NAME

make_latex – Makefile rules for LaTeX documents

SYNOPSIS

~/lib/make_latex

DESCRIPTION

The make(1) utility provides a general mechanism for regenerating a file which depends upon other files, when any of those other files is altered. The action is recursive. Although make(1) is normally used for compiling programs, there is no reason why make should not be used in other circumstances where several files must be brought together, and appropriately processed, in order to produce a desired result.

For example, to print a LaTeX document, a postscript file is required, which depends upon a .dvi file, and other .ps files of included figures. The .dvi file depends on a .tex file, the .ps files of included figures, and a .bib file. The .ps files of included figures may themseleves depend on .fig files or .gnu files generated by other utilities. Thus we have a tree of file dependencies, which is exactly what make was designed to cope with.

make_latex provides a set of rules which tell make(1) how to generate a file at one level in the file dependency tree, given the files at the next level higher up in the tree. Users need only specify the file dependencies in a makefile, and make(1) can do the rest. When anything is altered, make determines what needs to be done to bring other files up to date.

make_latex knows how to do the following:

– send a .ps file to be printed
– convert a .dvi file to a .ps file
– convert a .tex file to a .dvi file (including running BibTeX if necessary)
– convert a gnuplot(1) file to a .ps file
– convert an xfig(1) file to a .ps file
– convert an xfig(1) file to a latex picture file
– tidy up intermediate files

Other capabilities can easily be added, according to individual user's requirements.

INSTALLATION

The *make_latex* file should be installed in any convenient directory. Since make_latex is not large, we suggest that each user has their own copy, and places it in directory ~/lib

Some entries in the make_latex file will probably need customising to provide convenient defaults for each user. These appear near the top of the file. (Note that it is possible to override any of these settings on an individual basis. See below).

BIBFILES
The name(s) of the BibTeX files which hold your references.

PRINT_RES
The printer resolution (dots per inch) at which you normally print. 300 should be ok for most people.

PRINTER The name of the printer to which normal resolution print jobs should be sent by default.

PR_HI_RES
The name of the printer to which high resolution (>400 dpi) print jobs should be sent by default.

LPR The name of the print command. lpr should be ok for most people.

PR_OPT The default options to be sent with each print command. Generally set to "-r", to delete a postscript file after printing it.

OLDVERSIONS
The filename pattern(s) to match for files to be deleted by a "make clean" (see below).

MAKEFILE FORMAT

In each directory holding LaTeX documents, a file named *Makefile* is required. A sample file is as follows:

```
include $(HOME)/lib/make_latex
FILES = myfile paper
myfile.dvi : fig1.ps fig2.ps part2.tex
paper.dvi : complex.ps dna.ps
```

The first line loads the rules from the *make_latex* file. The second line (an example of a *macro assignment*) specifies the names (without extensions) of all the major LaTeX files in the directory. The final two lines specify file dependencies. In this example, myfile.tex has two included figures, fig1.ps and fig2.ps, and also an included LaTeX file, part2.tex.

If a file exists called fig1.fig (generated by xfig(1)) or fig1.gnu (a gnuplot(1) file), make knows that fig1.ps depends on fig1.fig or fig1.gnu – there is no need to specify this.

Entries on each line are separated by whitespace. If there are more entries than can fit on a single line, line breaks must be escaped by the "\" character.

Note that the order of lines in the *Makefile* is not critical, except as noted in the section: Overriding Defaults.

TARGETS FOR MAKE

Processing is initiated with the command make(1), followed by the name(s) of the "target" files to be generated. For example,

```
make myfile.ps
```

will do all that is necessary to generate myfile.ps; it will run LaTeX on myfile.tex (and if required, BibTeX, followed by re-running LaTeX), then run dvips on myfile.dvi to produce myfile.ps. Similarly, "make myfile.dvi" will just generate myfile.dvi. The command "make myfile" will also generate myfile.dvi, but will work unconditionally, even if make doesnt think that anything has changed. (This can be useful if, for example, a style file is altered, and make doesnt know about the dependency.)

The command

```
make myfile.pr
```

is an example of a special target. It does not produce a file called "myfile.pr" – instead it generates myfile.ps, and then sends it to be printed.

Other special targets are:

all to generate all .dvi files

allps to generate all .ps files

allpr to generate and print all .ps files

tidy to delete log files and intermediate files which can easily be recovered

clean to tidy, and delete intermediate files which can less easily be recovered

OVERRIDING DEFAULTS

Default macro assignments can easily be overridden on a per-Makefile or per-LaTeX file basis. For example, to specify "alternate.bib" (a file in the current directory) as the BibTeX source file for all LaTeX files in the *Makefile*, insert the line:

BIBFILES = alternate.bib

(This line must come after the "include" line.) If you wish to specify that "alternate.bib" should be consulted *in addition to* the default BibTeX file(s), use:

BIBFILES += alternate.bib

To override a default value only for specific LaTeX files in the *Makefile*, a *conditional macro assignment* can be used, with the format:

<target> [...] := <macro assignment>

For example, to specify a different BibTeX file for myfile.tex, use:

myfile.dvi := BIBFILES = newrefs.bib

which in effect means "when generating the file myfile.dvi, use the assignment: BIBFILES = newrefs.bib". Another example: to specify a different printer for plan.tex, use:

plan.pr := PR_LO_RES = laser1

Note that it is important use the right target name. An entry of the form:

plan.ps := PR_LO_RES = laser1

would have no effect, since the value of PR_LO_RES is irrelevant while plan.ps is being generated.

COMMAND LINE OPTIONS

In addition to being able to override defaults in the *Makefile*, macro values can be specified on the command line. For example, to print plan.tex on printer "pr1", use:

make PRINTER=pr1 plan.pr

Note that no spaces are allowed around the "=". Note also that command line macro assignments do not override conditional macro assignments in the *Makefile*. The assignments you would be most likely to want to change on the command line are:

PRINTER Printing device for low resolution (<400 dpi) output.

PRINT_RES
Resolution (dots per inch) to print at.

PR_HI_RES
Printing device for high resolution (>400 dpi) output.

PAGE_F Page number to start printing from.

PAGE_T Page number to print up to.

TIDYING UP

The targets **tidy** and **clean** can be used to delete unimportant files.

Tidy will delete all .dvi, .ps, .log, and .blg files generated from any file specified in the FILES assignment. It also deletes any files specified in an assignment to TIDYFILES. Note that other .ps files (such as those generated from .fig or .gnu files) are *not* deleted, unless specified in a TIDYFILES assignment.

Clean will delete all **tidy** files, and also delete .aux and .bbl files generated from any file specified in the FILES assignment. Editor backup files which match the pattern specified in OLDVERSIONS are also deleted, along with any files specified in an assignment to CLEANFILES.

SEE ALSO

make(1), latex(1), bibtex(1)

AUTHOR

David Beasley <David.Beasley@cm.cf.ac.uk>