The **comment** package*

Victor Eijkhout
`victor@eijkhout.net`

August 2016

1 Purpose:

Selectively in/exclude pieces of text: the user can define new comment versions, and each is controlled separately. Special comments can be defined where the user specifies the action that is to be taken with each comment line.

Plain TEX support has been phased out.

As of 3.8 the package will increasingly use eTEX features, for instance to solve Unicode support issues.

2 Usage:

The '`comment`' environment is defined by default: all text included between

```
\begin{comment}
 ...
\end{comment}
```

is discarded.

The opening and closing commands should appear on a line of their own. No starting spaces, nothing after it. This environment should work with arbitrary amounts of comment, and the comment can be arbitrary text.

Other '`comment`' environments are defined by and are selected/deselected with

```
\includecomment{versiona}
\excludecoment{versionb}
```

These environments are used as

```
\begin{versiona}
 ...
\end{versiona}
```

with the opening and closing commands again on a line of their own.

Note: for an included comment, the `\begin` and `\end` lines act as if they don't exist. In particular, they don't imply grouping, so assignments &c are not local.

Trick for short in/exclude macros (such as `\maybe{this snippet}`):

---

* This manual corresponds to comment v3.8 of July 2016.

```
\includecomment{cond}
\newcommand{\maybe}[1]{}
\begin{cond}
\renewcommand{\maybe}[1]{#1}
\end{cond}
```

3 Special comments

It is possible to make highly customized versions of the comment environment. Special comments are defined as

`\specialcomment{`⟨*name*⟩`}{`⟨*before commands*⟩`}{`⟨*after commands*⟩`}`

where the second and third arguments are executed before and after each comment block. You can use this for global formatting commands.

To keep definitions &c local, you can include `\begingroup` in the '⟨*before commands*⟩' and `\endgroup` in the '⟨*after commands*⟩'.

Example:

```
\specialcomment{smalltt}
    {\begingroup\ttfamily\footnotesize}{\endgroup}
```

Special comments are automatically included.

The comment environments use two auxiliary commands. You can get nifty special effects by redefining them.

3.1 The cutfile

The commented text is written to an external file, the 'cutfile'. Default definition:

```
\def\CommentCutFile{comment.cut}
```

Included comments are processed like this:

```
\def\ProcessCutFile{\input{\CommentCutFile}\relax}
```

and excluded files have

```
\def\ProcessCutFile{}
```

- By redefining the name of the cutfile, the value of the macro `\ommentCutFile`, it becomes possible to have nested comment environments.
- If you are writing a textbook, you could have the answers to exercises in your source, but write them to file rather than formatting them:

```
\generalcomment{answer}
  {\begingroup
   \edef\tmp{\def\noexpand\CommentCutFile
                {answers/\chapshortname-an\noexpand\arabic{excounter}.tex}}\tmp
   \def\ProcessCutFile{}}
  {\ifIncludeAnswers \begin{quote}
   \leavevmode
   \hbox{\kern-\unitindent
         \textbf Solution to exercise \arabic{chapter}.\arabic{excounter}.\hspace{1em}}
     \ignorespaces\it
   \input{\CommentCutFile}
   \end{quote}\fi
   \endgroup}
```

3.2 Comment inclusion

The inclusion of the comment is done by `\ProcessCutFile`, so you can redefine that:

```
\specialcomment
  {mathexamplewithcode}
  {\begingroup\def\ProcessCutFile{}} % arg1
  {\verbatiminput{\CommentCutFile}   % arg2
   \endgroup
   This gives:
   \begin{equation} \input{\CommentCutFile} \end{equation}
  }
```

The idea here is to disable inclusion of the file, but include it in the after commands, in display math.

3.3 Processing each line

You can also apply processing to each line. By defining a control sequence

```
\def\Thiscomment##1{...}
```

in the before commands the user can specify what is to be done with each comment line. If something needs to be written to file, use `\WriteCommentLine{the stuff}` Example:

```
\specialcomment
  {underlinecomment}
  {\def\ThisComment##1{\WriteCommentLine{\underline{##1}\par}}
   \par}
  {\par}
```

3.4 More examples

```
\newcount\comlines
\specialcomment{countedcomment}
 {\comlines=0\relax \def\ProcessCutFile{}%
  \def\ThisComment##1{\global\advance\comlines1\relax}}
 {**Comment: \number\comlines\ line(s) removed**}
```

```
\specialcomment
    {underlinecomment}
    {%
      \def\ProcessCutFile{\input{\CommentCutFile}\relax}
      \def\ThisComment##1{\WriteCommentLine{u: \underline{##1}\par}}
      \par
    }
    {\par}
```

4 Unicode support

Unicode support works if you use eTEX, which is for instance the case if you use pdflatex. You need the following lines:

```
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
```
in your preamble.

5 Change log

5.1 Changes in version 3.1

- updated author's address
- cleaned up some code
- trailing contents on `\begin{`⟨*env*⟩`}` line is always discarded even if you've done `\includecomment{`⟨*env*⟩`}`
- comments no longer define grouping!! you can even
  ```
  \includecomment{env}
  \begin{env}
  \begin{itemize}
  \end{env}
  ```
  Isn't that something...
- included comments are written to file and input again.

5.2 Changes in 3.2

- `\specialcomment` brought up to date (thanks to Ivo Welch).

5.3 Changes in 3.3

- updated author's address again
- parametrised `\CommentCutFile`

5.4 Changes in 3.4

- added GNU public license
- added `\processcomment`, because Ivo's fix (above) brought an inconsistency to light.

5.5 Changes in 3.5

- corrected typo in header.
- changed author email
- corrected `\specialcomment` yet again.
- fixed excludecomment of an earlier defined environment.

5.6 Changes in 3.6

- The 'cut' file is now written more verbatim, using `\meaning`; some people reported having trouble with ISO latin 1, or `umlaute.sty`.
- removed some `\newif` statements. Has this suddenly become `\outer` again?

5.7 Changes in 3.8

T1 font encoding is now supported. See t1test.tex.