LaTeX-Mk

For version 2.1, 28 December 2010

Dan McMahill

Copyright © 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010 Dan McMahill

This code is derived from software written by Dan McMahill
This manual is derived from documentation written by Dan McMahill

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software
   must display the following acknowledgement:
      This product includes software developed by Dan McMahill
4. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

1 Introduction

LaTeX-Mk is a tool for managing small to large sized LaTeX projects. The typical LaTeX-Mk input file is simply a series of variable definitions in a '`Makefile`' for the project. After creating a simple '`Makefile`' the user can easily perform all required steps to do such tasks as: preview the document, print the document, or produce a PDF file. LaTeX-Mk will keep track of files that have changed and how to run the various programs that are needed to produce the output.

As a quick example, consider a project which has a single LaTeX file, '`mydoc.tex`', as its input. To produce a '`.pdf`' file you might use the following sequence of commands:

```
latex mydoc.tex
latex mydoc.tex
latex mydoc.tex
dvips -Ppdf -j0 -o mydoc.ps mydoc.dvi
ps2pdf mydoc.ps mydoc.pdf
```

The triple invocation of '`latex`' is to ensure that all references have been properly resolved and any page layout changes due to inserting the references have been accounted for. The sequence of commands isn't horrible, but it still is several commands and one of them, '`dvips`', has some flags to remember. To use LaTeX-Mk for this project, you would create a '`Makefile`' that contains the following.

```
NAME=mydoc

include /usr/local/share/latex-mk/latex.gmk
```

Note that the `include /usr/local/share/latex-mk/latex.gmk` is the syntax for GNU '`make`'. If you are using BSD '`make`' you would replace the include line with `.include "/usr/local/share/latex-mk/latex.mk"`. In both examples, you would replace `/usr/local` with the installation prefix on your system. For the remainder of this document we will use the BSD style of include in the examples. Now to create a '`.pdf`' file you simply run '`make pdf`'. For larger projects which may need to run programs to export drawings to Postscript files for inclusion or run BibTeX to generate bibliographies, the generation of '`.pdf`' (or other) files becomes increasingly complicated to run manually. With LaTeX-Mk, such operations are still very simple.

As a more complicated example, consider a project whose LaTeX input is broken apart in to many '`.tex`' files which are all included by '`mydoc.tex`'. Also suppose the project includes a bibliography and a large number of figures created with the Tgif program. An example '`Makefile`' for this project might look like:

```
NAME=        mydoc
TEXSRCS=     ch1.tex ch2.tex ch3.tex refs.tex
BIBTEXSRCS=  mybib.bib
TGIFDIRS=    tgif_figs

.include "/usr/local/share/latex-mk/latex.mk"
```

In this example is it assumed that all of the Tgif figures reside in a subdirectory called '`tgif_figs`'. When the user issues a '`make`' command, all of the steps required to reformat the document are taken. Because of the dependency structure imposed by '`make`', only the

steps which need to be taken are done. This avoids re-exporting a large number of figures which may have not changed, but ensures that files which need processing are processed.

Hopefully this introduction has provided an adequate example for how LaTeX-Mk can simplify the management of LaTeX based documents. The LaTeX-Mk system is simple enough for small projects and powerful enough for large projects. The remainder of this manual will provide complete documentation on the use of LaTeX-Mk as well as configuration and installation instructions.

2 Targets

LaTeX-Mk provides a fixed set of targets, the argument to the '`make`' command, for all projects. The default target is '`view`' whose ultimate goal is to provide an on-screen preview of the formatted document. For additional information on the '`make`' program, please refer to the documentation for your copy of '`make`'.

2.1 Base Targets

The targets provided by LaTeX-Mk are:

clean Target

Cleans the current working directory by removing all LaTeX output and other output files created during processing of the project. In addition, emacs '`~`' files are removed.

dist Target

Creates a '`.tar.gz`' file containing an archive of the project. It contains the source files and additionally some generated files which are generated with tools which someone else may not have installed. For example, any tgif drawings are included both in tgif '`.obj`' form as well as encapsulated Postscript (or PDF if you are using pdflatex). For a multiple document project, a master '`.tar.gz`' file is created containing the entire project as well as smaller '`.tar.gz`' files for each document.

dvi Target

Performs all processing required to produce the '`.dvi`' file for the project.

html Target

Performs all processing required to produce HTML output for the project.

pdf Target

Performs all processing required to produce a PDF (Portable Document Format) file, '`.pdf`', for the project.

print Target

Sends the processed document to the printer.

ps Target

Performs all processing required to produce a Postscript file, '`.ps`', for the project.

rtf Target

Performs all processing required to produce a RTF (Rich Text Format) file, '`.rtf`', for the project. Please note that the ability of LaTex to RTF converters to work correctly is somewhat limited. Your mileage may vary.

show-var Target

This target is used to help debug users Makefiles as well as the LaTeX-Mk system. This target displays the value of the variable whose name is given by the variable VARNAME. For example:

```
make show-var VARNAME=TEXSRCS
```

will display the value of the `TEXSRCS` variable.

view Target

Previews the '`.dvi`' file.

viewpdf Target

Previews the PDF ('`.pdf`') file.

viewps Target

Previews the Postscript ('`.ps`') file.

2.2 Draft Targets

LaTeX-Mk supports adding a DRAFT watermark and timestamp for the Postscript, PDF, and printed output. To produce the draft versions, simply append `-draft` to the target. The currently supported draft targets are:

pdf-draft Target

Draft version of the `pdf` target.

ps-draft Target

Draft version of the `ps` target.

print-draft Target

Draft version of the `print` target.

viewpdf-draft Target

Draft version of the `viewpdf` target.

viewps-draft Target

Draft version of the `viewps` target.

2.3 Per Document Targets

LaTeX-Mk supports multiple top level documents in a single directory controlled by a single makefile. For each top level document specified in the `NAME` variable (more on variables later), there will be a set of targets defined which are specific to the document. The per document targets are:

print_<name> Target
Prints the Postscript file '`<name>.ps`'.

view_<name> Target
Previews the DVI file '`<name>.dvi`'.

viewpdf_<name> Target
Previews the PDF file '`<name>.pdf`'.

viewps_<name> Target
Previews the Postscript file '`<name>.ps`'.

In addition, draft versions of these targets exist:

print_<name>-draft Target
Draft version of the `print_<name>` target.

ps_<name>-draft Target
Draft version of the `ps_<name>` target.

view_<name>-draft Target
Draft version of the `view_<name>` target.

viewpdf_<name>-draft Target
Draft version of the `viewpdf_<name>` target.

viewps_<name>-draft Target
Draft version of the `viewps_<name>` target.

3 Variables

The variables used by LaTeX-Mk can be categorized roughly into two groups. The first set of variables are typically set during the installation of LaTeX-Mk and these defaults used for all projects. These variables can be overridden on a per-user or per-project basis for maximum flexibility. The second set of variables are set by the user on a per-project basis.

3.1 Site Configuration Variables

This section documents the variables which are typically set on a site-wide or user-wide basis.

3.1.1 Site and User Configuration File

MAKECONF Variable

This variable is set to the location of the site-wide configuration file. If this file exists, it is sourced at the beginning of the LaTeX-Mk code. Default is '`${sysconfdir}/latek-mk.conf`' for BSD make and '`${sysconfdir}/latex-gmk.conf`' for GNU make. This is where system administrators can set system wide configuration variables. Any variables defined here should be defined using `VARIABLE?= "new value"` instead of `VARIABLE= "new value"` so that individual users can easily override the setting. The default setting may be changed during configuration of the package using the `--with-mkconf` and `--with-gmkconf` flags to `configure`. The `sysconfdir` directory can be specified to `configure` with the `--sysconfdir=` option.

USER_MAKECONF Variable

This variable is set to the location of a users personal configuration file. If this file exists, it is sourced at the beginning of the LaTeX-Mk code. Default is '`$HOME/.latex-mk.conf`' for BSD make and '`$HOME/.latex-gmk.conf`' for GNU make. This file is sourced before the file specified by `MAKECONF`. The default setting may be changed during configuration of the package using the `--with-usermkconf` and `--with-usergmkconf` flags to `configure`.

3.1.2 Generic Project Variables

This section documents the variables which are typically set on a site-wide or user-wide basis. In a typical installation these variables do not need to be explicitly set as they will take on reasonable defaults.

BIBTEX Variable

The bibtex executible. Defaults to '`bibtex`'.

BIBTEX_ENV Variable

Deprecated. Actually this variable did not work correctly anyway. Use LATEX_ENV to set variables for both LaTeX and bibTeX runs.

BIBTEX_FLAGS Variable

A list of flags to be passed to the BIBTEX executible. Defaults to ‘’.

CONVERT Variable

The image file format conversion executible which is part of the ImageMagick (`http://imagemagick.org/`) suite. Defaults to ‘`convert`’.

DVIPDFM Variable

The executible which produces PDF files from ‘`.dvi`’ files. Defaults to ‘`dvipdfm`’. Note that the default behavior is to use DVIPS to produce Postscript and then PS2PDF to produce a PDF file. To use DVIPDFM to directly produce PDF from DVI, set the USE_DVIPDFM variable.

DVIPDFM_ENV Variable

A list of variables to be set in the environment when DVIPDFM is executed. Defaults to ‘’.

DVIPDFM_FLAGS Variable

A list of flags to be passed to the DVIPDFM executible. Defaults to ‘’.

To set flags on a per-document basis, you can use `<docname>_DVIPDFM_FLAGS` where `<docname>` is the name of the document.

DVIPDFM_LANDSCAPE_FLAGS Variable

A list of flags to be added to DVIPDFM_FLAGS when the LANDSCAPE variable is set. Defaults to ‘`-l`’.

DVIPS Variable

The executible which produces Postscript files from ‘`.dvi`’ files. Defaults to ‘`dvips`’.

DVIPS_ENV Variable

A list of variables to be set in the environment when DVIPS is executed. Defaults to ‘’.

DVIPS_FLAGS Variable

A list of flags to be passed to the DVIPS executible. Defaults to ‘`-j0`’. Note: versions of latex-mk prior to 1.2 used ‘`-Ppdf -j0`’ as the default. If you wish to maintain this behavior on latex-mk-1.2 and newer, you will need to set this variable in your site or user configuration file.

To set flags on a per-document basis, you can use `<docname>_DVIPS_FLAGS` where `<docname>` is the name of the document.

DVIPS_LANDSCAPE_FLAGS Variable

A list of flags to be added to DVIPS_FLAGS when the LANDSCAPE variable is set. Defaults to ‘`-t landscape`’.

GV Variable

The executible which previews Postscript files. Defaults to '`gv`'.

GV_FLAGS Variable

A list of flags to be passed to the GV executible. Defaults to ''.

GV_LANDSCAPE_FLAGS Variable

A list of flags to be added to GV_FLAGS when the LANDSCAPE variable is set. Defaults to '`-landscape`'.

GZCAT Variable

The gzcat file decompression utility. Defaults to '`gzcat`'.

GZIP Variable

The gzip file compression utility. Defaults to '`gzip`'.

HEVEA Variable

The Hevea executible. Defaults to '`hevea`'.

HEVEA_ENV Variable

A list of variables to be set in the environment when HEVEA or IMAGEN is run. For example:

```
HEVEA_ENV+= TEXINPUTS=.:/home/usr/tex:
```

Defaults to ''.

HEVEA_FLAGS Variable

A list of flags to be passed to the HEVEA executible. Defaults to '`-fix`'.

IMAGEN Variable

The imagen executible (part of HeVeA). Defaults to '`imagen`'.

JPG2EPS Variable

The command to convert a JPEG file to an Encapsulated Postscript (EPS) file. Defaults to '`${CONVERT}`'.

LATEX Variable

The LaTeX executible. Defaults to '`latex`'.

LATEX_ENV Variable

A list of variables to be set in the environment when LATEX is run. For example:

```
LATEX_ENV+= TEXINPUTS=.:/home/usr/tex:
```

Defaults to ''.

LATEX_FLAGS Variable

A list of flags to be passed to the LATEX executible. Defaults to ''.

LATEX2HTML Variable

The LaTex2HTML executible. Defaults to '`latex2html`'.

LATEX2HTML_ENV Variable

A list of variables to be set in the environment when LATEX2HTML is run. For example:

```
LATEX2HTML_ENV+= TEXINPUTS=.:/home/usr/tex:
```

Defaults to ''.

LATEX2HTML_FLAGS Variable

A list of flags to be passed to the LATEX2HTML executible. Defaults to '`-image_type png -local_icons -show_section_numbers`'.

LATEX2RTF Variable

The LaTex2rtf executible. Defaults to '`latex2rtf`'.

LATEX2RTF_ENV Variable

A list of variables to be set in the environment when LATEX2RTF is run.

LATEX2RTF_FLAGS Variable

A list of flags to be passed to the LATEX2RTF executible. Defaults to ''.

LPR Variable

The executible which spools Postscript files to a printer. Defaults to '`lpr`'.

LPR_FLAGS Variable

A list of flags to be passed to the LPR executible. For example:

```
LPR_FLAGS= -Pbeernuts
```

Defaults to ''.

MAKEGLS Variable

The executible used to make glossaries. Defaults to '`makeindex`'.

MAKEGLS_FLAGS Variable

A list of flags to be passed to the MAKEGLS executible. Defaults to ''.

MAKEIDX Variable

The makeindex executible. Defaults to '`makeindex`'.

MAKEIDX_FLAGS Variable

A list of flags to be passed to the MAKEIDX executible. Defaults to ''.

MPOST Variable

The METApost executible. Defaults to '`mpost`'.

MPOST_FLAGS Variable

A list of flags to be passed to the MPOST executible. Defaults to ''.

PDFLATEX Variable

The PDFLaTeX executible. Defaults to '`pdflatex`'.

PDFLATEX_ENV Variable

A list of variables to be set in the environment when PDFLATEX is run. For example:

```
PDFLATEX_ENV+= TEXINPUTS=.:/home/usr/tex:
```

Defaults to ''.

PDFLATEX_FLAGS Variable

A list of flags to be passed to the PDFLATEX executible. Defaults to ''.

PNG2EPS Variable

The command to convert a Portable Network Graphic (PNG) file to an Encapsulated Postscript (EPS) file. Defaults to '`${CONVERT}`'.

POST_BIBTEX_HOOK Variable

If this variable is set to the name of an executible, then LaTeX-Mk will run this program/script immediately after a BibTeX run. This hook provides the ability to do post-processing of the BibTeX output prior to the final LaTeX run(s). This feature is likely to be of interest to advanced users only. The program/script is run in the same environment as specified by LATEX_ENV and is given the project name as an argument. For example if your '`Makefile`' contains

```
NAME= mydoc
POST_BIBTEX_HOOK=  ./my_bib_fixup
```

then the after BibTeX is run, '`./my_bib_fixup mydoc`' will be run. POST_BIBTEX_HOOK defaults to ''.

PS2PDF Variable

The executible which produces PDF ('`.pdf`') files from Postscript ('`.ps`') files. Defaults to '`ps2pdf`'.

PS2PDF_FLAGS Variable

A list of flags to be passed to the PS2PDF executible. Defaults to ''.

TAR Variable

The tar tape archiver utility. Defaults to '`tar`'.

TEX2PAGE Variable

The tex2page executible. Defaults to '`tex2page`'.

TEX2PAGE_ENV Variable

A list of variables to be set in the environment when TEX2PAGE is run. For example:

```
TEX2PAGE_ENV+= TEXINPUTS=.:/home/usr/tex:
```

Defaults to ''.

TEX2PAGE_FLAGS Variable

A list of flags to be passed to the TEX2PAGE executible. Defaults to ‘’.

USE_DVIPDFM Variable

When set, this variable causes the DVIPDFM program to be used to directly generate ‘`.pdf`’ files from the ‘`.dvi`’ files instead of using DVIPS to generate a Postscript file and then PS2PDF to convert the Postscript to PDF. Please note that the use of this option currently precludes the generation of the -draft versions of PDF files.

USE_HEVEA Variable

When set, this variable causes the HEVEA program to be used to generate HTML output.

USE_LATEX2HTML Variable

When set, this variable causes the LATEX2HTML program to be used to generate HTML output.

USE_PDFLATEX Variable

When set, this variable causes the PDFLATEX program to be used to directly generate ‘`.pdf`’ files from the ‘`.tex`’ files instead of using LATEX to generate a ‘`.dvi`’ file, DVIPS to generate a Postscript file and then PS2PDF to convert the Postscript to PDF. Please note that the use of this option currently precludes the generation of the -draft versions of PDF files.

USE_TEX2PAGE Variable

When set, this variable causes the TEX2PAGE program to be used to generate HTML output.

VIEWPDF Variable

The executible which previews ‘`.pdf`’ files. Defaults to ‘`gv`’.

VIEWPDF_FLAGS Variable

A list of flags to be passed to the VIEWPDF executible. Defaults to ‘’.

VIEWPDF_LANDSCAPE_FLAGS Variable

A list of flags to be added to VIEWPDF_FLAGS when the LANDSCAPE variable is set. Defaults to ‘`-landscape`’.

XDVI Variable

The executible which previews ‘`.dvi`’ files. Defaults to ‘`xdvi`’.

XDVI_FLAGS Variable

A list of flags to be passed to the XDVI executible. Defaults to ‘’.

XDVI_LANDSCAPE_FLAGS Variable

A list of flags to be added to XDVI_FLAGS when the LANDSCAPE variable is set. Defaults to ‘`-paper usr`’.

3.1.3 Lgrind Site Configuration Variables

This section documents the variables related to lgrind source code processing. Lgrind is a source code formatter which takes source code files in various programming languages and formats them for inclusion in a LaTeX document.

LGRIND Variable

The lgrind executible used for formatting source code for inclusion into a LaTeX document. Defaults to '`lgrind`'.

LGRIND_FLAGS Variable

A list of flags to be passed to the LGRIND executible. For example:

```
LGRIND_FLAGS= -i -t 4 -d /my/private/lgrindef
```

Defaults to '`-i`'.

3.1.4 Tgif Site Configuration Variables

This section documents the variables related to Tgif file processing. Tgif (`http://bourbon.usc.edu:8001/tgif/tgif.html`) is a nice vector drawing program which works well with LaTeX. Please note that LaTeX-Mk requires that all Tgif files use the extension '`.obj`'.

TGIF Variable

The tgif executible. Defaults to '`tgif`'.

TGIF_FLAGS Variable

A list of flags to be passed to the TGIF executible to cause it to print to a file. These flags will be used for both PDF and EPS export. Defaults to '`-color -print`'.

TGIF_EPS_FLAGS Variable

A list of flags to be passed to the TGIF executible when exporting to an encapsulated Postscript, '`.eps`', file. Defaults to '`-eps`'.

TGIF_PDF_FLAGS Variable

A list of flags to be passed to the TGIF executible when exporting to a PDF, '`.pdf`', file. Defaults to '`-pdf`'.

3.1.5 Xfig Site Configuration Variables

This section documents the variables related to Xfig file processing. Please note that LaTeX-Mk requires that all Xfig files use the extension '`.fig`'.

FIG2DEV Variable

The executible which can convert xfig '`.fig`' files to encapsulated Postscript '`.eps`' files. Defaults to `fig2dev`.

FIG2DEV_FLAGS Variable

A list of flags to be passed to the FIG2DEV executible to cause it to print to a file. These flags will be used for both PDF and EPS export. Defaults to ''.

FIG2DEV_EPS_FLAGS Variable

A list of flags to be passed to the FIG2DEV executible when exporting to an encapsulated Postscript, '`.eps`', file. Defaults to '`-L eps`'.

FIG2DEV_PDF_FLAGS Variable

A list of flags to be passed to the FIG2DEV executible when exporting to a PDF, '`.pdf`', file. Defaults to '`-L pdf`'.

3.2 Per-Project Variables

This section documents variables which can be set in project Makefiles to accommodate other dependencies which may be added.

3.2.1 Required Variables

Every project must define the `NAME` variable.

NAME Variable

Name of the project. The top level LaTeX input file is assumed to be called '`<NAME>.tex`'. For projects which have multiple documents, you would list the top level name for each document here. For example, if you have a single document, '`mydoc`', you would use

```
NAME= mydoc
```

and if you have multiple documents, '`mydoc1`', '`mydoc2`', and '`mydoc3`', you would use

```
NAME= mydoc1 mydoc2 mydoc3
```

3.2.2 Generic Variables

The variables described in this section affect all of the top level documents listed in the `NAME` variable. This is useful for listing common dependencies like a header or style file used by all documents. To list dependencies which are specific to one of the top level documents, you can use the variable `<docname>_<VARNAME>` where `<docname>` is the name of the document and `<VARNAME>` is the name of the variable. For example,

```
NAME= doc1 doc2
TEXSRCS= header.tex
doc1_TEXSRCS= intro1.tex body1.tex conclusions.tex
```

defines a project with two top level documents, `doc1` and `doc2`. Both documents depend on '`header.tex`' and in addition, `doc1` depends on '`intro1.tex`', '`body1.tex`', and `conclusions.tex`. More information on the `TEXSRCS` variable is given later.

BIBTEXSRCS Variable

All files listed in this variable are assumed to be BibTeX input files. Listing files in this variable will cause a dependency to be added to the top level project and BibTeX will be run on these files as needed.

CLEAN_FILES Variable

Files listed in this variable will be removed when the `clean` target is made. When setting this variable in a '`Makefile`', the `+=` syntax should be used to append to this variable. For example:

```
CLEAN_FILES+=   my_leftover_file foo.bak
```

will add '`my_leftover_file`' and '`foo.bak`' to the list of files to be removed when '`make clean`' is run.

EXTRA_DIST Variable

Files listed in this variable will be added to the archive file created with the `dist` target.

```
EXTRA_DIST+=   README.txt
```

OTHER Variable

Files listed in this variable will be added to the dependency list for the '`.dvi`' file. For example if you want to add all '`.eps`' and '`.epsi`' files in a particular directory as well as some '`.png`' files from another directory to the dependency list, then using BSD make, you could add:

```
OTHER_EPS!=     ls eps/*.eps*
OTHER+=         $(OTHER_EPS)
OTHER_PNG!= ls png/*.png
OTHER+=         $(OTHER_PNG:.png=.eps)
CLEAN_FILES+=   $(OTHER_PNG:.png=.eps)
```

If you are using GNU make, you would use

```
OTHER_EPS=      $(wildcard eps/*.eps*)
OTHER+=         $(OTHER_EPS)
OTHER_PNG=      $(wildcard png/*.png)
OTHER+=         $(OTHER_PNG:.png=.eps)
CLEAN_FILES+=   $(OTHER_PNG:.png=.eps)
```

TEXSRCS Variable

All files listed in this variable are assumed to be LaTeX input files. Listing files in this variable will cause a dependency to be added to the top level project. All LaTeX files used in the project should be listed in this variable with the exception of the top level LaTeX file which is automatically included by LaTeX-Mk.

3.2.3 Per-Project Lgrind Variables

LGRINDSRCS Variable

All files listed in this variable are assumed to be source code files to be processed by lgrind. Listing files in this variable will cause a dependency to be added to the top level project and will cause these files to be automatically re-formatted as required.

LGRINDDIRS Variable

A list of directories containing source code can be listed in this variable. All files found in those directories which have extensions will be formated using lgrind. Files

without a '`.*`' extension will be ignored. These files will be added to the top level dependency list and will be automatically re-formatted as required.

foo_LGRIND_FLAGS Variable

This variable sets specific flags which should be passed to lgrind when processing the source file '`foo`'. For example,

```
mymodule.v_LGRIND_FLAGS=        -lverilog
```

If '`foo`' is a directory which has been listed in LGRINDDIRS, then foo_LGRIND_FLAGS will be used for all files in the specified directory. You can define flags for an entire directory and then override it for a single file if needed. For example, suppose you want to use 4 as the tab width for all sources in the directory '`srcs`' except for '`srcs/funny.c`' where you want to use a tab width of 8. You would achieve this with

```
srcs_LGRIND_FLAGS=              -t 4
srcs/funny.c_LGRIND_FLAGS=      -t 8
```

3.2.4 Per-Project META-post Variables

MPOSTSRCS Variable

All files listed in this variable are assumed to be META-post '`.mp`' files. Listing files in this variable will cause a dependency to be added to the top level project and will cause these files to be automatically re-exported to Postscript and/or PDF as required.

MPOSTDIRS Variable

A list of directories containing META-post figured can be listed in this variable. All '`.mp`' files found in those directories are assumed to be META-post '`.mp`' files. These files will be added to the top level dependency list and will be automatically re-exported to Postscript and/or PDF as required.

MPOST_TWICE Variable

By default META-post is run once for each of its input files. Setting this variable will cause META-post to run twice on each input file. Some figures may require this and I haven't figured out if there is a way to automatically make this determination like there is with LaTeX.

3.2.5 Per-Project Tgif Variables

TGIFSRCS Variable

All files listed in this variable are assumed to be tgif '`.obj`' files. Listing files in this variable will cause a dependency to be added to the top level project and will cause these files to be automatically re-exported to Postscript as required.

TGIFDIRS Variable

A list of directories containing tgif drawings can be listed in this variable. All '`.obj`' files found in those directories are assumed to be tgif '`.obj`' files. These files will be added to the top level dependency list and will be automatically re-exported to Postscript as required.

3.2.6 Per-Project Xfig Variables

XFIGSRCS Variable

All files listed in this variable are assumed to be xfig '`.fig`' files. Listing files in this variable will cause a dependency to be added to the top level project and will cause these files to be automatically re-exported to Postscript as required.

XFIGDIRS Variable

A list of directories containing xfig drawings can be listed in this variable. All '`.fig`' files found in those directories are assumed to be xfig '`.fig`' files. These files will be added to the top level dependency list and will be automatically re-exported to Postscript as required.

4 HTML Output

LaTeX-Mk includes support for generating HTML output. Currently Latex2HTML (see `http://www.latex2html.org`), HeVeA (see `http://para.inria.fr/~maranget/hevea/`), or tex2page (see `http://www.ccs.neu.edu/home/dorai/tex2page/`) can be used for producing an HTML version of your document. The selection of which program to use is done with the USE_HEVEA, USE_LATEX2HTML, and USE_TEX2PAGE variables. Simply define one of these in your '`${sysconfdir}/latex-mk.conf`' file (for site-wide configuration) or '`$HOME/.latex-mk.conf`' (for per-user configuration). If you are using GNU make, the variable would be set in '`${sysconfdir}/latex-gmk.conf`' or '`$HOME/.latex-gmk.conf`' instead. You can also override this setting in your project '`Makefile`'. For example, to use Latex2HTML, add

```
USE_LATEX2HTML=  yes
```

to your configuration file or to your project '`Makefile`'.

To generate HTML output, simply run '`make html`'. The HTML output along with any image files will be placed in a subdirectory called '`${NAME}.html_dir`'. For example, if you have a project with two top level documents, your '`Makefile`' might look like:

```
NAME=  doc1 doc2
.include "/usr/pkg/share/latex-mk/latex.mk"
```

After running '`make html`', you will have two new subdirectories called '`doc1.html_dir`' and '`doc2.html_dir`' containing HTML versions of the two documents.

To keep track of which files have been generated during the conversion, a temporary file, '`${NAME}.www_files`' gets created and all generated files are recorded there. This allows the output produced by HeVeA to be moved to the correct subdirectory as well as allowing '`make clean`' to work.

The HTML generation is still new and there are probably some bugs to work out. Please submit bug reports. There are also some features which may be useful that have not been integrated. For example the program '`hacha`' could be used for breaking the HeVeA output into several smaller files.

5 Using LaTeX-Mk Recursively

In some cases you may wish to organize multiple documents into their own subdirectories but still be able to build all of them with a single make call. This is supported in LaTeX-Mk via the use of the '`latex.subdir.mk`' makefile fragment. To use recursion, create a '`Makefile`', in your top level directory which looks something like the following example.

```
SUBDIR+=        project1
SUBDIR+=        project2
SUBDIR+=        project3

.include "/usr/local/share/latex-mk/latex.subdir.mk"
```

Now create your usual LaTeX-Mk Makefiles in the '`project1`', '`project2`', and '`project3`' subdirectories. Additional subdirectories are added to the `SUBDIR` variable. Multiple levels of subdirectories make be used. Please note that currently the use of both '`latex.mk`' and '`latex.subdir.mk`' in a single Makefile is not supported.

6 Obtaining LaTeX-Mk

The latest information and version of LaTeX-Mk can be found on the main LaTeX-Mk web site at `http://latex-mk.sourceforge.net`. A package for the NetBSD operating system (see `http://www.NetBSD.org` for information about NetBSD) exists at `ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc/print/latex-mk/README.html`. A port for the FreeBSD operating system (see `http://www.FreeBSD.org` for information about FreeBSD) exists at `http://www.freshports.org/misc/latex-mk`.

7 Installing LaTeX-Mk

7.1 System Requirements

To configure and install LaTeX-Mk, you will need a Unix-like operating system or shell with a compatible make program. In addition, to use LaTeX-Mk, you will require:

1. `latex`. The development of LaTeX-Mk was done using Thomas Esser's TEX distribution, teTeX, version 1.0.7. More information on teTeX can be found at `http://www.tug.org/tetex/`.
2. Either GNU make version 3.80 or higher or a BSD make program. For information on GNU make, see `http://www.gnu.org/software/make/`. For information on the NetBSD operating system (which of course includes BSD make), see `http://www.netbsd.org`. If you wish to install BSD make on a non-BSD system, you can try downloading one of the bmake snapshots from files area of the LaTeX-Mk sourceforge project page at `http://www.sourceforge.net/projects/latex-mk`. The GNU make version requirement is firm. LaTeX-Mk will not work with versions of GNU make prior to 3.80. It is highly unlikely that LaTeX-Mk will be ported to older GNU make versions due to the lack of some important features in older versions.

7.2 Installation

Installation of LaTeX-Mk consists of three steps: configuration, building, and installing. In a typical installation, this is as simple as

```
./configure
make
make install
```

This will configure LaTeX-Mk with the defaults, create the final files to be installed, and install them in the proper location. The '`configure`' script is a standard GNU autoconf script. The most common option is the '`--prefix=<installation prefix>`' option. This causes LaTeX-Mk to use `<installation prefix>` as the base directory for the installation.

Running `configure --help` will give a list of the available configuration options. The ones which are specific to LaTeX-Mk, as opposed to being generic configure options are listed here. `--with-mkconf=<mkconf>`: this option changes the default system configuration file for BSD make. This file defaults to '`${sysconfdir}/latex-mk.conf`'. `--with-gmkconf=<gmkconf>`: this option changes the default system configuration file for GNU make. This file defaults to '`${sysconfdir}/latex-gmk.conf`'. `--with-usermkconf=<usermkconf>`: this option changes the default user configuration file for BSD make. This file defaults to '`$HOME/.latex-mk.conf`'. `--with-usergmkconf=<usergmkconf>`: this option changes the default user configuration file for GNU make. This file defaults to '`$HOME/.latex-gmk.conf`'.

8 Feedback

To report bugs, provide feedback, suggest new features, etc. visit the LaTeX-Mk Project management page at `http://www.sourceforge.net/projects/latex-mk` or send email to the author at `danmc@users.sourceforge.net`. For information on the current version of LaTeX-Mk, visit the LaTeX-Mk homepage at `http://latex-mk.sourceforge.net`.

9 Alternatives

There are a few tools which are somewhat similar to LaTeX-Mk. I have not reviewed them in any detail and thus am unable to comment on how similar or different they are compared to LaTeX-Mk.

1. "mk" `http://www.servalys.nl/scripts/`

10 History

10.1 The Dark Ages

In the beginning I used a WYSIWYG word processor from a large software vendor in the Pacific Northwest of the US. It worked for short papers, it was horrible for medium to long documents, painful for equations, and painful for figures. Then I learned LaTeX and life was much much better.

10.2 The Giant Per-Project Makefile

In graduate school, a friend showed me a makefile he had set up for his thesis. It contained all sorts of targets and some intelligence about running LaTeX multiple times for resolving references. I made a modified version of that for my thesis and even wrote a book where I used yet another modified version of that makefile. This approach was much better than doing everything by hand because I had added a lot of functionality over my friends makefile. In particular, my new makefile automatically dealt with tgif figures and I had many many figures in the thesis and the book.

Despite the utility of the large customized makefile I had, it was not maintainable in the sense that every time I started a new document, I'd end up with another copy of a very large makefile to maintain. If I fixed a bug in one, I'd have several other documents in progress which needed updating.

10.3 Enter LaTeX-Mk

For those of you familiar with the build system used by the BSD operating systems, you'll know that for each program, there is a very simple makefile which lists the source files along with a couple of other variables which can optionally be set. Then a system makefile called '`bsd.prog.mk`' is included. That included makefile fragment has all the code required to provide all the standard targets, sets up the various compiler flags and correctly handles all the dependencies. It is maintainable because the bulk of the code is common and only needs to be maintained in one place.

Being inspired by the BSD style makefile approach, I converted my most up to date giant per-project makefile into an `include`-able makefile fragment and spent some time defining the interface a bit more generically than I'd done in the past. Since that time I've used the result, LaTeX-Mk, for the last few documents at school, some papers I've worked on since then and also for work related documentation. So far, the makefile framework has proven to be very useful and a big time saver for me. Since I believe in open-source software I felt it was appropriate to document my efforts and provide a packaged solution that others could also use.

10.4 The Modern Era of LaTeX-Mk

10.4.1 Version 0.9

Released on 2002-10-09, this was the first public release of LaTeX-Mk. My reason for using 0.9 instead of 1.0 is that LaTeX-Mk had not been tested or used much by others yet. Even though it works well for me, as with any product I'm sure others will quickly find other ways to use it that I had not anticipated. My goal is to collect feedback over the first few months of public consumption and come out with a 1.0 version which incorporates the primary improvements.

10.4.2 Version 0.9.1

This is a bug fix version released on 2002-10-29. The significant changes over the previous version are:

- Per-document Xfig dependencies are now supported. This was an oversight in the previous version.
- A big non-portable GNU make hack has been removed. Starting with this version of LaTeX-Mk, you will need version 3.80 or newer of GNU make (run `gmake --version` to check your GNU make version) if you are using the GNU make interface to LaTeX-Mk. The new implementation is much cleaner and should continue to work with all newer versions of GNU make.
- The history section of the manual was added.

10.4.3 Version 1.0

This is the long awaited 1.0 release! Hopefully LaTeX-Mk can be considered production/stable at this point. This release was made on 2003-02-26. The significant changes/additions over the previous version are:

- Support for dvipdfm (see `http://gaspra.kettering.edu/dvipdfm`) is included. By setting the USE_DVIPDFM variable, the `dvipdfm` program can be used to generate '`.pdf`' files from the '`.dvi`' file generated by LaTeX.
- Support for PDFLaTeX is included. By setting the USE_PDFLATEX variable, the `pdflatex` program can be used to generate '`.pdf`' files directly from the TeX sources.
- A testsuite is now included. This has resulted in the fixing of a handful of small bugs and should greatly contribute to the reliability and stability of this tool.
- Bugs related to processing multiple directories listed in TGIFDIRS and XFIGDIRS have been fixed.
- Bugs related to per-project processing of foo_TGIFDIRS and foo_XFIGDIRS have been fixed.
- A bug in `latex-mk` relating to BibTeX has been fixed. Previously, after a BibTeX run, `latex-mk` failed to run LaTeX the correct number of times.

10.4.4 Version 1.1

This is the "HTML Support" release. Version 1.1 was released on 2003-06-15. The significant changes/additions over the previous version are:

- Support for HTML output. Either LaTeX2HTML or HeVeA may be used.

- Fixed a bug where bibtex was not run sometimes when it needed to be run. This problem showed up with some versions of LaTeX.
- Added a POST_BIBTEX_HOOK variable which specifies a program to be run after a BibTeX run. This gives users the ability to insert an additional processing step if desired.

10.4.5 Version 1.2

Version 1.2 was released on 2004-03-21. The significant changes/additions over the previous version are:

- Fixed a bug which prevented the POST_BIBTEX_HOOK hook from actually doing anything.
- Dropped -Ppdf from the default DVIPS_FLAGS. Users who wish to keep -Ppdf as part of DVIPS_FLAGS can add it to the site configuration file, user configuration file, or project Makefile.
- Added DVIPDFM_ENV variable for running `dvipdfm` inside a customized environment.
- Preliminary Rich Text Format (RTF) output support. The new `rtf` target will use `latex2rtf` to produce an RTF version of your document. Use this when sending your documents to the text-formatter-challenged.
- Fixed a bug where a list of figures, list of tables, and table of contents were sometimes not fully up to date in the final output.
- Added support for using ImageMagick to convert JPEG and PNG files to EPS for inclusion in a document.

10.4.6 Version 1.3

Version 1.3 was released on 2004-05-29. The significant changes/additions over the previous version are:

- Fixed a bug which prevented BibTeX from being run in the case where the source document did not have explicit \cite{} commands but rather used \nocite{}. Bug report #927068.
- Fixed some file names in the examples/ directory to avoid a file name clash on file systems which are not case sensitive. This should fix a long standing bug where latexmk would not build under cygwin. Bug report #946216.

10.4.7 Version 1.4

Version 1.4 was released on 2005-10-04. The significant changes/additions over the previous version are:

- Added support for lgrind.
- Added a `dist` target for creating a distribution archive of all source files.
- Added support for using tex2page for html output.
- When using pdflatex, directly convert tgif and xfig drawings to PDF instead of to encapsulated Postscript.

- When using pdflatex, do not create '`.dvi`' files as part of the default target.
- Added a LANDSCAPE variable which when set will add landscape flags to various tools.
- Make the default flag for exporting xfig drawings to encapsulated Postscript be `-L eps` instead of `-L ps`.

10.4.8 Version 1.5

- When using tex2page or HeVeA for html output, do not force the running of LaTeX. With latex2html, LaTeX will still be run because latex2html makes use of the '`.aux`' files generated by LaTeX.
- Fixed a bug where if the BibTeX input file was modified LaTeX would be run again but not BibTeX.
- Added a `--tex2page` mode for `latex-mk` (the script) which allows `latex-mk` to run tex2page the appropriate number of times to resolves all references.
- Improve the cleaning of tex2page generated output.
- Added '`latex.subdir.mk`' to support recursive builds.

10.4.9 Version 1.6

- Fix a syntax error in the (not used yet) ieee-copyout script.
- Fix a bug in the `latex-mk` script when BibTeX is used.
- Add quoting of command names in the BSD makefiles. This will properly deal with pathnames to the programs which contain spaces. Currently GNU make will not properly deal with this.

10.4.10 Version 1.7

- Added support for makeindex. Suggested by Antoine Reilles who provided a preliminary patch.
- Added a `--help` flag to the `latex-mk` script and documented the script a bit more there. It seems that some users are using the `latex-mk` script only and not the makefile system. Suggested by Reuben Thomas.
- Added the ability in the `latex-mk` script to work with a read only current directory and use the TEXMFOUTPUT environment variable to control where the real output goes. Suggested by Reuben Thomas.

10.4.11 Version 1.8

- Fix a syntax error in the `latex-mk` script which showed up with some shells.
- Fix a bug in the `latex-mk` script where some of the '`.old`' files that are used for determining when to re-run various tools were not being cleaned up properly.
- Add a few more files which latex and makeindex may generate to the list of files removed by the `clean` target.
- Add a testsuite entry for the `clean` target.

- Improve the makefile testsuite robustness. In particular it now deals with the bmake program having different names (make, bmake, bsdmake, etc).
- Improve and expand the testsuite for the `latex-mk` script.

10.4.12 Version 1.9

- Added support for per-document DVIPS_FLAGS and DVIPDFM_FLAGS.
- Avoided the use of hardcoded csh in some scripts.
- Removed claims of a BIBTEX_ENV variable in the documentation. It didn't do anything.
- Added METAPOST support.
- Fixed a bug when using BibTeX and PDFLaTeX at the same time.
- Added glossary support.
- Fixed a bug with GNU make draft output.
- Put the actual installation prefix into the manual.
- Avoid a case which caused the testsuite to hang.
- Expanded the testsuite and fixed some bugs in the testsuite.
- Fix a bug in cleaning xfig and tgif output.

10.4.13 Version 1.9.1

- Fixed a bug in the `clean` target when METAPOST is in use.

10.4.14 Version 2.0

- Added support for the bibunits LaTeX package.
- Fixed a bug where PDFLATEX_FLAGS was not being properly passed down to the latex-mk script when bibtex was in use.
- Fixed a bug where exporting Xfig figures to PDF actually produced postscript instead of PDF.

10.4.15 Version 2.1

- Fixed a bug in the quoting of lgrind related variables. The result is that the clean target didn't quite work right.
- Improved quoting in the latex-mk script to better handle the case of a file name with spaces in it.

Target Index

C

clean 3

D

dist 3
dvi 3

H

html 3

P

pdf 3
pdf-draft 4
print 3
print-draft 4
print_<name> 5
print_<name>-draft 5
ps 3
ps-draft 4
ps_<name>-draft 5

R

rtf 3

S

show-var 4

V

view 4
view_<name> 5
view_<name>-draft 5
viewpdf 4
viewpdf-draft 4
viewpdf_<name> 5
viewpdf_<name>-draft 5
viewps 4
viewps-draft 4
viewps_<name> 5
viewps_<name>-draft 5

Variable Index

B

BIBTEX . . . 6
BIBTEX_ENV . . . 6
BIBTEX_FLAGS . . . 7
BIBTEXSRCS . . . 13

C

CLEAN_FILES . . . 14
CONVERT . . . 7

D

DVIPDFM . . . 7
DVIPDFM_ENV . . . 7
DVIPDFM_FLAGS . . . 7
DVIPDFM_LANDSCAPE_FLAGS . . . 7
DVIPS . . . 7
DVIPS_ENV . . . 7
DVIPS_FLAGS . . . 7
DVIPS_LANDSCAPE_FLAGS . . . 7

E

EXTRA_DIST . . . 14

F

FIG2DEV . . . 12
FIG2DEV_EPS_FLAGS . . . 13
FIG2DEV_FLAGS . . . 13
FIG2DEV_PDF_FLAGS . . . 13
foo_LGRIND_FLAGS . . . 15

G

GV . . . 8
GV_FLAGS . . . 8
GV_LANDSCAPE_FLAGS . . . 8
GZCAT . . . 8
GZIP . . . 8

H

HEVEA . . . 8
HEVEA_ENV . . . 8
HEVEA_FLAGS . . . 8

I

IMAGEN . . . 8

J

JPG2EPS . . . 8

L

LATEX . . . 8
LATEX_ENV . . . 8
LATEX_FLAGS . . . 8
LATEX2HTML . . . 9
LATEX2HTML_ENV . . . 9
LATEX2HTML_FLAGS . . . 9
LATEX2RTF . . . 9
LATEX2RTF_ENV . . . 9
LATEX2RTF_FLAGS . . . 9
LGRIND . . . 12
LGRIND_FLAGS . . . 12
LGRINDDIRS . . . 14
LGRINDSRCS . . . 14
LPR . . . 9
LPR_FLAGS . . . 9

M

MAKECONF . . . 6
MAKEGLS . . . 9
MAKEGLS_FLAGS . . . 9
MAKEIDX . . . 9
MAKEIDX_FLAGS . . . 9
MPOST . . . 9
MPOST_FLAGS . . . 10
MPOST_TWICE . . . 15
MPOSTDIRS . . . 15
MPOSTSRCS . . . 15

N

NAME . . . 13

O

OTHER . . . 14

P

PDFLATEX . . . 10
PDFLATEX_ENV . . . 10
PDFLATEX_FLAGS . . . 10
PNG2EPS . . . 10
POST_BIBTEX_HOOK . . . 10
PS2PDF . . . 10
PS2PDF_FLAGS . . . 10

T

TAR 10
TEX2PAGE 10
TEX2PAGE_ENV 10
TEX2PAGE_FLAGS 11
TEXSRCS 14
TGIF 12
TGIF_EPS_FLAGS 12
TGIF_FLAGS 12
TGIF_PDF_FLAGS 12
TGIFDIRS 16
TGIFSRCS 15

U

USE_DVIPDFM 11
USE_HEVEA 11
USE_LATEX2HTML 11
USE_PDFLATEX 11
USE_TEX2PAGE 11
USER_MAKECONF 6

V

VIEWPDF 11
VIEWPDF_FLAGS 11
VIEWPDF_LANDSCAPE_FLAGS 11

X

XDVI 11
XDVI_FLAGS 11
XDVI_LANDSCAPE_FLAGS 11
XFIGDIRS 16
XFIGSRCS 16

General Index

A

alternatives to LaTeX-Mk 22

B

BSD make, versus GNU make 1, 14
BUGS, reporting 21

E

E-mail, bug reports 21

G

GNU make, version required 24
GNU make, versus BSD make 1, 14

I

include, Makefile syntax for 1

M

make, differences between GNU and BSD . . . 1, 14

R

Reporting BUGS 21

W

wildcard, Makefile syntax for 14