The `eqlist` package*

Martin Väth
vaeth@mathematik.uni-wuerzburg.de

2002/08/15

Abstract

With this package you can write lists with equal indentation. This package requires the `eqparbox` package.

1 Changes

v2.1 (*2002/08/18*) Added `\longitem` and the related `\eqlistauto` mechanism. The latter was suggested by Rolf Niepraschk `Rolf.Niepraschk@ptb.de`. Changed default of `\eqlistlabel` and gave some comments on spaces at the end of labels in the documentation. Documented special usage of `\makelabel`.

v1.2 (*2001/08/17*) Added the `{Eqlist}` and `{Eqlist*}` environments.

v1.1 (*2001/08/16*) First release.

2 Description

This package provides a list environment which sets a description-like list but with the difference that the indentation corresponds to the longest item of the list. The usage is simply

`eqlist`

```
\begin{eqlist}[⟨optional modifications⟩]
\item[First item] Text
\item[Second item] Text
\longitem[A special very long item] Text
...
\end{eqlist}
```

`eqlist*` and there is also the environment `{eqlist*}` which is similar but has slightly different defaults (which make the list appear more compact). There is also the alternative call

Eqlist

```
\begin{Eqlist}[⟨optional modifications⟩]{⟨tag⟩}
\item[First item] Text
\item[Second item] Text
\longitem[A special very long item] Text
...
\end{eqlist}
```

Eqlist*

and a corresponding `{Eqlist*}` environment.

All texts within the list are indented by the length of the largest label (i.e. `\item` entry) plus `\labelsep`. For the `Eqlist` or `Eqlist*` environment, all lists with the same ⟨*tag*⟩ are treated equally in the sense that the indentation of these lists is determined by the largest `\item` of all these lists. In this case, you may also use the ⟨*tag*⟩ for the `eqparbox` package to read or modify the length of the largest `\item` (which is internally treated as a `\eqparbox`).

\longitem

`\longitem` is like `\item`, but the corresponding label is excluded from the calculation of the longest `\item`. The intention of `\longitem` is to allow exceptionally long labels to occur without forcing a corresponding extreme indentation of the whole list. If you want LaTeX to decide automatically whether `\longitem` or `\item` should be used, you can use the `\eqlistauto` mechanism which is described later.

The ⟨*optional modifications*⟩ are any commands which are used to initialize the list: You can modify here essentially the same variables as for any LaTeX 2_ε

\eqlistinit
\eqliststarinit

list. If this argument is not given, the default initializations `\eqlistinit` respectively `\eqliststarinit` (for `{eqlist*}` and `{Eqlist*}`) are used: You can just modify these definitions to change the defaults. If you have given the argument ⟨*optional modifications*⟩ and additionally want to use the defaults, you have to include the command `\eqlistinit` respectively `\eqliststarinit` into the argument ⟨*optional modifications*⟩ (see the examples below).

\eqlistinitpar

The macros `\eqlistinit` and `\eqliststarinit` both call `\eqlistinitpar` which sets the values for `\parindent` and `\parskip` to the values outside the list (this is not standard in LaTeX 2_ε, but I prefer this style; if you do not like this, use `\let\eqlistinit\relax`).

\topsep
\itemsep
\partopsep

Currently, this is all which is done by `\eqlistinit`; for `\eqliststarinit` additionally the values of `\topsep` and `\itemsep` are set to 0. Note that currently `\partopsep` is not changed from the LaTeX 2_ε default. Note that the latter is by default positive which means that if you are in vertical mode before the list (e.g. if you have a `\par` in front of the list), you get slightly more space above the list.

\labelwidth
\leftmargin
\labelsep

Before ⟨*optional modifications*⟩ (or `\eqlistinit` respectively `\eqliststarinit`) are expanded, the length of the largest label is already stored in `\labelwidth` and `\leftmargin`. After your modifications, `\labelsep` is added to the actual value of `\leftmargin`.

\makelabel

As usual, the layout of `\item`s is done by the command `\makelabel`. If you want, you can change the default initialization of this command in the ⟨*optional modifications*⟩ argument. However, it is *not* admissible to redefine this macro within the list. If you really want to change the layout of `\item`s in the middle of a list, you can initialize `\makelabel` to expand to another command whose definition you can change within the list instead of `\makelabel` (an example will be given later). The default value of `\makelabel` is the content of `\eqlistlabel` which in turn is by default defined with

\makelabel
\eqlistlabel

*This file has version number 2.1, last revised 2002/08/15.

```
\def\eqlistlabel#1{#1}
```

In version 1.1 and 1.2 of this package, the default was different: A space was added at the end. This has been changed, because by a bug (or feature?) in `\eqparbox` spaces at the end of an `\item` are ignored anyway. If you want to force a space at the end which will not be ignored, you have to hide it in a box:

```
\def\eqlistlabel#1{\mbox{#1 }}
```

will force a space at the end of every `\item`.

There is a special mechanism provided which will automatically decide for an `\item` depending on the length of its label whether it is treated as normal or whether it should be treated like a `\longitem`: If you want to use this mechanism, you only have to insert the command

`\eqlistauto` `\eqlistauto{`⟨*maximal length*⟩`}`

either inside the ⟨*optional modifications*⟩ argument or within the list. Here, ⟨*maximal length*⟩ must be in a format which can be used within TEX's internal `\ifdim` command. It describes the maximal length of the label such that the corresponding `\item` will be treated as usual—for longer labels the corresponding `\item` will be treated as a `\longitem`. If you want to switch off the `\eqlistauto`-mechanism again, you can use the command

`\eqlistnoauto` `\eqlistnoauto`

The commands `\eqlistauto` and `\eqlistnoauto` need not occur in matching pairs, and they can also be used several times within the same list: Only the latest of the corresponding commands takes effect for a corresponding `\item`.

Unfortunately, the `\eqlistauto` mechanism has a disadvantage: As long as it is in effect, the corresponding `\makelabel`s for the `\item`s are always executed twice (once to calculate the length and once for the actual typesetting). This can cause problems if e.g. counters in `\item`s are increased.

The `\eqlistauto` mechanism only effects the `\item` command, not `\longitem`: Even if the `\eqlistauto` mechanism is active, you can use `\longitem` which will have its original meaning (independent of the length of the label). This means in particular that the `\makelabel` is executed only once by `\longitem`, no matter whether the `\eqlistauto` mechanism is active or not.

3 Examples

```
\begin{eqlist}[\eqliststarinit\def\makelabel#1{\bfseries#1:}\labelsep1em]
\item[Short label] Descriptive text
\item[A longer label] Descriptive text
\longitem[An exceptionally long label] Descriptive text
\item[Short again] Descriptive text
\end{eqlist}
```

will produce an output like

Short label: Descriptive text
A longer label: Descriptive text
An exceptionally long label: Descriptive text
Short again: Descriptive text

The same output can be obtained using the `\eqlistauto` mechanism

```
\begin{eqlist}[\eqliststarinit
   \def\makelabel#1{\bfseries#1:}\labelsep1em\eqlistauto{3cm}]
\item[Short label] Descriptive text
\item[A longer label] Descriptive text
\item[An exceptionally long label] Descriptive text
\item[Short again] Descriptive text
\end{eqlist}
```

or by using the `\eqlistauto` mechanism only locally:

```
\begin{eqlist}[\eqliststarinit\def\makelabel#1{\bfseries#1:}\labelsep1em]
\item[Short label] Descriptive text
\item[A longer label] Descriptive text
\eqlistauto{0pt}
\item[An exceptionally long label] Descriptive text
\eqlistnoauto
\item[Short again] Descriptive text
\end{eqlist}
```

The next example demonstrates how one can change the layout of labels within the list. Recall that it is forbidden to redefine `\makelabel`.

```
\begin{eqlist}[\eqliststarinit
  \def\mylabel#1{\bfseries#1:}\def\makelabel{\mylabel}\labelsep1em]
\item[First label] Descriptive text
\item[Second label] Descriptive text
\def\mylabel#1{\slshape#1:}
\item[First new-style label] Descriptive text
\longitem[Second new-style label which is long] Descriptive text
\end{eqlist}
```

The above example will produce an output as follows.

First label: Descriptive text
Second label: Descriptive text
First new-style label: Descriptive text
Second new-style label which is long: Descriptive text

4 Implementation

```
1
2 \typeout{eqlist.sty by M. Vaeth: Revision 2.1}
3 \NeedsTeXFormat{LaTeX2e}
4 \ProvidesPackage{eqlist}[2002/08/15 v2.1]
5
6 \RequirePackage{eqparbox}
7
```

eqlist

```
8 \newenvironment{eqlist}[1][\eqlistinit]{\eql@start{#1}}{\eql@end}
```

eqlist*

```
9 \newenvironment{eqlist*}[1][\eqliststarinit]{\eql@start{#1}}{\eql@end}
```

Eqlist

```
\newenvironment{Eqlist}[2][\eqlistinit]{\eql@startp{#1}{#2}}{\eql@end}
```

Eqlist*

```
\newenvironment{Eqlist*}[2][\eqliststarinit]{\eql@startp{#1}{#2}}{\eql@end}

```

\eqlistinitpar

```
\ifx\eqlistinitpar\undefined
  \def\eqlistinitpar{\relax\listparindent\parindent\relax\parsep\parskip\relax}
\fi
```

\eqlistinit

```
\ifx\eqlistinit\undefined
  \def\eqlistinit{\eqlistinitpar}
\fi
```

\eqliststarinit

```
\ifx\eqliststarinit\undefined
  \def\eqliststarinit{\topsep0pt\relax\itemsep0pt\relax%\partopsep0pt\relax
    \eqlistinitpar}
\fi
```

\eqlistlabel

```
\ifx\eqlistlabel\undefined
  \def\eqlistlabel#1{#1}
\fi

```

\eql@cnt The counter \eql@cnt is used to generate "unique" names for the labels.

```
\newcount\eql@cnt\relax\eql@cnt=0

```

\eql@start The macro \eql@start advances the counter \eql@cnt (globally!) and then starts a list, where for the initialization \eql@mainprep is called with the corresponding "unique" name and the argument.

```
\long\def\eql@start#1{\global\advance\eql@cnt by1\begin{list}{}{\expandafter
  \eql@mainprepn\expandafter{\romannumeral\eql@cnt}{#1}}}
```

\eql@startp The macro \eql@startp is similar to \eql@start with the difference that the first argument is used as the "unique" name instead of a counter. Thus, \eql@startp starts a list, where for the initialization \eql@mainprep is used with the corresponding arguments.

```
\long\def\eql@startp#1#2{\begin{list}{}{\eql@mainprep{#2}{#1}}}
```

\eql@end

```
\def\eql@end{\end{list}}

```

\eql@mainprep In \eql@mainprep, we allow the environment-specific commands and initialize

\eql@makelabel \eql@makelabel

which will become later our actual `\makelabel`. We also save this definition into

`\eql@normal` `\eql@normal`

and (as always if `\eql@makelabel` is changed non-temporary) we also save it into

`\eql@current` `\eql@current`.

Finally, `\makelabel` is initialized to `\eqlistlabel`, and the length of the label (and the left margin) is initialized with the length of the corresponding eqparbox with the "unique" name `#1`. Then the (default or user) initialization `#2` is executed. After this, `\makelabel` is replaced by `\eql@makelabel`. The "user" definition of `\makelabel` is before saved into

`\eql@long` `\eql@long`.

```
\long\def\eql@mainprep#1#2{\let\longitem\eql@longitem
  \let\eqlistauto\eql@auto
  \let\eqlistnoauto\eql@noauto
  \def\eql@makelabel##1{\eqparbox[b]{#1}{\eql@long{##1}}\hfil}%
  \let\eql@normal\eql@makelabel
  \let\eql@current\eql@makelabel
  \setlength{\labelwidth}{\eqboxwidth{#1}}%
  \setlength{\leftmargin}{\labelwidth}%
  \let\makelabel\eqlistlabel
  #2\addtolength{\leftmargin}{\labelsep}%
  \let\eql@long\makelabel\def\makelabel{\eql@makelabel}}
```

`\eql@mainprepn` Despite of its definition, `\eql@mainprepn` actually expects two arguments: `\eql@mainprepn` does the same as `\eql@mainprep` with the difference that the first argument should be a unique roman number instead of a "unique" name: The actual name is then obtained by prepending the text `eqlistbox` to the number. Note that the number must actually be roman, since otherwise (some release(s) of) the `eqparbox` package becomes confused.

```
\def\eql@mainprepn#1{\eql@mainprep{eqlistbox#1}}

```

`\longitem` `\longitem` is essentially a call of `\item` with the difference that we redefine `\eql@makelabel` temporarily. To avoid usage outside of the intended environment, the actual code is put into

`\eql@longitem` `\eql@longitem`.

```
\newcommand{\longitem}{\eql@illlegal\longitem\item}
\def\eql@longitem[#1]{\let\eql@makelabel\eql@long
  \item[#1]\let\eql@makelabel\eql@current}

```

`\eqlistauto` To avoid usage outside of the intended environment, the actual code is put into

`\eql@auto` `\eql@auto`.

```
\newcommand{\eqlistauto}[1]{\eql@illegal\eqlistauto}
\def\eql@auto#1{\def\eql@makelabel##1{\setbox0\hbox{\eql@long{##1}}%
    \ifdim#1>\wd0\relax
      \expandafter\eql@normal
```

```
    \else
      \expandafter\eql@long
    \fi{##1}}%
  \let\eql@current\eql@makelabel}
```

\eqlistnoauto To avoid usage outside of the intended environment, the actual code is put into

\eql@noauto \eql@noauto.

```
\newcommand{\eqlistnoauto}{\eql@illegal\eqlistnoauto}
\def\eql@noauto{\let\eql@makelabel\eql@normal\let\eql@current\eql@makelabel}
```

\eql@illegal Write error message in case a command is used outside of the intended environment.

```
\def\eql@illegal#1{\errmessage{\string#1
  can only be used in eqlist or Eqlist environment}}

```