Abstract

This package is build to format menu sequences, paths and keystrokes.

You’re welcome to send me feedback, questions, bug reports and feature requests. If you like to support this package – especially improving or proofreading the manual – send me an e-mail, please.

Many thanks to Ahmed Musa, who provided the original list parsing code at https://tex.stackexchange.com/a/44389/4918.
Contents

1 Introduction 4

2 Installation 4

3 Package loading and options 5

4 Usage 5
  4.1 Basics .................................................. 5
  4.2 Styles .................................................... 6
    4.2.1 Predefined styles ................................ 6
    4.2.2 Declaring styles ................................. 10
    4.2.3 Copying styles ................................ 11
    4.2.4 Changing styles ................................ 11
  4.3 Color themes ......................................... 12
    4.3.1 Predefined themes ................................ 12
    4.3.2 Create a theme .................................. 12
    4.3.3 Copy a theme ................................... 13
    4.3.4 Change a theme ................................ 13
  4.4 Menu macros ........................................ 13
    4.4.1 Predefined menu macros ....................... 13
    4.4.2 Defining or changing menu macros ............ 13
  4.5 Keys .................................................. 14

5 Known issues and bugs 15

6 Implementation 15
  6.1 Required packages .................................. 15
  6.2 Helper macros ....................................... 17
  6.3 Options .............................................. 18
  6.4 Workarounds ........................................ 18
    6.4.1 hyperref's colorlinks option ................ 18
  6.5 Color themes ........................................ 19
    6.5.1 Internal commands ............................... 19
    6.5.2 User-level commands ........................... 19
    6.5.3 Predefined themes ............................. 20
  6.6 Styles ............................................... 20
    6.6.1 Internal commands .............................. 21
    6.6.2 User-level commands .......................... 22
    6.6.3 Copying and changing .......................... 23
    6.6.4 Predefined styles ............................ 24
  6.7 Menu macros ......................................... 30
    6.7.1 Internal commands .............................. 30
    6.7.2 User-level commands .......................... 32
    6.7.3 Predefined menu macros ...................... 32
  6.8 Keys .................................................. 33
7  Change history  
8  Macro index
1 Introduction

The menukeys package is mainly designed to parse and print sequences of software menus, folders and files, or keystrokes. Most predefined styles use the power of TikZ\(^1\) to format the output.

For example if you want to tell the reader of a manual how to set the ruler unit you may type

To set the unit of the rulers go to \texttt{\textbackslash menu\{Extras > Settings > Rulers\}} and choose between millimeters, inches and pixels. The shortcut to view the rulers is \texttt{\textbackslash keys\{cmd + R\}}. Pressing these keys again will hide the rulers.

The standard path for saving your document is \texttt{\textbackslash directory\{Macintosh HD/ Users/Your Name/Documents\}} but you can change it at \texttt{\textbackslash menu\{Extras > Settings > Saving\}} by clicking \texttt{\textbackslash menu\{Change save path\}}.

and get this:

To set the unit of the rulers go to \texttt{\textbackslash menu\{Extras > Settings > Rulers\}} and choose between millimeters, inches and pixels. The shortcut to view the rulers is \texttt{\textbackslash keys\{cmd + R\}}. Pressing these keys again will hide the rulers.

The standard path for saving your document is \texttt{\textbackslash directory\{Macintosh HD/ Users/Your Name/Documents\}} but you can change it at \texttt{\textbackslash menu\{Extras > Settings > Saving\}} by clicking \texttt{\textbackslash menu\{Change save path\}}.

The package is loaded as usual via

\texttt{\usepackage\{menukeys\}}

2 Installation

To install menukeys manually run

\texttt{latex menukeys.ins}

and copy \texttt{menukeys.sty} to a path where L\TeX{} can find it.

To typeset this manual run

\texttt{pdflatex menukeys.dtx}
\texttt{makeindex -s gglo.ist -o menukeys.gls menukeys.glo}
\texttt{makeindex -s gind.ist -o menukeys.ind menukeys.idx}
\texttt{pdflatex menukeys.dtx}
\texttt{pdflatex menukeys.dtx}

\(^{1}\) See \url{https://www.ctan.org/pkg/pgf}. 

4
3 Package loading and options

Since menukeys used to use catoptions, which does some heavy changes on key-value options, it was recommended to load menukeys as the last package (even after hyperref\footnote{See \url{https://tex.stackexchange.com/q/237683/4918} and \url{https://github.com/tweh/menukeys/issues/41}.}). This is no longer the case!

These are the possible options:

- **definenummacros**: Most of menukeys’ macros should not conflict with other packages\footnote{If you find a conflict send an e-mail.} but the predefined menu macros should be short and easy-to-read commands, which means that `\menu{A,B,C}` is preferred against `\printmenusequence{A,B,C}`. For that it’s not unlikely that they conflict with other packages. To prevent this you can tell menukeys to not define them by calling the option `definenummacros=false`. The default value is `true`. If you do so you have to define your own menu macros, see section 4.4 for details.

- **definekeys** (opt.) **definekeys**: Equal to `definenummacros` for the key macros. The default value is `true`.

- **mackeys** (opt.) **mackeys**: This option allows you to decide whether the mac keys are shown as text (mackeys=text) or symbols (mackeys=symbols). The default value is symbols.

- **os** (opt.) **os**: You can specify the OS by saying `os=mac` or `os=win`. This will cause some key macros to be rendered differently. The default value is mac.

- **hyperrefcolorlinks**: *Obsolete* (see sec. 5 and 6.4.1).

4 Usage

4.1 Basics

menukeys comes with three “menu macros” that parse and print lists. We have `\menu{⟨menu sequence⟩}`, with `>` as default input list separator, `\directory{⟨path and files⟩}` with `/` as default separator and `\keys{⟨keystrokes⟩}` with `+` as default separator. You’ve seen examples for all of them in section 1.

These macros have also an optional argument to set the input list separator. E.g. if you want to put in your menus with `,` instead of `>` you can say `\menu[,]{⟨menu sequence⟩}`.\footnote{If you want to change the input separator globally it’s recommended to renew the menu macro as described in section 4.4.}

The possible input separators are `/`, `=`, `*`, `,`, `;`, `.` ` `, `>`, `< and `\bslash` (to use `\` as separator). You can hide a separator from the parser by putting a
part of the sequence in braces. Spaces around the separator will be ignored, i.e. \keys{\ctrl+C} equals \keys{\ctrl + C}.

Example \menu[\]\{Extras,Settings,\{Units, rulers and origin\}\} gives

4.2 Styles

\texttt{menukeys} defines several “styles” that determine the output format of a menu macro. There are some predefined styles and others can be created by the user.

4.2.1 Predefined styles

Name: \texttt{menus}

\begin{center}
\begin{tabular}{l}
File \texttt{\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\end{tabular}
\end{center}

Name: roundedkeys

\[\text{Ctrl} + \text{Alt} + Q\]

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This \(\text{Ctrl} + \text{Alt} + Q\) is the result of a style which name is \texttt{roundedkeys}. And again some blind text without any sense.

\textit{The color of + is taken from optional color B.}

Name: shadowedroundedkeys

\[\text{Ctrl} + \text{Alt} + Q\]

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This \(\text{Ctrl} + \text{Alt} + Q\) is the result of a style which name is \texttt{shadowedroundedkeys}. And again some blind text without any sense.

\textit{The color of + is taken from optional color B.}
\textit{The shadow color is taken from optional color C.}

Name: angularkeys

\[\text{Ctrl} + \text{Alt} + Q\]

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This \(\text{Ctrl} + \text{Alt} + Q\) is the result of a style which name is \texttt{angularkeys}. And again some blind text without any sense.

\textit{The color of + is taken from optional color B.}

Name: shadowedangularkeys

\[\text{Ctrl} + \text{Alt} + Q\]

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This \(\text{Ctrl} + \text{Alt} + Q\) is the result of a style which name is \texttt{shadowedangularkeys}. And again some blind text without any sense.

\textit{The color of + is taken from optional color B.}
\textit{The shadow color is taken from optional color C.}
Name: typewriterkeys

This is some more or less blind text, to demonstrate how the sequence looks in text. This \texttt{typewriterkeys} is the result of a style which name is \texttt{typewriterkeys}. And again some blind text without any sense.

\textit{The color of + is taken from optional color B.}

Name: paths

This is some more or less blind text, to demonstrate how the sequence looks in text. This \texttt{C: \textbackslash \textbackslash User \textbackslash Folder \textbackslash MyFile.tex} is the result of a style which name is \texttt{paths}. And again some blind text without any sense.

\textit{The sep color is taken from optional color C.}

Name: pathswithfolder

This is some more or less blind text, to demonstrate how the sequence looks in text. This \texttt{C: \textbackslash \textbackslash User \textbackslash Folder \textbackslash MyFile.tex} is the result of a style which name is \texttt{pathswithfolder}. And again some blind text without any sense.

\textit{The folder draw color is taken from optional color B.}
\textit{The folder fill color is taken from optional color A.}
\textit{The sep color is taken from optional color C.}

Name: pathswithblackfolder

This is some more or less blind text, to demonstrate how the sequence looks in text. This \texttt{C: \textbackslash \textbackslash User \textbackslash Folder \textbackslash MyFile.tex} is the result of a style which name is \texttt{pathswithblackfolder}. And again some blind text without any sense.

\textit{The folder draw color is taken from optional color B.}
\textit{The folder fill color is taken from optional color C.}
\textit{The sep color is taken from optional color C.}

The following three styles allow paths elements to be hyphenated, but they insert only a line break without a hyphen dash. Note that they only work with T1 and
OT1 encoding (at least I tested only these ones) and that this in some cases doesn’t work very well.

Name: hyphenatepaths

```
C:\Database\User\ALongUserNameHere\ALongerFolderNameAtThisPlace\MyFile.tex
```

MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This `C:\Database\User\ALongUserNameHere\ALongerFolderNameAtThisPlace\MyFile.tex` is the result of a style which name is `hyphenatepaths`. And again some blind text without any sense.

The sep color is taken from optional color C.

Name: hyphenatepathswithfolder

```
C: Database User ALongUserNameHere ALongerFolderNameAtThisPlace MyFile.tex
```

MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This `C: Database User ALongUserNameHere ALongerFolderNameAtThisPlace MyFile.tex` is the result of a style which name is `hyphenatepathswithfolder`. And again some blind text without any sense.

The folder draw color is taken from optional color B.
The folder fill color is taken from optional color A.
The sep color is taken from optional color C.

Name: hyphenatepathswithblackfolder

```
□ C: Database User ALongUserNameHere ALongerFolderNameAtThisPlace MyFile.tex
```

MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This `□ C: Database User ALongUserNameHere ALongerFolderNameAtThisPlace MyFile.tex` is the result of a style which name is `hyphenatepathswithblackfolder`. And again some blind text without any sense.

The folder draw color is taken from optional color B.
The folder fill color is taken from optional color C.
The sep color is taken from optional color C.
\drawtikzfolder  
**Hint**  The folder is drawn with the command \drawtikzfolder which is part of menukeys and has two optional arguments to change the color of the lines and the fill color of the front:
\drawtikzfolder\[⟨front fill⟩\]\[⟨draw⟩\]

4.2.2 Declaring styles

\newmenustylesimple  
The simplest way to define a new style is to use \newmenustylesimple. It has six arguments:
\newmenustylesimple\(*\){⟨name⟩}{⟨pre⟩}{⟨style⟩}{⟨sep⟩}{⟨post⟩}{⟨theme⟩}

- **name** is the name of the new style. It must follow the specifications of \TeX control sequences, which means it must contain only letters and no numbers.
- **pre** is the code which is executed before a menu macro.
- **style** is the style for the first list element. It has to be a TikZ-style which is applied to a node, e.g. \texttt{draw,blue}.
- **sep** is the code executed between the lists elements, e.g. some space or a symbol.
- **post** is the code which is executed after a menu macro.
- **theme** is a color theme (see section 4.3).

**Example**  Let us consider we want a list that prints a frame around its elements and separates them by a star. We can use
\newmenustylesimple\{mystyle\}\{draw\}\[$\ast$\]\{mycolors\}

\newmenustyle  
The more advanced command is \newmenustyle. It has nine arguments:
\newmenustyle\(*\){⟨name⟩}{⟨pre⟩}{⟨first⟩}{⟨sep⟩}{⟨mid⟩}{⟨last⟩}{⟨single⟩}{⟨post⟩}{⟨theme⟩}

- **name** is the name of the new style. It must follow the specifications of \TeX control sequences, which means it must contain only letters and no numbers.
- **pre** is the code which is executed before a menu macro.
- **first** is the style for the first list element. It has to be a TikZ-style which is applied to a node, e.g. \texttt{draw,blue}.
- **sep** is the code executed between the lists elements, e.g. some space or a symbol.
- **mid** is the style for all elements between the first and the last one. It has to be a TikZ style.
- **last** is the style for the last list element. It has to be a TikZ style.
- **single** this style is used if the list contains only one element. It has to be a TikZ style.
post is the code which is executed after a menu macro.

theme is a color theme (see section 4.3).

Example We can extend the previous example and desire that the first and the last element became red, and a single element should have a dashed frame. Furthermore the menu sequence should be preceded and followed by a bullet point:

\newmenustyle{mystyle}{\$\bullet\$}{draw,red}{\$\ast\$}{\$\bullet\$}

\{draw\}{draw,red}{draw,dashed}{\$\bullet\$}

If the TikZ node system doesn’t fit your needs there are the starred versions: Use them and the arguments \(first\), \(mid\), \(last\), \(single\) can be any \LaTeX{} code. To access the current list element use \CurrentMenuElement. 

Example consider that we want all menu elements simple be fat and not drawn with a TikZ node. The separator should be the star again:

\newmenustylesimple*{mystyle}{\textbf{\CurrentMenuElement}}{\$\ast\$}

If you want to make your own style you must take care of using the color theme. To access a color of the currently applied theme while defining a style use \usemenucolor\{\(element\)\} (See section 4.3 for details about possible elements).

4.2.3 Copying styles

\copymenustyle

Example To copy the definition of mystyle to mycopy use

\copymenustyle{mycopy}{mystyle}

4.2.4 Changing styles

The simplest change we can imagine is to change a single element or the color theme of an existing style. For the first case there is \changemenuelement\{\(\ast\)\}{\(name\)}{\(element\)}{\(definition\)}}, where the starred version works like the one of \newmenustyle does.

Example To change the single element of mystyle from dashed to solid use the following code. You may save the original style by copying it as described above.

\changemenuelement{mystyle}{single}{draw}

\changemenucolortheme

Example To change the color theme of mystyle to myothercolors call

\changemenucolortheme{mystyle}{myothercolors}
The next level is redefining a style. This package provides the following macros: \renewmenustylesimple, \providemenustylesimple, \renewmenustyle and \providemenustyle.

4.3 Color themes

To make the colors of a style become changeable without touching the style itself, menukeys uses “color themes”. Every color theme must contain three color definitions that can be used to draw a node background, a node frame and a text color, and additionally two optional colors used by some themes.

4.3.1 Predefined themes

There are two predefined color themes

Name: gray
Background: ■ Border: ■ Text: ■ (A: ■ B: ■ C: ■)

Name: blacknwhite
Background: ■ Border: ■ Text: ■ (A: ■ B: ■ C: ■)

4.3.2 Create a theme

To create a new theme use \newmenucolortheme. It uses the following arguments:
\newmenucolortheme{⟨name⟩}{⟨model⟩}{⟨bg⟩}{⟨br⟩}{⟨txt⟩}[⟨a⟩][⟨b⟩][⟨c⟩]

name is the name of the theme and must contain only letters.
model is the xcolor color model which is used to define a color, e.g. named, rgb, cmyk, ...
bg is the color definition for the node background.
br is the color definition for the node border.
txt is the color definition for the node’s text.
a is an optional additional color (by default same as bg).
b is an optional additional color (by default same as br).
c is an optional additional color (by default same as txt).

Example To create a theme called mycolors we can say
\newmenucolortheme{mycolors}{named}{red}{green}{blue}
4.3.3 Copy a theme

\copymenucolortheme{(copy)}{(original)}.

**Example**  To copy the colors of mycolors to copycolors type

\copymenucolortheme{copycolors}{mycolors}

4.3.4 Change a theme

\changenmenucolor If you want to change the color of a theme’s element use \changenmenucolor{⟨name⟩}{⟨element⟩}{⟨model⟩}{⟨color definition⟩}, where name is the theme’s name and ⟨element⟩ is bg, br, or txt.

**Example**  Let’s change the text color of mycolors:

\changenmenucolor{mycolors}{txt}{named}{gray}

\renewmenucolortheme  To redefine a complete theme use \renewmenucolortheme. It works with the same arguments as \newmenucolortheme.

4.4 Menu macros

The “menu macros” take a list separated by a special symbol to print it with a menu style.

4.4.1 Predefined menu macros

See section 4.1.

4.4.2 Defining or changing menu macros

\newmenumacro To define a new menu macro call \newmenumacro{⟨macro⟩}{⟨input sep⟩}{⟨style⟩}.

name is a LATEX control sequence name.

input sep is the default separator used in the input list (see section 4.1 for a list of valid separators).

If you don’t give it the package’s default (,) is used.

style is a menu style.

This will give you a macro like \⟨macro⟩{⟨input sep⟩}{⟨list⟩}

**Example**  Assuming you need a command to format Windows paths, you can define it with

\newmenumacro{\winpath}{\bslash}{\mystyle}
and then use it as e.g. `\winpath{C:\System\Deep\Deeper\YourFile.txt}`. Note that `mystyle` must be defined before you call `\newmenumacro`.

There are also the two commands `\providemenumacro` and `\renewmenumacro` which take the same arguments as `\newmenumacro` and work like the `\providecommand` and `\renewcommand` macro.

**Example** To change the default input separator of \menu you must know the default style (which is `menus`) and then you can say

```
\renewmenumacro{\menu}{[,]\{menus\}}
```

### 4.5 Keys

The menukeys package comes with some macros to print special keys in the sequences set with `\keys`. Depending on the given OS (see section 3) some macros behave differently to be able to use a key even if it's undefined via the `os` option macros like `\key{mac}` and `\key{win}` that will always give the right symbol.

The full list of key macros is shown in table 1.

<table>
<thead>
<tr>
<th>Macro</th>
<th>Mac</th>
<th>Win.</th>
</tr>
</thead>
<tbody>
<tr>
<td>shift</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>capslock</td>
<td>⇧</td>
<td>⇧</td>
</tr>
<tr>
<td>tab</td>
<td>→</td>
<td>→</td>
</tr>
<tr>
<td>esc</td>
<td>esc / ^</td>
<td>Esc</td>
</tr>
<tr>
<td>oldesc</td>
<td>esc / ⊳</td>
<td>Esc</td>
</tr>
<tr>
<td>ctrl</td>
<td>ctrl</td>
<td>Ctrl</td>
</tr>
<tr>
<td>Alt</td>
<td>alt / `</td>
<td>Alt</td>
</tr>
<tr>
<td>AltGr</td>
<td>Alt Gr</td>
<td></td>
</tr>
<tr>
<td>cmd</td>
<td>cmd / ⍨</td>
<td>⍨</td>
</tr>
<tr>
<td>Space</td>
<td>[empty sp.]</td>
<td>[empty sp.]</td>
</tr>
<tr>
<td>SPACE</td>
<td>Space</td>
<td>Space</td>
</tr>
<tr>
<td>return</td>
<td>←</td>
<td>↓</td>
</tr>
<tr>
<td>enter</td>
<td>←</td>
<td>Enter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Macro</th>
<th>Mac</th>
<th>Win.</th>
</tr>
</thead>
<tbody>
<tr>
<td>winmenu</td>
<td>←</td>
<td>←</td>
</tr>
<tr>
<td>backspace</td>
<td>←</td>
<td>←</td>
</tr>
<tr>
<td>del</td>
<td>Del. / ⏎</td>
<td>Del.</td>
</tr>
<tr>
<td>backdel</td>
<td>Del. / ⏎</td>
<td>Del.</td>
</tr>
<tr>
<td>arrowkey{^}</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>arrowkeyup</td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td>arrowkeydown</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>arrowkey{&lt;}</td>
<td>←</td>
<td>←</td>
</tr>
<tr>
<td>arrowkeyright</td>
<td>←</td>
<td>←</td>
</tr>
<tr>
<td>arrowkeyleft</td>
<td>←</td>
<td>←</td>
</tr>
</tbody>
</table>

The macro `\arrowkey{direction}` is a little special since it takes the direction as a single character: ^, v (lower case v), > or <.

The texts for `\ctrl`, `\del` and `\SPACE` are saved in `\ctrlname`, `\delname`, `\spacename` respectively. So you can change them with `\renewcommand`.

The rendering of some Mac macros depend on the option `mackeys` The different versions are shown in the table (left: text, right: symbols).
I apologize that there are no commands for the windows key and the apple logo, but that would be a copyright infringement.

5 Known issues and bugs

- If you use the `inputenc` package `menukeys` must be loaded after it. Otherwise some key macros get corrupted.

- `menukeys` must be loaded after `xcolor`, if you load the latter with options. Otherwise you’ll get an option clash. Since `menukeys` loads `xcolor` internally you may pass options as global options via `\documentclass` or directly to it via `\PassOptionsToPackage`.

  **Example**  Set xcolor to cmyk model:

  \begin{verbatim}
\documentclass{article}
\PassOptionsToPackage{cmyk}{xcolor}
\usepackage{menukeys}
\begin{document}
  Hello World!
\end{document}
\end{verbatim}

If you find something to add to this list please send me an e-mail or report a bug on GitHub ([https://github.com/tweh/menukeys](https://github.com/tweh/menukeys)).

6 Implementation

6.1 Required packages

Load the required packages

1 `\RequirePackage{xparse}`
2 `\RequirePackage{xstring}`
3 `\RequirePackage{etoolbox}`

Furthermore we need Ti\kZ and some of its libraries,

4 `\RequirePackage{tikz}`
5 `\usetikzlibrary{calc, shapes, symbols, shadows}`

the color package `xcolor` and `adjustbox` for the typewriterkeys style.

6 `\RequirePackage{xcolor}`
7 `\RequirePackage{adjustbox}`

Load `relsize` to be able to change the font size relative to the surrounding text.

8 `\RequirePackage{relsize}`
To define the list parsing commands and allow `\` as a separator we used to load \texttt{catoptions}. Instead we now use some \texttt{expl3} functions to replace the macros we required from \texttt{catoptions}.

The first few of these functions are more or less direct equivalents. A bit of attention has to be paid for \texttt{\tw@mk@ifinsetTF} as it requires the arguments to get swapped.

```latex
\ExplSyntaxOn
\cs_new_eq:NN \tw@mk@trimspaces \tl_trim_spaces:n
\cs_new_eq:NN \tw@mk@exp@Nnno \exp_args:Nnno
\cs_new_eq:NN \tw@mk@string \cs_to_str:N
\prg_generate_conditional_variant:Nnn \tl_if_in:nn { xx } { TF }
\cs_new:Npn \tw@mk@xifinsetTF #1 #2
{ \tl_if_in:xxTF {#2} {#1} }
\ExplSyntaxOff
```

The replacement for \texttt{\indrisloop} will not set the conditional \texttt{\iflastindris}, instead we can check whether the sequence is empty to see whether this is the last element. This test will not use a TeX-like \texttt{\iflastindris...\else...\fi} construct but instead two branches.

```latex
\cs_new:Npn \tw@mk@iflastindris
{ \seq_if_empty:NTF \l__twmk_indris_seq }
```

Replacing \texttt{\indrisloop} is a bit more work as it requires us to push some values to a stack (to allow nested usage, this may not be necessary for \texttt{menukeys}, but it is part of the original \texttt{\indrisloop} so we should play nice here). First we'll need a few variables.

```latex
\seq_new:N \l__twmk_indris_seq
\int_new:N \l__twmk_indris_int
\tl_new:N \l__twmk_indris_tl
\seq_new:N \l__twmk_indris_seqstack_seq
\seq_new:N \l__twmk_indris_intstack_seq
```

Our stack will use another sequence in which the definitions of the parent call will be stored for the sequence and the integer. The other variables put on a stack by \texttt{\indrisloop} aren't required. The synopsis of \texttt{\tw@mk@indrisloop} will be different to the one provided by \texttt{catoptions}. The delimiter will be a mandatory argument (not in brackets), and there is no starred version.

```latex
\cs_new_protected:Npn \__twmk_pushseq:
{ \seq_push:No \l__twmk_indris_seqstack_seq \l__twmk_indris_seq }
```

```latex
\cs_new_protected:Npn \__twmk_pushint:
{ \seq_push:NV \l__twmk_indris_intstack_seq \l__twmk_indris_int }
```
The real loop works by first splitting the input into a sequence according to the delimiter in \texttt{#1}. Then this sequence is stepped through, but instead of using \texttt{\seq_map:NN} we’ll have to pop the sequence into a local variable so that our test for the last element works. The parameter \texttt{#2} has to be expanded once as it is handed in as a token storing the real argument in later use.

6.2 Helper macros

Define macros to call \texttt{\PackageError} and warnings

\newcommand*{\tw@mk@error}[2]{\PackageError{menukeys}{#2}{#1}}
\newcommand*{\tw@mk@warning}[1]{\tw@mk@warning@noline{Please consult the manual for more information.}}
\PackageWarning{menukeys}{#1} \newcommand*{\tw@mk@warning@noline}[1]{\PackageWarningNoLine{menukeys}{#1}}
\tw@mk@tempa \tw@mk@tempb
Some commands for temporary use:
\def{\tw@mk@tempa}{}
\def{\tw@mk@tempb}{}
\tw@mk@gobble@args
Define a command to gobble arguments.
\DeclareDocumentCommand{\tw@mk@gobble@args}{m}{\RenewDocumentCommand{\tw@mk@tempa}{#1}{}{\tw@mk@tempa}}
\tw@mk@gobble@args

6.3 Options
First we declare and process the package options
\RequirePackage{kvoptions}
\SetupKeyvalOptions{
  family=tw@mk,
  prefix=tw@mk@
}
\DeclareBoolOption[true]{definemenumacros}
\DeclareBoolOption[true]{definekeys}
\DeclareBoolOption[false]{hyperrefcolorlinks}
\DeclareStringOption[mac]{os}
\DeclareStringOption[symbols]{mackeys}
\ProcessKeyvalOptions{tw@mk}\relax
Now we have to do some error treatment:
\IfSubStr{.mac.win.}{.\tw@mk@os.}{}{\tw@mk@error{Unknown value for option 'os'\MessageBreak Possible values are 'mac' or 'win'.}}
\IfSubStr{.symbols.text.}{.\tw@mk@mackeys.}{}{\tw@mk@error{Unknown value for option 'mackeys'\MessageBreak Possible values are 'symbols' or 'text'}.}

6.4 Workarounds
Some workarounds to “solve” some incompatibilities:

6.4.1 hyperref's colorlinks option
There used to be an issue with using the colorlinks option of hyperref due to catoptions being loaded. Since catoptions isn’t required any more, this
workaround isn’t necessary. For backwards compatibility the `hyperrefcolorlinks` option is still evaluated and just uses \hypersetup or \PassOptionsToPackage depending on whether `hyperref` is already loaded.

```latex
\iftw@mk@hyperrefcolorlinks
\tw@mk@warning{The option `hyperrefcolorlinks' is obsolete}
\fi
```

6.5 Color themes

6.5.1 Internal commands

First we define an internal command to make a color theme

```latex
\newcommand*{\tw@make@color@theme}[8]{%
    \definecolor{tw@color@theme@#1@bg}{#2}{#3}%
    \definecolor{tw@color@theme@#1@br}{#2}{#4}%
    \definecolor{tw@color@theme@#1@txt}{#2}{#5}%
    \definecolor{tw@color@theme@#1@a}{#2}{#6}%
    \definecolor{tw@color@theme@#1@b}{#2}{#7}%
    \definecolor{tw@color@theme@#1@c}{#2}{#8}%
}\endinput
```

6.5.2 User-level commands

After that we define the user-level commands:

```latex
\newcommand*{\newmenucolortheme}{ m m m m O{#3} O{#4} O{#5} }{%
    \IfSubStr{ bg br txt }{ #2 }{%
        \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
    }{%
        \tw@mk@error{Color theme '#1' already defined!\MessageBreak
            Use \string\renewmenucolortheme\space instead.}%
    }
}\endinput
```

Lastly we define the changing and copying commands

```latex
\newcommand*{\changemenucolor}{bg br txt }{ #2 }{%
    \definecolor{tw@color@theme@#1@#2}{#3}{#4}%
}\endinput
```

19
To be able to change the color theme of a style we must define this:

\newcommand{\changemenucolortheme}[2]{% 
  \ifcsundef{tw@style@#1@pre}{% 
    \tw@mk@error{Style '#1' undefined! \MessageBreak Maybe you misspelled it??}% 
  }{% 
    \@ifundefinedcolor{tw@color@theme@#2@bg}{% 
      \tw@mk@error{Color theme '#2' is not defined!}% 
    }{% 
      \csdef{tw@style@#1@color@theme}{#2}% 
    }% 
  }% 
}

To use a color of a theme we define \usemenucolor as following.

\newcommand{\usemenucolor}[1]{% 
  \tw@color@theme@\tw@current@color@theme @#1% 
}

6.5.3 Predefined themes

There are two predefined color themes

\newmenucolortheme{gray}{gray}{0.95}{0.3}{0}%[0.95][0][0]

\newmenucolortheme{blacknwhite}{gray}{1}{0}{0}%[1][0][0]

6.6 Styles

The style generating commands will set some commands that are named like \tw@style@⟨name⟩@⟨element⟩.

Before we can define the internal declaring macro to use it later in the user level commands, we have to set some defaults for the optional arguments

\newcommand{\tw@default@sep}{% 
  \hspace{0.2em plus 0.1em minus 0.5em}% 
} \newcommand{\tw@default@pre}{% 
} \newcommand{\tw@default@post}{%
6.6.1 Internal commands

Now we can define the internal commands.

```
\DeclareDocumentCommand{\tw@declare@style@simple}{%}
\tw@default@pre m %O{\tw@default@sep} O{\tw@default@post} m}

\csdef{\tw@style@#2@color@theme}{#7} %
\csdef{\tw@style@#2@pre}{#3} %
\csdef{\tw@style@#2@sep}{#5} %
\csdef{\tw@style@#2@post}{#6} %
\IfBooleanTF{#1}{% 
\csdef{\tw@style@#2@single}{#4} %
\csdef{\tw@style@#2@first}{#4} %
\csdef{\tw@style@#2@mid}{#4} %
\csdef{\tw@style@#2@last}{#4} %
}\% 
\csdef{\tw@style@#2@single}{% 
\tikz[baseline=(tw@node.base)]{\node(tw@node)[#4]{\strut\CurrentMenuElement};} %
\csdef{\tw@style@#2@first}{% 
\tikz[baseline=(tw@node.base)]{\node(tw@node)[#4]{\strut\CurrentMenuElement};} %
\csdef{\tw@style@#2@mid}{% 
\tikz[baseline=(tw@node.base)]{\node(tw@node)[#4]{\strut\CurrentMenuElement};} %
\csdef{\tw@style@#2@last}{% 
\tikz[baseline=(tw@node.base)]{\node(tw@node)[#4]{\strut\CurrentMenuElement};} %
\%}
```

The next step is to create the extended command. This command must have ten arguments (including the star) so we have to define a helping macro to grab the last two macros.

```
\DeclareDocumentCommand{\tw@declare@style@extra@args}{%}
\tw@default@post m}

\csdef{\tw@style@\tw@current@style @post}{#1} %
\csdef{\tw@style@\tw@current@style @color@theme}{#2} %
```

Now we can define \tw@declare@style:

```
\DeclareDocumentCommand{\tw@declare@style}{%}
\tw@default@pre m %O{\tw@default@sep} O{\tw@default@post} m m m}

\def{\tw@current@style}{#2} %
\csdef{\tw@style@\tw@current@style #2}{#3} %
\csdef{\tw@style@\tw@current@style #2}{#5} %
```
6.6.2 User-level commands

It's time to define the user-level commands now:

\NewDocumentCommand{\newmenustylesimple}{s m}{%  
\ifsundef{tw@style@#2@pre}{%  
 \IfBooleanTF{#1}{%  
 \tw@declare@style@simple*{#2}%;  
 }{%  
 \tw@declare@style@simple{#2}%;  
 }%  
 %}  
 \tikz[baseline=(tw@node.base)]{  
 \node(tw@node)[#8]{\strut\CurrentMenuElement};}%;  
 \tw@declare@style@simple@extra@args%  
}%}

\NewDocumentCommand{\renewmenustylesimple}{s m}{%  
\IfBooleanTF{#1}{%  
 \tw@declare@style@simple*{#2}%;  
 }{%  
 \tw@declare@style@simple{#2}%;  
 }%  
 %}

\NewDocumentCommand{\providemenustylesimple}{s m}{%  
\ifsundef{tw@style@#2@pre}{%  
 \IfBooleanTF{#1}{%  
 \tw@declare@style@simple*{#2}%;  
 }{%  
 \tw@declare@style@simple{#2}%;  
 }%  
 %}  
 %}

newmenustylesimple
renewmenustylesimple
providemenustylesimple
newmenustyle
renewmenustyle
providemenustyle
6.6.3 Copying and changing

\copymenustyle The last two steps in this part are to define a command to copy styles
\newcommand*{\copymenustyle}[2]{%
  \ifcsundef{tw@style@#1@pre}{%
    \ifcsundef{tw@style@#2@pre}{%
      \tw@mk@error{Can't copy not existing style ('#2')!}\
    }{%
      \csletcs{tw@style@#1@pre}{tw@style@#2@pre}\%\n      \csletcs{tw@style@#1@post}{tw@style@#2@post}\%\n      \csletcs{tw@style@#1@sep}{tw@style@#2@sep}\%\n      \csletcs{tw@style@#1@single}{tw@style@#2@single}\%\n      \csletcs{tw@style@#1@first}{tw@style@#2@first}\%\n      \csletcs{tw@style@#1@mid}{tw@style@#2@mid}\%\n      \csletcs{tw@style@#1@last}{tw@style@#2@last}\%\n      \csletcs{tw@style@#1@color@theme}{tw@style@#2@color@theme}\%
    }{%
      \tw@mk@error{Style '#1' already exists!}\
    }%
  }{%
}
}

\changemenuelement
and one to change a single element of a style.

\NewDocumentCommand{\changemenuelement}{s m m m}{%
  \ifcsundef{tw@style@#2@pre}{%
    \tw@mk@error{Style '#2' undefined.}\
  }{%
    \IfSubStr{ single first middle last pre post sep }{ #3 }{ %
      \IfBooleanTF{#1}{%
        \csdef{tw@style@#2@#3}{#4}\
      }{%
        \IfSubStr{ pre post sep }{ #3 }{ %
          \csdef{tw@style@#2@#3}{#4}\
        }{%
          \csdef{tw@style@#2@#3}{%\n            \tikz[baseline=(tw@node.base)]{\%\n              \node(tw@node)[#4]{\strut\color{\usemenucolor{txt}} \CurrentMenuElement};}}\%
        }{%
          \tw@mk@error{No element '#3'. Possible values are single, first, middle, last, pre, post or sep.}\
        }%
      }%
    }%
}

\begin{tikzpicture}
  \node[draw, fill, text=\usemenucolor{txt}]{\CurrentMenuElement};
\end{tikzpicture}

6.6.4 Predefined styles
We define several styles for menu sequences, paths and keystrokes.

\tw@set@tikz@colors
First we define a TikZ-style to apply the color theme to a node easily

\tikzset{\tw@set@tikz@colors/.style={%
  draw=\usemenucolor{br},
  fill=\usemenucolor{bg},
  text=\usemenucolor{txt},
}}
Now we can define the styles. To keep the most settings of a style together we make additional \texttt{tikz}-styles instead of setting everything directly to the nodes.

\begin{verbatim}
\tikzset{tw@menus@base/.style={%  
  tw@set@tikz@colors,  
  rounded corners=0.15ex,  
  inner sep=0pt,  
  inner xsep=2pt,  
  text height=1.825ex,  
  text depth=0.7ex,  
  minimum width=1.5em,  
  font=\relsize{-1}\sffamily,  
  signal,  
  signal to=nowhere,  
  signal pointer angle=110,  
}}
\tw@declare@style*{menus}{%  
  \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%  
    \node(tw@node)[tw@menus@base,signal to=east]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%  
  }{  
    \hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}%  
    {\%  
      \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%  
        \node(tw@node)[tw@menus@base,signal from=west,signal to=east]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%  
      }{  
        \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%  
          \node(tw@node)[tw@menus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%  
      }{gray}%  
  }{\%  
  \tikzset{tw@roundedmenus@base/.style={%  
  tw@set@tikz@colors,  
  rounded corners=0.3ex,  
  inner sep=0pt,  
  inner xsep=2pt,  
  text height=1.825ex,  
  text depth=0.7ex,  
  minimum width=1.5em,  
  font=\relsize{-1}\sffamily,  
  signal,  
  signal to=nowhere,  
  signal pointer angle=110,  
}}}
\tw@declare@style*{roundedmenus}{%  
  \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%  
    \node(tw@node)[tw@roundedmenus@base,signal to=east]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%  
  }{  
    \hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}%  
    {\%  
      \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%  
        \node(tw@node)[tw@roundedmenus@base,signal from=west,signal to=east]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%  
      }{  
        \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%  
          \node(tw@node)[tw@roundedmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%  
      }{gray}%  
  }{\%
\end{verbatim}
\tw@declare@style@simple{*}{angularkeys}{%
\begin{tikzpicture}[baseline={($(tw@node.base)+(0,-0.2ex)$)}]
\node(tw@node)[tw@angularkeys@base]
{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};
\end{tikzpicture}%
\hspace{0.1em plus 0.1em minus 0.05em}%
\textcolor{\usemenucolor{b}}{\sffamily\relsize{-2}+}%
\hspace{0.1em plus 0.1em minus 0.05em}%
}\[gray]\]
\tikzset{tw@shadowedangularkeys@base/.style={%
\tw@set@tikz@colors,
inner sep=0pt,
inner xsep=2pt,
text height=1.825ex,
text depth=0.7ex,
minimum width=1.5em,
font=\sffamily,\relsize{-1},
general shadow={%
shadow xshift=.2ex, shadow yshift=-.15ex,
fill=\usemenucolor{c},
},%
}}%
\tw@declare@style@simple{*}{shadowedangularkeys}{%
\begin{tikzpicture}[baseline={($(tw@node.base)+(0,-0.2ex)$)}]
\node(tw@node)[tw@shadowedangularkeys@base]
{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};
\end{tikzpicture}%
\hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
\textcolor{\usemenucolor{b}}{\sffamily\relsize{-2}+}%
\hspace{0.1em plus 0.1em minus 0.05em}%
]\[\hspace{0.2ex}\]{gray}]
\tikzset{tw@typewriterkeys@base/.style={%
\tw@set@tikz@colors,
\tw@shape=circle,
minimum size=2ex,
inherent sep=0.5pt, outer sep=1pt,
font=\ttfamily,\relsize{-1},}]
\tw@declare@style@simple{*}{typewriterkeys}{%
\begin{tikzpicture}[baseline={($(tw@node.south)+(0,0.8ex)$)}]
\node(tw@node)[tw@typewriterkeys@base, inner sep=1.25pt, line width=0.6pt]
{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
\node[tw@typewriterkeys@base]
{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
\end{tikzpicture}%
\hspace{0.2ex}\hspace{0.05ex}%
\textcolor{\usemenucolor{b}}{\sffamily\relsize{-2}+}%
\hspace{0.05ex}%
]\[\hspace{0.2ex}\]{gray}]
\begin{tikzpicture}[baseline={($(tw@node.south)+(0,0.8ex)$)}]
\node(tw@node)[tw@typewriterkeys@base, inner sep=1.25pt, line width=0.6pt]
{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
\end{tikzpicture}%
\hspace{0.2ex}\hspace{0.05ex}%
\textcolor{\usemenucolor{b}}{\sffamily\relsize{-2}+}%
\hspace{0.05ex}%
];
\end{tikzpicture}
\end{document}
6.7 Menu macros

6.7.1 Internal commands

\tw@default@input@sep

First we define our default input separator
\def\tw@default@input@sep{,}

\CurrentMenuElement

and the $\CurrentMenuElement$ dummy
\def\CurrentMenuElement{}

Then we set up the internal command to create new menu macros. The list parsing code was essentially provided by Ahmed Musa at \url{https://tex.stackexchange.com/a/44989/4918}. Jonathan P. Spratte made some major changes to make menukeys work without catoptions and reimplemented the parsing code using \LaTeX3. Thank you both very much!
\begingroup
\lccode'\,=1
\lowercase{endgroup
\@ifdefinable\tw@mk@test@input@sep

\tw@define@menu@macro
\tw@define@menu@macro

\protected\def\tw@mk@test@input@sep#1{\tw@mk@xifinsetTF{,\tw@mk@trimspaces(#1),}{,bslash,backslash,\texttt{directory,location},}\}}\newcommand\tw@define@menu@macro[3]{\IfNoValueTF{#3}{\tw@mk@exp@Nnno\tw@define@menu@macro@{#1}{#2}\tw@default@input@sep}{\tw@define@menu@macro@{#1}{#2}{#3}}}\newcommand\tw@define@menu@macro@[4]{\ifcsundef{tw@style@#4@sep}{\tw@mk@error}{\Can't\ define\ menu\ macro\ \texttt{\string#2}\space,\MessageBreak\ because\ the\ style\ '##4'\ is\ not\ available!}\}}\csdef{parse@menu@list@\tw@mk@string#2}{\CurrentMenuElement}{\tw@mk@iflastindris{\ifnum\tw@mk@indrisnr=\@ne}{\@nameuse{tw@style@#4@single}}{\else}{\@nameuse{tw@style@#4@sep}}{\@nameuse{tw@style@#4@mid}}}{\if\ifnum\tw@mk@indrisnr=\@ne}{\@nameuse{tw@style@#4@first}}{\else}{\@nameuse{tw@style@#4@sep}}{\@nameuse{tw@style@#4@mid}}}#1#2{+O{#3}+m}\leavevmode\begingroup\def\tw@current@color@theme
6.7.2 User-level commands

Now it's time to build the user-level commands

\newmenumacro
\renewmenumacro
\providemenumacro

6.7.3 Predefined menu macros

Now we got all tools to predefine some menu macros. To be sure that these commands won't conflict with other packages we introduced the option definemacros. Here we have to check it:
And then we define three basic macros.

```latex
\newmenumacro{\menu}{\>[>]{menus}}
\newmenumacro{\directory}{[/]{paths}}
\newmenumacro{\keys}{[+]\{roundedkeys\}}
```

Lastly we close the `definemacros` if statement:

```latex
\fi
```

## 6.8 Keys

Before we define anything we check if the user allows it:

```latex
\iftw@mk@definekeys
```

Before define the key macros we create some macros that save some typing by condensing the similarities between the key macros.

### \tw@make@key@box

The first of these macros helps us building save boxes to store the `{tikzpicture}`, that will draw the key later. This is necessary because otherwise the picture will inherit the style of the key sequence `node`.

```latex
\NewDocumentCommand{\tw@make@key@box}{m m}{%
  \expandafter\newbox\csname tw@mk@box@#1\endcsname
  \expandafter\sbox\csname tw@mk@box@#1\endcsname{#2}%
  \csdef{tw@mk@#1}{%
    \expandafter\usebox\csname tw@mk@box@#1\endcsname#2%}
}
```

### \tw@make@key@macro

The next macro defines the user level command by accessing a macro like `tw@mk@\langle key \rangle` or `tw@mk@\langle key \rangle\langle os \rangle`, if the appearance differs between Mac and Windows. To use this macro we assume that the `tw@mk@\langle key \rangle` commands are defined.

```latex
\NewDocumentCommand{\tw@make@key@macro}{s m}{%
  \IfBooleanTF{#1}{% \
    \expandafter\providecommand\csname tw@mk@string#2\endcsname{\expandonce{\maxsizebox{!}{1.8ex}{\@nameuse{tw@mk@tw@mk@string#2@tw@mk@os}}}}% \
  }% 
  \expandafter\providecommand\csname tw@mk@string#2mac\endcsname{\expandonce{\maxsizebox{!}{1.8ex}{\@nameuse{tw@mk@tw@mk@string#2@mac}}}}% 
  \expandafter\providecommand\csname tw@mk@string#2win\endcsname{\expandonce{\maxsizebox{!}{1.8ex}{\@nameuse{tw@mk@tw@mk@string#2@win}}}}%}
```
The last helping macro is \tw@define@mackey. We use it to execute code depending on the mackeys option.

Next thing to do is to set up some TikZ-styles.

Now we are prepared to generate the key macros. I will be nearly the same way for all keys. Step one is to build a \tw@mk@⟨key⟩ macro and then we define the user-level command \langle key⟩.

It's a little more complicated if the appearance should differ depending on the OS: The first step again is to define \tw@mk@⟨key⟩@mac and \tw@mk@⟨key⟩@win. And then use the starred version \tw@make@key@macro* which creates \langle key⟩ that depends on the os option, \langle key⟩@mac and \langle key⟩@win, that are not affected by os.
\capslock

\tw@make@key@box{capslock@mac}{%
  \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
  \draw (0.3ex,0.7ex) -- (1.1ex,0.7ex) -- (1.1ex,1.2ex) -- %
  (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
  (0.3ex,1.2ex) -- cycle;
  \draw (0.3ex,0) rectangle (1.1ex,0.4ex);
  \end{tikzpicture}%
}\tw@make@key@box{capslock@win}{%
  \begin{tikzpicture}[yscale=-1,yshift=-1.8ex,menukeys key symbol]
  \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
  (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
  (0.3ex,1.2ex) -- cycle;
  \end{tikzpicture}%
}\tw@make@key@macro*{\capslock}

Here are the other macros:

\tab

\tw@make@key@box{tab@mac}{%
  \begin{tikzpicture}[yshift=0.6ex,menukeys key symbol]
  \draw [->] (0,0) -- (1em,0);
  \draw (1em,-0.35ex) -- (1em,0.35ex);
  \end{tikzpicture}%
}\tw@make@key@box{tab@win}{%
  \begin{tikzpicture}[yshift=0.1ex,menukeys key symbol]
  \draw [->] (0.2em,0) -- (1.2em,0);
  \draw (1.2em,-0.35ex) -- (1.2em,0.35ex);
  \draw [<->] (0,1ex) -- (1em,1ex);
  \draw (0,0.65ex) -- (0,1.35ex);
  \end{tikzpicture}%
}\tw@make@key@macro*{\tab}

\esc
\oldesc
\tw@mk@esc@win{Esc}
\tw@define@mackey{%
\tw@mk@esc@mac{esc}
}\tw@make@key@box{esc@mac}{%
  \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
  \draw [->] (0.5ex,0.5ex) -- ++(135:1.1ex);
  \draw (0.5ex,0.5ex) ++(105:0.6ex) arc (105:-195:0.6ex);
  \end{tikzpicture}%
}\tw@make@key@macro*{\esc}
\def\tw@mk@oldesc@win{Esc}
\tw@define@mackey{%
\def\tw@mk@oldesc@mac{esc}
}\tw@make@key@box{oldesc@mac}{%
\begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
\draw [>-] (0.5ex,0.5ex) -- ++(45:1.1ex);
\draw (0.5ex,0.5ex) ++(15:0.6ex) arc (15:-285:0.6ex);
\end{tikzpicture}%
}%
\tw@make@key@macro*{\oldesc}

\ctrl
\providecommand\ctrlname{Ctrl}
\def\tw@mk@ctrl@win{\ctrlname}
\def\tw@mk@ctrl@mac{ctrl}
\tw@make@key@macro*{\ctrl}

\Alt
\AltGr
\def\tw@mk@Alt@win{\Alt}
\tw@define@mackey{%
\def\tw@mk@Alt@mac{alt}%
}\tw@make@key@box{Alt@mac}{%
\begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
\draw (0,1ex) -- (0.5ex,1ex) -- (1ex,0.3ex) -- (1.8ex,0.3ex);
\draw (0.8ex,1ex) -- (1.8ex,1ex);
\end{tikzpicture}%
}%
\tw@make@key@macro*{\Alt}
\providecommand*{\AltGr}{Alt\,Gr}

\cmd
\def\tw@mk@cmd@win{%
\tw@mk@warning{\string\cmd' only for Mac!}%
}%
\tw@define@mackey{%
\def\tw@mk@cmd@mac{cmd}%
}\tw@make@key@box{cmd@mac}{%
\begin{tikzpicture}[yshift=-0.15ex,menukeys key symbol]
\draw (0.5ex,0.7ex) -- (0.5ex,1.25ex) arc (0:270:0.25ex) -- %
(1.25ex,1ex) arc (-90:180:0.25ex) -- (1ex,0.25ex) %
arc (-180:90:0.25ex) -- (0.25ex,0.5ex) arc (90:360:0.25ex) %
-- cycle;
\end{tikzpicture}%
}%
\providecommand{\Space}{\expandonce{\rule{3em}{0pt}}}
\newcommand{\spacename}{Space}
\providecommand{\SPACE}{\expandonce{\rule{2em}{0pt}\spacename\rule{2em}{0pt}}}
\return
\tw@make@key@box{return@mac}{\begin{tikzpicture}[yshift=0.25ex,menukeys key symbol]
\draw [->, rounded corners=0.2ex] (1.25ex,1ex) -| (2ex,0) -- (0,0);
\end{tikzpicture}}
\tw@make@key@box{return@win}{\begin{tikzpicture}[menukeys key symbol]
\draw [->] (1ex,1.25ex) |- (0,0);
\end{tikzpicture}}
\tw@make@key@macro{\return}
\enter
\def\tw@mk@enter@win{Enter}
\tw@make@key@box{enter@mac}{\begin{tikzpicture}[menukeys key symbol]
\draw (0,0) -- (0.5ex,0.5ex) -- (1ex,0);
\draw (0,0.55ex) -- (1ex,0.55ex);
\end{tikzpicture}}
\tw@make@key@macro{\enter}
\winmenu
\def\tw@mk@winmenu@mac{\tw@mk@warning{\string\winmenu only for Windows!}}
\backspace
\tw@make@key@box{backspace}{\begin{tikzpicture}[yshift=0.65ex,menukeys key symbol]
\draw (0,0) rectangle (1.5ex,1.8ex);
\draw (0.25ex,1.4ex) -- +(1ex,0);
\draw (0.25ex,1ex) -- +(1ex,0);
\draw (0.25ex,0.6ex) -- +(1ex,0);
\end{tikzpicture}}
\tw@make@key@macro{\winmenu}
Lastly we define the arrow macros:

\begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
\draw [-] (0,0) -- (0,0.8em);
\end{tikzpicture}
\arrowkey And the \arrowkey macro that get's it's direction as argument.

\newcommand{\arrowkey}{\IfStrEq{^}{#1}{\arrowkeyup}{\IfStrEq{v}{#1}{\arrowkeydown}{\IfStrEq{<}{#1}{\arrowkeyleft}{\IfStrEq{>}{#1}{\arrowkeyright}{\tw@mkerror{Wrong value '{#1}' for \string\arrowkey}'\MessageBreak Possible values are '{^}', '{v}', '{<}' or '{>'}}}}}}

Close the \iftw@mk@definekeys
7 Change history

v1.0
General: Initial version ........... 1

v1.1
\directory: Renamed \path to \directory because it crashes with biblatex ............. 33
General: Improved manual .......... 1
Load xcolor before menukeys. .... 15

v1.1a
\newmenumacro: Added a line to make a new macro robust. ....... 32
\tw@define@menu@macro\: Fixed minor bug, that causes a warning about robustifying (issue #23), by deleting the line to make the command robust. ......................... 30

v1.2
\tw@define@menu@macro\: Added \leavevmode .................. 30
\protect@edef .................. 30
General: Added \normalsize before symbol definitions to make the \text command available .......... 1
Added \SPACE and \spacename . 1
Fixed GitHub issues #9, #10, #11, #13, #17, #24 and #26 . 1
Tidy up version and date .......... 1

v1.2a
General: Added braces to the \tikz macro since the parser seems to crash with babel\’s french option otherwise. ........... 1
Replaced obsolete \tikzstyles . 1

v1.2c
\tw@define@menu@macro\: Replaced \protect@edef by \def .................. 30

v1.3
General: Added Ti\kZ-styles for the key symbols. ............... 1

v1.4
Improved key symbols. ............ 1
\backdel: Added \backdel ........ 38
\oldesc: Fixed direction of \escmac; added \oldesc .......... 35
General: Extended color theme features. ......................... 1
The \path... styles now use the text color of the selected color theme (fix issue #16). ........ 1

v1.5
General: New option hyperrefcolorlinks .......... 18
\newmenumacro: use \NewDocumentCommand .......... 32
\providemenumacro: use \ProvideDocumentCommand .. 32
\renewmenumacro: use \RenewDocumentCommand .... 32
\tw@define@menu@macro\: Don\’t use \NewDocumentCommand ... 30
General: hyperrefcolorlinks obsolete ............... 18
Don\’t load catoptions .......... 15
Load order no longer important 5

v1.6
\newmenumacro: default handled by \tw@define@menu@macro .......... 32
\providemenumacro: default handled by \tw@define@menu@macro .......... 32
\renewmenumacro: default handled by \tw@define@menu@macro ..... 32
\tw@define@menu@macro\: Handles default input separator. .......... 30
\tw@define@menu@macro\: No x-type expansion on the separator to call the loop .......... 30
Renamed from \tw@define@menu@macro .......... 30
8 Macro index

Numbers written in bold face refer to the page where the corresponding entry is described; italic numbers refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

\ifdefinable ...................................... 625
\tw@hyphenatepaths@warnigfalse ........................ 573
\tw@hyphenatepaths@warnigtrue ........................ 561
\_ ........................................ 28, 32, 36, 42, 57, 58, 67, 68

A
\Alt .............................................. 837
\AltGr ............................................ 837
guarke (style) .................................... 7
guarlemenus (style) ................................ 6
\arrowkey ..................................... 14, 963
\arrowkeydown ................................... 936, 965
\arrowkeyleft ................................... 936, 966
\arrowkeyright .................................. 936, 967
\arrowkeyup .................................... 936, 964

B
\backdel ......................................... 907
\backspace ...................................... 901
\blacknwhite (theme) ......................... 12
\bool ............................................. 61

C
\capslock ........................................ 778
\changemenucolor .................................. 13, 130
\changemenuscolorscheme ..................... 11, 151
\changemenuelement ............................... 11, 320, 598, 604, 610, 616
\cmd ............................................. 850
\color ........................................... 333, 361, 366, 370, 373, 392, 397, 401, 404, 422, 427, 431, 434, 450, 474, 494, 517, 538, 540, 549, 577
Color themes:
  \blacknwhite ................................ 12
  \gray ........................................ 12
\copymenucolorscheme ......................... 13, 130
\copymenustyle ................................ 11, 302, 597, 603, 609, 615
\cs .............................................. 10, 11, 12, 14, 18, 25, 28, 32, 36, 42, 54, 55
\ctrl ........................................... 833
\ctrlname ..................................... 14, 833, 834

D
\CurrentMenuElement .............................

D
\currentmenuelement .............................

E
\CurrentMenuElement .............................

F
\font ........................................... 576

G
\gray (theme) .................................... 12
\group ......................................... 47, 50

H
\hypersetup ...................................... 108
\hyphenatepaths (style) ...................... 9
\hyphenatepathsblackfolder (style) ........ 9
\hyphenatepathsblackfolder (style) ........ 9
\if@tw@hyphenatepaths@warnig ............... 560, 570

I
\if\tw@hyphenatepaths@warnigfalse ........ 635
\iftw@mk@definekeys ......................... 720
\iftw@mk@definemenuacros ................. 715
\iftw@mk@hyperrefcolorlinks ............. 105
\int ........................................... 23, 45, 51, 60, 63
<table>
<thead>
<tr>
<th>Section</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>5, 716</td>
</tr>
<tr>
<td>L</td>
<td>20, 22, 23, 24, 25, 26, 27, 30, 34, 38, 39, 40, 44, 45, 48, 51, 59, 60, 61, 63, 64, 65</td>
</tr>
<tr>
<td>M</td>
<td>5, 14</td>
</tr>
<tr>
<td>N</td>
<td>12, 119, 166, 167</td>
</tr>
<tr>
<td>O</td>
<td>809</td>
</tr>
<tr>
<td>P</td>
<td>109</td>
</tr>
<tr>
<td>R</td>
<td>353, 384, 414, 445, 453, 465, 478, 489, 497, 508, 520, 529, 544</td>
</tr>
<tr>
<td>S</td>
<td>20, 22, 26, 27, 30, 34, 39, 40, 44, 48, 54, 59, 61, 64</td>
</tr>
<tr>
<td>T</td>
<td>794</td>
</tr>
</tbody>
</table>

Styles:
- angularkeys ........................................... 7
- angularmenus ......................................... 6
- hyphenatepathswithblackfolder .................. 9
- hyphenatepathswithfolder ....................... 9
- hyphenatepaths ...................................... 9
- menus ................................................ 6
- pathswithblackfolder ............................... 8
- pathswithfolder ..................................... 8
- paths .................................................. 8
- roundedkeys .......................................... 7
- roundedmenus ........................................... 6
- shadowedangularkeys ................................. 7
- shadowedroundedkeys ................................ 7
- typewriterkeys ....................................... 8

Option Definitions:
- definekeys ............................................. 5
- definemenumacros .................................... 5
- mackeys .............................................. 5, 14
- os ..................................................... 5
- os (option) ............................................ 5

Pass Options To Package ................................ 109
Path Definitions:
- pathswithblackfolder (style) ...................... 8
- pathswithfolder (style) ............................. 8
- \pgf ................................................... 13
- \protected ............................................ 627
- \providemenumacro .................................. 14, 696
- \providemenustyle ................................. 12, 233, 288
- \providemenustylesimple ........................ 12, 233, 253
<table>
<thead>
<tr>
<th>Command</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tw@make@color@theme</td>
<td>71, 121, 128</td>
</tr>
<tr>
<td>\tw@make@key@box</td>
<td>721, 770, 778, 786, 794, 800, 813, 825, 841, 856, 869, 875, 882, 892, 901, 912, 926, 936, 943, 950, 957</td>
</tr>
<tr>
<td>\tw@make@key@macro</td>
<td>721, 770, 778, 786, 794, 800, 813, 825, 841, 856, 869, 875, 882, 892, 901, 912, 926, 936, 943, 950, 957</td>
</tr>
<tr>
<td>\tw@menu@list</td>
<td>680, 684, 689</td>
</tr>
<tr>
<td>\tw@mk@Alt@mac</td>
<td>839</td>
</tr>
<tr>
<td>\tw@mk@Alt@win</td>
<td>837</td>
</tr>
<tr>
<td>\tw@mk@backdel@mac</td>
<td>924</td>
</tr>
<tr>
<td>\tw@mk@backdel@win</td>
<td>922</td>
</tr>
<tr>
<td>\tw@mk@cmd@mac</td>
<td>854</td>
</tr>
<tr>
<td>\tw@mk@cmd@win</td>
<td>850</td>
</tr>
<tr>
<td>\tw@mk@ctrl@mac</td>
<td>835</td>
</tr>
<tr>
<td>\tw@mk@ctrl@win</td>
<td>834</td>
</tr>
<tr>
<td>\tw@mk@del@mac</td>
<td>910</td>
</tr>
<tr>
<td>\tw@mk@del@win</td>
<td>908</td>
</tr>
<tr>
<td>\tw@mk@enter@win</td>
<td>881</td>
</tr>
<tr>
<td>\tw@mk@error</td>
<td>71, 98, 102, 123, 134, 147, 153, 157, 241, 276, 305, 317, 322, 336, 643, 700, 968</td>
</tr>
<tr>
<td>\tw@mk@esc@mac</td>
<td>811</td>
</tr>
<tr>
<td>\tw@mk@esc@win</td>
<td>809</td>
</tr>
<tr>
<td>\tw@mk@exp@Nnno</td>
<td>11, 636</td>
</tr>
<tr>
<td>\tw@mk@gobble@args</td>
<td>82, 243, 264, 278, 299</td>
</tr>
<tr>
<td>\tw@mk@iflastindris</td>
<td>18, 653</td>
</tr>
<tr>
<td>\tw@mk@indrisloop</td>
<td>55, 689</td>
</tr>
<tr>
<td>\tw@mk@indrisnr</td>
<td>23, 655, 663</td>
</tr>
<tr>
<td>\tw@mk@mackeys</td>
<td>101, 757, 758</td>
</tr>
<tr>
<td>\tw@mk@oldesc@mac</td>
<td>823</td>
</tr>
<tr>
<td>\tw@mk@oldesc@win</td>
<td>821</td>
</tr>
<tr>
<td>\tw@mk@os</td>
<td>97, 735</td>
</tr>
<tr>
<td>\tw@mk@string</td>
<td>71, 106, 261, 296, 571, 711, 851, 890</td>
</tr>
<tr>
<td>\tw@mk@tempa</td>
<td>10, 629, 685</td>
</tr>
<tr>
<td>\tw@mk@tempb</td>
<td>80, 868, 889</td>
</tr>
<tr>
<td>\tw@mk@test@input@sep</td>
<td>625, 627, 678</td>
</tr>
<tr>
<td>\tw@mk@trimspaces</td>
<td>10, 629, 685</td>
</tr>
<tr>
<td>\tw@mk@warning</td>
<td>71</td>
</tr>
<tr>
<td>\tw@mk@warning@noline</td>
<td>14, 628</td>
</tr>
<tr>
<td>\tw@mk@winmenu@mac</td>
<td>889</td>
</tr>
<tr>
<td>\tw@mk@worksetTF</td>
<td>340</td>
</tr>
<tr>
<td>\tw@typewriterkeys@curr@elem</td>
<td>532, 538, 540</td>
</tr>
<tr>
<td>\usemenucolor</td>
<td>11, 163, 333, 341, 342, 343, 361, 366, 370, 373, 392, 397, 401, 404, 422, 427, 431, 434, 450, 453, 468, 474, 478, 494, 497, 511, 517, 520, 538, 540, 544, 549, 553, 577, 582, 599, 605, 611, 617</td>
</tr>
<tr>
<td>\usemenucolor</td>
<td>889</td>
</tr>
<tr>
<td>\usemenucolor</td>
<td>889</td>
</tr>
</tbody>
</table>