FLaTTeN: A Program to Flatten LaTeX Source Files

Peter R. Wilson
Catholic University of America and NIST
Email: pwilson@cme.nist.gov

October 1995

Abstract

FLaTTeN is a table-driven program that will automatically incorporate the text of 'included' files into a LaTeX root file. This report describes Version 1.1 of the program.

Contents

1 Introduction

LaTeX is a popular document tagging and typesetting system [1]. A LaTeX source file can reference other LaTeX files whose contents are to be treated as an integral part of the source file. The standard LaTeX has two *inclusion* commands for this purpose, namely `\input{FileName}` and `\include{FileName}`. In its turn, a file that is `\input`ed can recursively `\input` other files. This LaTeX facility provides a means of splitting a large document into several files. Sometimes, though, it is desirable to combine the individual files into a single source file. An example might be when storing a multi-file LaTeX document into a text database where documents are maintained as one file per document.

The FlaTTeN program will read a LaTeX source file and automatically incorporate the source of inclusion files that are designated via the appropriate LaTeX commands.

2 The program

FlaTTeN requires two files to be specified: the name of the input source LaTeX file and the name of the file to which the concatenated source is to be output. For example, if `>` is the system prompt, then

```
> flatten in.tex inall.tex
```

will read the file `in.tex` and write out the source to file `inall.tex`, incorporating the source of any inclusion files identified within `in.tex`. FlaTTeN will write progress reports to the terminal, and also to an error file called `flatten.err`. This latter file can be examined using your favorite text editor after a run if required.

In the output file, FlaTTeN comments out any file inclusion commands, incorporates the source from the inclusion file, adds a comment at the end of this source, and then goes on outputting from the original source file. As far as FlaTTeN is concerned a file ends when either it reaches the end of the file or when it comes across the TeX `\endinput` command before the file physically ends.

If FlaTTeN comes across an inclusion file that it cannot find or read for any reason, it prints a warning message, both to the user's terminal and as a LaTeX comment in the output file, and then continues processing the current file.

FlaTTeN correctly ignores any inclusion commands that are embedded within LaTeX comments or are within verbatim text.

2.1 Command file

By default, FlaTTeN uses the LaTeX `\input` and `\include` commands as the inclusion commands[1]. It may happen that other file inclusion commands have been defined in a macro package that is used by the source document. In this case, by using the `-f` option

[1] FlaTTeN does not recognize the `\includeonly` command; do not attempt to make it do so.

(see §3), FLaTTeN can be told at runtime about the relevant inclusion commands to be used.

The `-f` option is used to specify a command table file. FLaTTeN reads the designated file which must contain *all* the inclusion commands (including `\input` and `\include` if relevant). It will then use these commands for deciding upon inclusions instead of the default inclusions. The format of the command file is very simple — no blank lines and one inclusion command per line. For example, here is a command file that contains three inclusion commands:

```
\input
          \include
    \infile
```

Note that a command does not have to be at the start of a line. In all cases, FLaTTeN assumes that an inclusion command, say `\inclusion`, has the syntax

```
\inclusion{filename}
```

and further assumes that if `filename` has no extension, then it is actually called `filename.tex`. This means that FLaTTeN will not be able to process an inclusion command like `\usepackage` because (a) this command assumes a file name extension of `.sty` and (b) the command can take an optional first parameter.

An example LaTeX source file could be like the following:

```
% root.tex starts here
....
\infile{file1}
...
\input{file2.mac}
...
\include{fileN}
...
% root.tex ends here
```

Running FLaTTeN on this source file and the above command table file will produce an output file along the lines of:

```
% root.tex starts here
...
%\infile{file1}
         % contents of file1(.tex) here
         % and any recursively called inclusions
%    END file1.tex
...
%\input{file2.mac}
         % contents of file2.mac here
         % and any recursively called inclusions
%    END file2.mac
...
```

```
%\include{fileN}
         % contents of fileN(.tex) here
         % and any recursively called inclusions
%    END fileN.tex
...
% root.tex ends here
```

On the other hand, running FLATTEN in its default mode will produce an output file similar to:

```
% root.tex starts here
...
\infile{file1}
...
%\input{file2.mac}
         % contents of file2.mac here
         % and any recursively called inclusions
%    END file2.mac
...
%\include{fileN}
         % contents of fileN(.tex) here
         % and any recursively called inclusions
%    END fileN.tex
...
% root.tex ends here
```

3 Running FLATTEN

The full syntax of the command to run FLATTEN is:

```
flatten [-d number] [-f CommandFile] [-D dir_cat_char] [-P path_separators]
        InFile OutFile
```

where the square brackets enclose runtime options.

`-d` An option to print debugging information to `flatten.err`. A value of 1 will print diagnostics related to file searching, while a value of 2 or more will produce full diagnostic information.

`-f` An option to read file inclusion commands from the file `CommandFile`.

`-D` The value of this option is the character that the operating system uses to catenate directory names to form a path. The default value is a slash (i.e., `/`). This could be changed to a backslash, for example, by including the option `-D \`. (See §3.1).

`-P` The character(s) that the operating system uses to separate path names in an environment variable. The default characters are space, colon and semi-colon (i.e. `␣:;`). The separator characters could be changed to colon and space by including the option `-P :` (space is always available as a separator character). (See §3.1).

`InFile` The name of the input file (including any extension).

`OutFile` The name of the output file.

3.1 Directory searching

The program employs a search algorithm to find files that are not in the current directory. It first looks in the current directory and if a file of the given name is found, then that is used. If the file is not found, then it searches for it among directories that are specified in a system environment variable. This variable specifies a list of pathnames, where the directories forming the path are combined using a catenation character. For example, `dir1/dir2/dir3` could be a pathname, where the slash (`/`) is the catenation character. If it is looking for file `afile.txt` it will catenate the file name to the path name (e.g. `dir1/dir2/dir3/afile.txt`) and look for that. The pathnames in the list are separated by another character (in fact it can be one from a list of characters). For example here is a list of two pathnames; `dir1/dir2;dir1/dir4`, where the semi-colon (`;`) is the pathname separator.

By default, the program uses a slash (`/`) as the directory catenation character and the pathname separators can be a space, or a colon or a semi-colon (i.e., any of `␣:;`). All these characters can be altered via the program command line options, and should be set to match the conventions of your operating system.

The environment variable used by the program is FLATINPUTS. On the operating system that I use, I set this in my login file like:

```
setenv FLATINPUTS .:/dir1/dir2:/dir3/dir4
```

Your system may have different conventions. Note that if the environment variable is not set, then the program only looks for files in the current directory.

A System installation

This section describes how to install the FLATTEN program.

FLATTEN is written using flex[2] and C [4]. The system consists of the following source files:

`flatten.l` The main source for FLATTEN.

`getopt.c` Functions for handling command line arguments [5, Chapter 6].

`getopt.h` A header file for `getopt.c`.

`srchenv.c` Functions for directory searching [6, page 747].

`srchenv.h` A header file for `srchenv.c`

`man` A manpage for FLATTEN.

The file `flatten.l` has to be processed by flex, or an equivalent tool, to generate C a file containing C code. This is then compiled and linked, together with the other system files, to form the executable program.

A.1 A `make` file

For those who understand such things, here is a `make` file [7] for FLATTEN.

```
# makefile for program flatten --- include all "include" files
#                                   into a LaTeX root file
#
###################### Change the following for your setup
# The compiler
CC = cc

# We use flex (or equivalent) to generate the lexer
LEX = flex
# and the options
LEXFLAGS = -v

# Libraries to be used
LIBS = -ll -lm

# The root directory for installation (e.g., /usr/local )
ROOTDIR = /proj/ltx/teTeX033

# Where to place the running code (e.g. /usr/local/bin )
BINDIR = ${ROOTDIR}/bin
```

[2] flex is a version of the lex program [2] but with increased functionality [3].

```
# Where to place the man page (e.g., /usr/local/man/man1 )
MANEXT = 1
MANDIR = ${ROOTDIR}/man/man${MANEXT}

# Just in case you want to change the name of the binary
# (and then you should also change the man page and documentation).
# So, do not change this.
PROG = flatten

# Where to place the user documentation (e.g., /usr/local/doc/flatten )
DOCDIR = ${ROOTDIR}/doc/${PROG}

# The file copy command (copy but do not delete original)
COPY = cp

# The file move command (move and delete original)
MOVE = mv

# The file delete command
DELETE = rm

# The make directory (hierarchy) command
MAKEDIR = mkdirhier

# The stream editor command
SED = sed

################### You should not have to change anything after this

# The object modules
OBJS = flatten.o getopt.o srchenv.o

# Link object code together into PROG
flatten : ${OBJS}
	${CC} -o ${PROG} ${OBJS} ${LIBS}

# Compile C source code into object code
flatten.o : flatten.c getopt.h srchenv.h
	${CC} -c flatten.c
getopt.o : getopt.c getopt.h
	${CC} -c getopt.c
srchenv.o : srchenv.c
	${CC} -c srchenv.c

# Generate C code via LEXing
flatten.c : flatten.l
	${LEX} ${LEXFLAGS} flatten.l
	${MOVE} lex.yy.c flatten.c
```

```
# Only call make install if BINDIR has been set
install : flatten
        ${MAKEDIR} ${BINDIR}
        ${MOVE} ${PROG} ${BINDIR}

# Edit file man to replace DOCUMENTDIR by the actual directory
# where the user manual is to be placed.
# Then copy the man page to the proper place.
manpage :
        ${SED} 's!DOCUMENTDIR!${DOCDIR}!' man > tman
        ${MAKEDIR} ${MANDIR}
        ${COPY} tman ${MANDIR}/${PROG}.${MANEXT}

# Copy the user manuals to the proper place
doc :
        ${MAKEDIR} ${DOCDIR}
        ${COPY} flatten.tex ${DOCDIR}/${PROG}.tex
        ${COPY} flatten.ps ${DOCDIR}/${PROG}.ps

# Do everything except clean up
all : flatten install manpage doc

# Call make clean to remove object files and edited man page
clean :
        ${DELETE} *.o
        ${DELETE} tman
```

When using this `makefile`, edit the first part of the file to configure it for your system and environment. If you are brave, you can then do `make all` followed by `make clean`. Otherwise just do `make` which will compile and link FLATTEN. After testing FLATTEN, do `make install` to move the binary to its proper location, `make manpage` to edit and copy the manpage to its desired location, and `make doc` to copy the user documentation to its final location. Finally, do `make clean` to get rid of the temporary files.

A.2 Limits

There are some built-in limits within FLATTEN. If necessary, these can be changed and the system re-built.

`MAX_COMMANDS` The maximum number of inclusion commands within a command file.

`MAX_DEPTH` The maximum allowable depth of file nesting. This is set to a value that is probably greater than the number of simultaneously open files that your operating (and LaTeX) system can support.

`MAX_ERRORS` The maximum number of errors before FLATTEN gives up.

MAX_NAME The maximum number of characters in a file name.

MAX_LINE The maximum length of a line in a LaTeX input file.

MAX_TABLE_LINE The maximum number of characters for an inclusion command in the command file.

A.3 Availability

Source code and documentation for FLaTTeN is available from the NIST SOLIS (STEP On-Line Information Service) system [8] in directory `pub/step/latex/programs/flatten`. SOLIS can be accessed by:

- Anonymous ftp to `ftp.cme.nist.gov`
- URL `gopher://elib.cme.nist.gov`
- URL `http://elib.cme.nist.gov:70/`

A.3.1 Copyright

A.3.2 Disclaimer

WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD ANY PORTION OF THE FLaTTeN SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT WILL NIST BE LIABLE FOR DAMAGES, INCLUDING ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH PROGRAMS NOT DISTRIBUTED BY NIST) THE PROGRAMS, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley Publishing Company, second edition, 1994.

[2] M. E. Lesk and E. Schmidt. 'LEX — A Lexical Analyser Generator'. In *UNIX Programmer's Manual 2.* AT&T Bell Laboratories, Murray Hill, NJ, 1975.

[3] John R. Levine, Tony Mason, and Doug Brown. *lex & yacc.* O'Reilly & Associates, Inc., second edition, 1992.

[4] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language.* Prentice Hall, second edition, 1988.

[5] Don Libes. *Obfuscated C and Other Mysteries.* John Wiley & Sons, Inc., 1993.

[6] A. I. Holub. *Compiler Design in C.* Prentice-Hall, Inc., 1990.

[7] Andrew Oram and Steve Talbott. *Managing Projects with make.* O'Reilly & Associates, Inc., second edition, 1991.

[8] Gaylen R. Rinaudot. *The IGES/PDES Organization STEP On-Line Information Service.* NISTIR 5511, NIST, Gaithersburg, MD 20899, October 1994.