Using latexmk (4.04) With TEXShop 2.25.

by
Herbert Schulz
herbs2@mac.com

What is latexmk?

Compiling a tex file that contains cross-references, bibliographic references and/or indexes is a multi-pass process; i.e., you've got to run (pdf/xe)latex multiple times with possible intermediate runs of bibtex and/or makeindex until all references are resolved. The latexmk perl program, rewritten and presently maintained by John Collins[1], automates this multi-pass process. By default it first runs (pdf/xe)latex on a source file, determines file dependencies by examining the log file produced by the run and then automatically runs bibtex and/or makeindex, if needed, and the correct number of additional runs of (pdf/xe)latex to generate the bibliography, index and cross-references. Recent versions of latexmk also work correctly with the nomencl package as well as the glossary and glossaries packages and other packages that produce multiple bibliographies or indexes.

What is here?

There is a set of five engine files to be moved from ~/Library/TeXShop/Engines/Inactive/Latexmk/ (where you are reading this document) two directories up, ~/Library/TeXShop/Engines/. A second set of eight files are already in ~/Library/TeXShop/bin/ and consists of the latexmk program, five initialization (rc) files used by the five engine files and two shell scripts used for pdftricks and pst-pdf figure processing.

Using latexmk with TEXShop.

There are five engine files; one for running latex (with a final run through dvips and ps2pdf14) [latexmk.engine], one for using pdflatex [pdflatexmk.engine], one for xelatex [xelatexmk.engine] and two for using the pdftricks or pst-pdf packages with pdflatex [pdftricksmk.engine or pst-pdfmk.engine respectively]. The exact form of the commands and options used are in the corresponding rc file (e.g., latexmkrc for the latexmk.engine) in ~/Library/TeXShop/bin/.

You can use these engine files by using the drop down menu on the source tool bar or placing the line

```
%!TEX TS-program = pdflatexmk
```

(for using pdflatex—similar lines for latex and xelatex) at the top of your document; then simply using Typeset (⌘-T) will automatically run the proper engine. Using latexmk with the epstopdf, pdftricks and pst-pdf packages is discussed later.

I have only tested these engines with relatively trivial distributed documents (which include other files using \include commands) but it appears that latexmk deals

[1]The latexmk web site is <http://www.phys.psu.edu/~collins/software/latexmk-jcc/>. You can get the latest version of latexmk, presently 4.04, at <http://www.phys.psu.edu/~collins/software/latexmk-jcc/versions.html>.

with them properly. Note that when compiling a file with name rootname.tex a file with name rootname.fdb_latexmk[2] is created. This file contains the dependency information for the distributed document so making changes in an included file will force the recompile of the root document by latexmk.

Noteworthy Changes with latexmk 4.04.

Versions of latexmk prior to 3.21c weren't able to deal with the glossary, glossaries or nomencl packages because they re-write their output file(s) with each run of (pdf/xe)latex or use custom file extensions. This changed with latexmk 3.21c. The rc files included with this version of latexmk for TeXShop are set to recognize the standard file extensions produced by the these packages and process them correctly and "auto-magically." If you are creating custom glossaries or indexes you will have to properly edit the rc files (e.g., pdflatexmkrc) found in the ~/Library/TeXShop/bin/ directory to add the dependencies; it should be fairly clear from the contents of the rc files what has to be added to those files.

Another major addition in latexmk since 3.21c is support for packages that create multiple bibliographies and/or indexes; e.g., when the bibunits, chapterbib, multibib, multind or similar packages are used. The extra processing needed for those packages happens automatically. Unfortunately, the index package uses the same naming scheme[3] as the glossary and glossaries packages (see the sub-section below) so you need to define extra dependencies and processing rules in the provided rc files. There was a bug in latexmk 3.21j that didn't allow it to work properly with the index package when creating an ordinary index (an .idx file); this was corrected with version 4.01 of latexmk.

Using the epstopdf package with latexmk.

Including eps graphics files directly in pdflatex documents requires the use of the epstopdf package. If you have an included eps file *and a converted pdf version of the file doesn't exist* the epstopdf package converts the eps file into a corresponding pdf file.

Using latexmk with epstopdf version 1.4 and earlier.

With epstopdf versions 1.4 and earlier once the pdf image file exists the conversion no longer takes place *even if the eps file is updated*. The pdflatexmkrc file now contains a dependency that uses a new rule, built into latexmk 4.01 and later, that will delete a previously generated pdf file and then run pdflatex so that epstopdf will regenerate the pdf image file. **Note: The file name in your \includegraphics commands should *not* have an eps extension to prevent extra, unnecessary runs of pdflatex.**

Using latexmk with epstopdf version 1.5 and later.

You can use the same (default with this distribution) processing with epstopdf 1.5 and later, however the epstopdf package, version 1.5 and later can check for an updated eps file and then recreate the pdf file if the [update,prepend] package options are used. The dependency checking by latexmk is still important to let latexmk "know" that an included eps file has changed but the deletion of the pdf image file is unnecessary. The pdflatexmkrc, etc., support files for latexmk 4.01 and later now contain a dependency and rule that will detect an updated eps file but let epstopdf do the conversion to pdf. By default this dependency is turned *off* in pdflatexmkrc; to

[2]The dependency file had extension dep in previous versions of latexmk but didn't do a complete job of keeping track of those dependencies.

[3]Custom extensions rather than standard extensions with custom root file names.

turn it on you should edit that file by commenting out the original dependency (place a # before the line

```
add_cus_dep('eps', 'pdf', 0, 'cus_dep_delete_dest');
```

in that file) and uncommenting the new dependency (remove the # from the start of the line

```
#add_cus_dep('eps', 'pdf', 0, 'cus_dep_require_primary_run');
```

in that same file). Remember that `latexmk` will work properly without making this change.

In version 1.5 and later of the `epstopdf` package you can also specify non-default processing for the `eps` to `pdf` conversion[4]. Since `latexmk` now lets the `epstopdf` package do all of the necessary processing of the `eps` file any customized processing defined in the `tex` source file will be used.

Note: I have noticed that there are times when including the `eps` extension in `\includegraphics` still gives rise to additional runs of `pdflatex` so I still recommend you leave off the extension in `\includegraphics` commands.

Using the `pdftricks` package with `latexmk`.

The `pdftricks` package allows the inclusion of `pstricks` graphics in documents compiled with `pdflatex`. The package generates a file for each postscript figure included in the document. Each of those figure files is then processed to produce a `pdf` file containing a figure with a tight enclosing bounding box. The `pdftricksmk` engine included with this version of `latexmk` processes the original file, the figure files, etc., all only if they have changed. To use the engine place the line

```
%%!TEX TS-program = pdftricksmk
```

at the start of the file and Typeset the file. The processing steps for each of the figure files is `latex`→`dvips`→`ps2pdf14`→`pdfcrop` to ensure the proper bonding box is created for each figure. **NOTE: you must use the `[noshell]` option to the `pdftricks` package or `latexmk` will get into a run-on condition. All figure processing will be taken care of by `latexmk`.**

Using the `pst-pdf` package with `latexmk`.

The `pst-pdf` package also allows the inclusion of `pstricks` graphics in documents compiled with `pdflatex`. When the source file is compiled with `latex` a `dvi` file containing all of the figures is created. Further processing through the sequence `dvips`→`ps2pdf14`→`pdfcrop` produces a single file that contains all of the figures with proper bounding boxes. A run of `pdflatex` on the source file then includes all of the figures previously generated. The `pst-pdfmk` engine takes care of all of the intermediate processing of the figures as well as the final run(s) of `pdflatex`, etc. To use the engine place the line

```
%%!TEX TS-program = pst-pdfmk
```

at the start of the file and Typeset the file.

The `glossary`, `glossaries` and such packages.

Packages that produce multiple and custom indexes, glossaries, etc., use one of two naming schemes for the multiple files they create:

1. The first uses standard extensions but special files names for the generated files. `Latexmk` can keep track of real changes in and “knows” how to process these files. The `multibib` and `multind` packages are examples that use this method.

[4]The default processing uses the `epstopdf` command which, in turn, uses `ghostscript`.

2. The second uses the source file name for the file but uses custom extensions to create the files. `Latexmk` needs "help" to know how to process these files in the form of dependencies and rules. Dependencies tell `latexmk` what the input and output extensions are and which rule to use to go from input to output. The `index`, `glossary` and `glossaries` packages are examples that use this second method.

In addition, while the `glossaries` package supersedes the `glossary` package the order of the file extensions created by acronym and custom lists, processed by `makeindex` and then read in by subsequent runs of `(xe/pdf)latex` are reversed in the two packages. This latest version of `latexmk` configured for TEXShop works correctly for both packages. If you need to create your own custom lists see the examples in the `rc` files for creating dependancies and rules for `latexmk`.

What these engines won't do, etc.

There are many features of `latexmk` that aren't used in these simple `engine` files. See the documentation for `latexmk` in the supplied full distribution.

In addition, the placement of the `latexmk` program in `~/Library/TeXShop/bin/` is non-standard; that directory is not on the standard path. It is possible to put the program in `/usr/local/bin/` and it will then be usable from the command line. If you install the program there you should modify the `engine` files to reflect the change in location.

The contents of the `rc` files corresponds to the the settings for commands for TEXShop on my system. They are simply text files. Please read the `latexmk` documentation before changing the contents.

Finally, changes in `eps` files *included in figures* created by the `pdftricks` or `pst-pdf` packages are *not* detected by this packaging `latexmk` at this time. I hope correct that problem at a later date.

Update for TEXShop 2.18 (and later) with MacTEX 2008 (ditto).

The `rc` files for this version of `latexmk` for use with TEXShop have been updated to allow use of `synctex`, a `tex`↔`pdf` synchronization technology, with `MacTeX-2008` and TEXShop 2.18. If you are using `MacTeX-2007` or earlier TEX distributions and the inconsequential error message about an unknown option bother you, remove the `-synctex=1` options provided in the supplied `rc` files.

Try it... I hope you like it.

Good Luck,
Herb Schulz
(herbs2@mac.com)