DE-MACRO

July 6, 2020

Version 1.4 - Luca Citi made it python2.7 and python3 compatible. Peter Gacs improved the parsing of `\input{<filename>}`, and made @ a letter in the style files.

Version 1.3 - this version is much more conservative about deleting comments and inserting or deleting blank space: tries to leave in all comments, adds space only when necessary, and tries not to delete space in the main text. The motivating comments came from Daniel Webb.

Version 1.2 - a syntactical bug corrected, thanks Brian de Alwis!

Purpose

This program can eliminate most private macros from a LaTeX file. Applications:

– your publisher has difficulty dealing with many private macros

– you cooperate with colleagues who do not understand your macros

– preprocessing before a system like latex2html, which is somewhat unpredictable with private macros.

Platform

This is a Python program, which I only have tried to run under Unix. But the Unix dependence is minimal (for example all the directory path references are platform-independent). It should be easy to adapt the program to Windows, and also to avoid command-line arguments.

In case your Python is not in `/usr/bin`, you should change the top line (the "shebang" line) of the program accordingly. This top line uses the `-O` option for python (stands for "optimize"). Without it, the program may run too slowly. If you do not care for speed, a number of other complications (the database, the checking for newer versions) could be eliminated.

Usage

Command line:

```
de-macro [--defs <defs-db>] <tex-file-1>[.tex] [<tex-file-2>[.tex] ...]
```

Simplest example: `de-macro testament`
(As you see, the <> is used only in the notation of this documentation, you should not type it.)

If `<tex-file-i>` contains a command `\usepackage{<defs-file>-private}` then the file `<defs-file>-private.sty` will be read, and its macros will be replaced in `<tex-file-i>` with their definitions. The result is in `<tex-file-i>-clean.tex`.

Only `newcommand`, `renewcommand`, `newenvironment` and `renewenvironment` are understood (it does not matter, whether you write `new` or `renew`). These can be nested but do not be too clever, since I do not guarantee the same expansion order as in TeX.

Files

```
<tex-file-1>.db
<tex-file>-clean.tex
<defs-file>-private.sty
```

For speed, a macro database file called `<defs-file>.db` is created. If such a file exists already then it is used. If `<defs-file>-private.sty` is older than `<tex-file-1>.db` then it will not be used.

It is possible to specify another database filename via `--defs <defs-db>`. Then `<defs-db>.db` will be used.

(Warning: with some Python versions and/or Unix platforms, the database file name conventions may be different from what is said here.)

For each `<tex-file-i>`, a file `<tex-file-i>-clean.tex` will be produced. If `<tex-file-i>-clean.tex` is newer than `<tex-file-i>.tex` then it stays.

Input command

If a tex file contains a command `\input{<tex-file-j>}` or `\input <tex-file-j>` then `<tex-file-j>.tex` is processed recursively, and `<tex-file-j>-clean.tex` will be inserted into the final output. For speed, if `<tex-file-j>-clean.tex` is newer than `<tex-file-j>.tex` then `<tex-file-j>.tex` will not be reprocessed.

The dependency checking is not sophisticated, so if you rewrite some macros then remove all `*-clean.tex` files!