A package for recursive embedding of versions

Didier Rémy

October 6, 2006

Abstract

This package provides macros for organizing serveral versions of a same file. This defined a `\version` environment with a boolean expression argument that is used to decide whether the version is to be part of the current compilation. Version environments may be embedding, and the depth of the environment is visible in the evaluation of the condition.

As a default, simple case, versions can be used to simply comment out code—with as opposed to other packages allows embedding of comments.

The Boolean expression package can be used to combine several booleans on the fly driving the compilation.

1 Examples

The following source code is

```
\sf Here is some outer text, followed by
\begin{version}{\TRUE}
\em
some middle text, which itself contains some
\begin{version}{\TRUE}
\it inner text.
\end{version}
We continue with middle text,
\end{version}
and finish with outer text.
```

will render as:

> Here is some outer text, followed by *some middle text, which itself contains some inner text. We continue with middle text,* and finish with outer text.

While setting the inner flag to `\FALSE` will produce.

> Here is some outer text, followed by *some middle text, which itself contains some We continue with middle text,* and finish with outer text.

The argument of `\begin{version}` is passed to the command `\VersionEval`, which by default pass its argument to `\ORL`. Hence several comma-separated boolean expressions can actually be passed as argument and will be interpreted as `OR`. Hence, one may wirte

```
\newcommand{\LONG}{\TRUE}
\newcommand{\DRAFT}{\FALSE}
\begin{version}{\LONG,\DRAFT}
This is visible if and only if \verb"\ORL{\LONG,\DRAFT}"
evaluates to \verb"\TRUE".
\end{version}
```

and the code will appear only if either boolean `\LONG` or `\DRAFT` is true:

> This is visible if and only if `\ORL{\LONG,\DRAFT}` evaluates to `\TRUE`.

The command "" may be changed, as long as it evaluates its argument to a Church boolean, *i.e.* to a command that expect two arguments and return either one.

The is also a short form `\ifversion` that expect an argument of the same form and two arguments to be evaluated if the condition is true or false respectively. Note that `\ifversion` simply calls `\VersionEval`, which returns a Church Numeral.

The package also maintains the depth of the current version environment in the counter `\VersionDepth`, which can be questioned. The value during the evaluation of the argument is the outer value, *i.e.* 0 for the outer most one.