Documentation for xcomment.sty*

Timothy Van Zandt Timothy.VAN-ZANDT@insead.edu

May 14, 2010

*Documentation revised by Herbert Voß. This file borrows much from verbatim.sty, v.1.4c (90/10/18) Copyright (C) 1989, 1990 by Rainer Schöpf.

Contents

Abstract

The LaTeX style option verbatim.sty, by Rainer Schöpf, allows one to redefine an environment to be a comment, and thereby selectively omit certain environments when typesetting a document. Suppose instead that one wants to typeset only selected environments? For example, one might want to print only a document's tables and figures, without having to enclose all the text outside these environments within comments. This style option allows such selection.

1 Usage notes

This style file defines a new environment, `xcomment`, which permits one to typeset only selected environments, without having to enclose all the text outside these environments within comments. The main interest in such a feature is that it allows document styles to have great control over what parts of a document are typeset, thus extending the modularity of LaTeX. For example, this option was originally written for use in a document style for seminar notes and slides. The main text of the input file consists of the notes for a seminar, and each slide goes in a `slide` environment. A simple modification of the style options allows one to typeset only the slides, only the notes, or both together, in a variety of styles.

The `xcomment` environment has as a single mandatory argument a list (possibly empty) of environments, separated by commas and with no spaces. Within the `xcomment` environment, only text within each of the specified environments is typeset. The `\xcomment` command can also be used directly, with the same argument, and it would typically go in the preamble. Invoking the `\xcomment` command is equivalent to putting `\begin{xcomment}` at the invocation of the command or immediately after `\begin{document}`, whichever occurs later in the document, and `\end{xcomment}` just before the end of the document.

For example, if `\xcomment{table,figure}` is put in the preamble, only tables and figures are typeset (but see `\nofloat` below). If the list of environments is empty, as in `\begin{xcomment}{}`, the `xcomment` environment is essentially like the `comment` environment in verbatim.sty.

`xcomment` environments can be nested, but if the nested environments have the same name, the inner environments must be inside text that is typeset as specified by the `xcomment` environment the next level up.

Here is an example of such nesting. Suppose we want to include only figures, but we also want to be able to comment out some of the figures so that they are not included. This is achieved in the following example:

```
\newxcomment[]{mycomment}
\begin{xcomment}{figure,mycomment}
  This is stuff that is not included.
  \begin{figure}
    This figure is included.
  \end{figure}
  More stuff that is not included.
  \begin{mycomment}               Out and back into comment mode
    \begin{figure}                Ignored by mycomment envir.
      This figure is NOT included.
    \end{figure}                  Also ignored by mycomment envir.
  \end{mycomment}                 In and back out of comment mode.
  More stuff that is not included.
\end{xcomment}
```

`xcomment` will follow `\input` and `\include` commands. You must use the LaTeX syntax `\input{file}` (as opposed to `\input␣file␣`), and the inputted file must end with `\endinput`. You should be in `xcomment` mode when the `\endinput` command occurs if and only if you were in `\xcomment` mode when the `\input` or `\include` command was found.

`xcomment` is searching literally for `\begin{foo}` and `\end{foo}` when determining whether to switch in or out of comment mode. It will not find, e.g., `!begin[foo]`, even if `!`, `[` or `]` have category codes 0, 1 and 2, respectively. (If you don't understand this, you can safely ignore it.)

The comment character, %, is still a comment character in `xcomment` environments (even if used with \%). You can change the command character by redefining `\xcommentchar`. For example, if I want to use " as a comment character:

```
    \renewcommand{\xcommentchar}{\"}
```

If you define `\xcommentchar` to be empty, then no comment character is used.

In the `xcomment` environment, text is processed a line at a time and discarded until `\end{xcomment}` or `\begin{environment}` is encountered,

where *environment* is to be included. The remaining text on the line is not thrown away, as it is in verbatim.sty. Instead, it is rescanned, and the only restrictions are that it have balanced braces and that it not contain commands that again shift into *and* out of "comment" mode.

This is an important feature, because the included environments may have arguments that are best placed on the same line as the `\begin{environment}` command. However, the rescanning creates a temporary file. By default, the file is `\jobname.tmp`. The command `\rescanfile{file}` causes `file` to be created instead. `\rescanfile{}` suppresses the creation of a temporary file; the leftover text is simply discarded. `\norescanfile` also suppresses the creation of a temporary file, but in this case the text is simply inserted without being rescanned (i.e., with category codes 0 (escape \), 1 (begin group {), 2 (end group }) and 6 (parameter #) switched to 12 (other).

The command `\envirsep` is executed between the environments that are typeset. Its default definition is `\par`.

The command

```
1   \newxcomment[|\meta{environment list}|]{|\meta{name}|}
```

defines `\name` and the `name` environment to work like `\xcomment` and the `xcomment` environment. A list of environments to be included can be given as an optional first argument to `\newxcomment`, in which case the new command and environment do not take an argument. For example, if you put `\newxcomment[]{mycomment}` in the preamble, the `mycomment` environment works like the `comment` environment in verbatim.sty, except for the rescanning.

The command `\nofloat{environment list}` is provided to disable the floating of environments in the list (also separated by commas and without spaces), since if there are only floats and no text, the floats will accumulate to the end of the document and TEX may run out of memory.

Caveats:

- If `\xcomment` is invoked in the preamble, `\document` should not be subsequently redefined.
- Be careful what argument you give to `\rescanfile`, since any existing file with that name will be destroyed, and the extension `tex` is added if no extension is given.

Changes to v1.2:

- `\include` now allowed.
- Comment characters not obeyed, as determined by `\xcommentchar`.

Changes to v1.1:

- Fixed bug in `\@xcomment` that caused problems when invoking `\xcomment` in the preamble.

Changes to v1.0:

- Main loop rewritten.
- `\rescanfile{trash}` changed to `\rescanfile{}`.
- `\rescanfile{bounce}` no longer supported. Use `\norescanfile`.
- `\input` now allowed.

2 The implementation

The code is an adaption of code in verbatim.sty, and in fact the latter's structure has been preserved when possible. On the one hand, the code is simpler than in verbatim.sty because we throw away text from the input file rather than typeset it verbatim. On the other hand, it is more involved because we have to check for more possible endings to the `xcomment` than in verbatim.sty and we have to preserve the input that follows the `\begin`ning of an included environment on the same line.

As in verbatim.sty and the `\comment\endcomment` commands in $\mathcal{AMS}$-TEX, the basic strategy is to change the category codes so that control sequences and other troublesome special characters are neutralized, and then to scan for the strings that mark the end of the comment. This would be a 3-liner in `awk`, but this is TEX, and the amount of code required to do the scanning increases rapidly as we allow for a greater variety of ending strings.

We begin by ensuring that file is not read in twice, and then we identify the file on the VDU and the transcript file.

```
\@ifundefined{xcomment@@@}{}{\endinput}
\typeout{Style-Option: `xcomment' v\fileversion \space <\filedate> (tvz) }
```

2.1 User commands

The main toplevel macro is `\@xcomment`, which is defined in the next section. It works like `\xcomment` as described above, except that it has as a first argument an *environment* so that it stops when it encounters `\end{`*environment*`}`. `\newxcomment` then has a simple definition, and `\xcomment` is defined using the `\newxcomment` command.

```
\def\newxcomment{\@ifnextchar [{\@newxcommentwitharg}%
  {\@newxcomment}}
\def\@newxcomment#1{%
  \expandafter\def\csname #1\endcsname##1{\@xcomment{#1}{##1}}}%
\def\@newxcommentwitharg[#1]#2{%
  \expandafter\def\csname #2\endcsname{\@xcomment{#2}{#1}}}
\newxcomment{xcomment}
```

`\envirsep` is inserted between the environments that are typeset, and is set to `\par` by default. `\rescanfile` is the name of the temporary file used for rescanning, and is set to `\jobname.tmp` by default.

```
\def\envirsep{\par}
\def\rescanfile#1{\def\@rescanfile{#1}}
\rescanfile{\jobname.tmp}
\def\norescanfile{\let\@rescanfile\relax}
```

`\nofloat` disables floating, and `\vfill` is inserted forcefully on each side of each float. The macro is just a hack, and is mainly for printing only floats. It may not work well even for that. `\envirsep` is still inserted between floats, and if it is set to `\vspace{}`, the added space is inserted only between floats on the same page, given the user some control over spacing.

```
\def\@nofloat#1{\hrule height\z@\nobreak\vfill\vbox\bgroup\def\@captype{#1}}
\def\end@nofloat{\egroup\nobreak\vfill\nobreak\hrule height\z@\medbreak}
\def\nofloat#1{\@for\@tempa:=#1\do{\@namedef{#1}{\@nofloat{#1}}%
  \@namedef{end#1}{\end@nofloat}}}
```

2.2 Preliminaries

`\xc@makeother` takes as argument a character and changes its category code to 12 (other) if its category code is originally 0, 1, 2 or 6. We do not have to change as many catcodes as in Verbatim.sty, because the input is either thrown away or rescanned. The fewer codes we change the better.

```
\def\xc@makeother#1{%
  \ifnum\the\catcode`#1=0\catcode`#112%
  \else \ifnum\the\catcode`#1=1\catcode`#112%
    \else \ifnum\the\catcode`#1=2\catcode`#112%
      \else \ifnum\the\catcode`#1=6\catcode`#112%
  \fi\fi\fi\fi\relax}
```

This macro changes the category codes of a token list that is already in TEX's stomach. The code is a modification of a `\retokenize` macro by Raymond Chen.

```
\newwrite\tokout
\newread\tokin
```

The argument of `\rescan` is a token list register, say, `\mytoks`. `\mytoks` presumably contains a string from the input file, sent to `\TeX`'s stomach as a stream of tokens. `\rescan` gives `\mytoks` the token list that would have arisen if `\TeX` had had the current catcodes in effect when it read the input string, with the following exceptions:

- The token list must have balanced braces under the new catcodes.
- Parameter tokens in the original list are written twice, and so mess things up.
- Some characters with catcode 10 (space) under the old catcodes are lost, and all that remain are treated as character 32 (␣) under the new catcodes, whatever this character's new catcode is.
- Escape characters under the old catcodes are treated as escape characters even if their catcodes have changed. (This can be fixed by setting `\escapchar` during the write to the escape character under the old catcodes, assuming there was only one and it is known)

- Leading spaces under the new catcodes are not ignored, which is wrong if and only if the string started at the beginning of a line and the space characters were not spaces under the old catcodes.

Only the balanced braces exception is a problem in this application, but the other exceptions are pointed out since such a rescanning macro has other applications.

```
\def\rescan#1{%
```

First we check if `\rescanfile`, which contains the name of the temporary file to be used, has a special meaning. If `\relax`, do nothing. If empty, empty the token register.

```
  \ifx\@rescanfile\relax\else
    \ifx\@rescanfile\@empty #1{}\else
```

Put the list of tokens in braces and write them to the temporary file.

```
      \immediate\openout\tokout=\@rescanfile
      \immediate\write\tokout{{\the#1}\relax}%
      \immediate\closeout\tokout
```

Read the contents of the file `\@tempd`.

```
      \openin\tokin=\@rescanfile
      \read\tokin to\@tempd
      \closein\tokin
```

Suppose `#1` is `\mytok`, and `token list` is the list of tokens with the current catcodes. Then the next line expands to `\mytok{token list}\relax`.

```
      \expandafter#1\@tempd%
  \fi\fi}
```

2.3 Toplevel macros

`\@xcomment` checks whether it was invoked before `\begin{document}` and outside of an environment. If so, it modifies `\begin{document}` to invoke `\@xcomment` with the same arguments. Otherwise, it processes its arguments and goes into "comment" mode by invoking `\xc@begin`.

```
\def\@xcomment#1#2{%
  \ifx\@preamblecmds\@notprerr
    \def\xc@csname{#1}%
    \edef\xc@envirlist{#2}%
    \ifx\xc@envirlist\@empty \@bsphack \else
      \begingroup
```

`\@envirsep` is what is actually placed before each environment. Initially, it is empty because nothing should precede the first environment that is typeset.

```
      \def\@envirsep{}%
```

`\do@end` is appended to the meaning of `\end`, and it is initially set to `\xc@begin`. This means that the `\end` of an included environment switches us back into "comment" mode, as desired. The `\end` of nested environments should have their usual meaning, however. Therefore, `\do@begin` is prepended to the definition of `\begin`, and it sets `\do@end` to `\relax`. Included environment are begun with `\normal@begin` because we only want to reset `\do@end` for nested environements. The group begun above keeps the `\end` of the `xcomment` environment from be being affected by these changes.

```
      \@ifundefined{normal@begin}{\let\normal@begin\begin}{}%
      \@ifundefined{normal@end}{\let\normal@end\end}{}%
      \def\begin##1{\do@begin{##1}\normal@begin{##1}}%
      \def\end##1{\normal@end{##1}\do@end}%
      \def\do@begin##1{\@ifundefined{##1}{}{\def\do@end{}}}%
      \let\do@end\xc@begin
    \fi
    \let\next\xc@begin
  \else
    \expandafter\@temptokena\expandafter{\document\@xcomment{@@@}{#2}}%
    \edef\document{\the\@temptokena}%
    \let\next\relax
  \fi
  \next}
\def\end@xcomment{\ifx\xc@envirlist\@empty \@esphack \else \endgroup \fi}
```

`\xc@begin` starts up "comment" mode by changing the category codes and invoking `\xcomment@`. The `\begingroup` keeps these catcode changes local.

```
\def\xc@begin{%
  \begingroup
    \let\do\xc@makeother
    \dospecials
    \ifx\xcommentchar\@empty\else
      \expandafter\catcode\expandafter`\xcommentchar=14
    \fi
    \catcode`\^^M\active
    \xcomment@}
\def\xcommentchar{\%}
```