the hand-option.

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

\menuitemactivesymbol

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

```
\fi%ifmenuoptionhand
```

8.1.2 Option framed

Activate the framed-option.

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

\menufolderentry

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

\menuitem

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

\menuitemactive

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

```
\fi%Option framed
```

8.1.3 Option grey

Activate the grey-option.

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

\menufolderentry

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

\menuitem

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

FiXme: *offset 1ex*

\menuitemactive

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

FiXme: *verbessern*

\menuseparator

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

```
\fi%Option grey
```