UseLATEX.cmake: LaTeX Document Building Made Easy

Kenneth Moreland

Version 1.7.1

Contents

1 Description

Compiling LaTeX files into readable documents is actually a very involved process. Although CMake comes with FindLATEX.cmake, it does nothing for you other than find the commands associated with LaTeX. I like using CMake to build my LaTeX documents, but creating targets to do it is actually a pain. Thus, I've compiled a bunch of macros that help me create targets in CMake into a file I call "UseLATEX.cmake." Here are some of the things UseLATEX.cmake handles:

- Runs LaTeX multiple times to resolve links.
- Can run bibtex, makeindex, and makeglossaries to make bibliographies, indexes, and/or glossaries.
- Optionally runs configure on your LaTeX files to replace `@VARIABLE@` with the equivalent CMake variable.
- Automatically finds png, jpeg, eps, and pdf files and converts them to formats latex and pdflatex understand.

2 Download

UseLATEX.cmake is currently posted to the CMake Wiki at

http://public.kitware.com/Wiki/CMakeUserUseLATEX.

3 Usage

Using UseLATEX.cmake is easy. For a basic LaTeX file, simply include the file in your CMakeLists.txt and use the `ADD_LATEX_DOCUMENT` command to make targets to build your document. For an example document in the file MyDoc.tex, you could establish a build with the following simple CMakeLists.txt.

```
PROJECT(MyDoc NONE)

INCLUDE(UseLATEX.cmake)
ADD_LATEX_DOCUMENT(MyDoc.tex)
```

The `ADD_LATEX_DOCUMENT` adds the following targets to create a readable document from MyDoc.tex:

dvi Creates MyDoc.dvi.

pdf Creates MyDoc.pdf using pdflatex. Requires the `PDFLATEX_COMPILER` CMake variable to be set.

ps Creates MyDoc.ps. Requires the `DVIPS_CONVERTER` CMake variable to be set.

safepdf Creates MyDoc.pdf from MyDoc.ps using ps2pdf. Many publishers prefer pdfs are created this way. Requires the `PS2PDF_CONVERTER` CMake variable to be set.

html Creates html pages. Requires the `LATEX2HTML_CONVERTER` CMake variable to be set.

One caveat about using UseLATEX.cmake is that you are required to do an out-of-source build. That is, CMake must be run in a directory other than the source directory. This is necessary as latex is very picky about file locations, and the relative locations of some generated or copied files can only be maintained if everything is copied to a separate directory structure.

3.1 Using a Bibliography

For any technical document, you will probably want to maintain a BibTeX database of papers you are referencing in the paper. You can incorporate your .bib files by adding them after the `BIBFILES` argument to the `ADD_LATEX_DOCUMENT` command.

```
ADD_LATEX_DOCUMENT(MyDoc.tex BIBFILES MyDoc.bib)
```

This will automatically add targets to build your bib file and link it into your document. To use the BibTeX file in your LaTeX file, just do as you normally would with `\cite` commands and bibliography commands:

```
\bibliographystyle{plain}
\bibliography{MyDoc}
```

You can list as many bibliography files as you like.

3.2 Incoporating Images

To be honest, incorporating images into LaTeX documents can be a real pain. This is mostly because the format of the images needs to depend on the version of LaTeX you are running (latex vs. pdflatex). With these CMake macros, you only need to convert your raster graphics to png or jpeg format and your vector graphics to eps or pdf format. Place them all in a common directory (e.g. images) and then use the `IMAGE_DIRS` option to the `ADD_LATEX_DOCUMENT` macro to point to them. UseLATEX.cmake will take care of the rest.

```
ADD_LATEX_DOCUMENT(MyDoc.tex BIBFILES MyDoc.bib
                   IMAGE_DIRS images)
```

If you want to break up your image files in several different directories, you can do that, too. Simply provide multiple directories after the IMAGE_DIRS command.

```
ADD_LATEX_DOCUMENT(MyDoc.tex BIBFILES MyDoc.bib
                   IMAGE_DIRS icons figures)
```

Alternatively, you could list all of your image files separatly with the IMAGES option.

```
SET(MyDocImages
  logo.eps
  icons/next.png
  icons/previous.png
  figures/flowchart.eps
  figures/team.jpeg
  )
ADD_LATEX_DOCUMENT(MyDoc.tex IMAGES ${MyDocImages})
```

Both the IMAGE_DIRS and IMAGES can be used together. The combined set of image files will be processed. If you wish to provide a separate eps file and pdf or png file, that is OK, too. UseLATEX.cmake will handle that by copying over the correct file instead of converting.

Once you establish the images directory, CMake will automatically find all files with known image extensions (currently eps, pdf, png, jpeg, and jpg) in it and add makefile targets to use ImageMagick's convert to convert the file times to those appropriate for the build. If you do not have ImageMagick, you can get it for free from http://www.imagemagick.org. CMake will also give you a LATEX_SMALL_IMAGES option that, when on, will downsample raster images. This can help speed up building and viewing documents. It will also make the output image sizes smaller.

Depending on what program is launched to build your LaTeX file (either latex or pdflatex, and UseLATEX.cmake supports both), a particular format for your image is required. As stated, UseLATEX.cmake handles the necessary conversions for you. However, you will not know in advance what file extension is used on the image. That is no problem. Simply leave out the file extension in the file name argument to \includegraphics and LaTeX will find the file with the appropriate extension for you.

One more note about vector graphics. Encapsulated postscript (eps) files have a bounding box that is often lost when converting to pdf types. When using eps files, it is best to search for a line starting with %%BoundingBox: such as

```
%%BoundingBox: 58 77 734 536
```

and then copy these numbers to the `bb` option of the LaTeX `\includegraphics` command:

```
\includegraphics[width=\linewidth,bb=58 77 734 536]
```

This problem does not seem to occur when converting pdf files to eps. Thus, I find it more convenient to create my vector graphics as pdf. This, of course, is assuming that your program gives you enough control over the pdf output to adjust the page size.

3.3 Create a PDF by Default

By default, when you use `ADD_LATEX_DOCUMENT` and then run make with no arguments, the dvi file will be created. You have to specifically build the pdf target to use pdflatex to create a pdf file. However, oftentimes we want the pdf to be generated by default. To do that, simply use the `DEFAULT_PDF` option to `ADD_LATEX_DOCUMENT`:

```
ADD_LATEX_DOCUMENT(MyDoc.tex BIBFILES MyDoc.bib
                   IMAGE_DIRS images
                   DEFAULT_PDF)
```

3.4 Building Multiple LaTeX Documents

The most commen use for UseLATEX.cmake is to build a single document, such as a paper you are working on. However, some use cases involve building several documents at one time. To do this, you must call `ADD_LATEX_DOCUMENT` multiple times. However, if you do this, the dvi, pdf, etc. targets will be generated multiple times, and that is illegal in the current version of CMake.[1] To get around this, you need to mangle the names of the targets that `ADD_LATEX_DOCUMENT` creates. To do this, use the `MANGLE_TARGET_NAMES` option.

```
ADD_LATEX_DOCUMENT(MyDoc1.tex MANGLE_TARGET_NAMES)
ADD_LATEX_DOCUMENT(MyDoc2.tex MANGLE_TARGET_NAMES)
```

In the example above, the first call to `ADD_LATEX_DOCUMENT` will create targets named MyDoc1_dvi, MyDoc1_pdf, MyDoc1_ps, etc. whereas the second call will create targets named MyDoc2_*.

If you still want the simple, short targets to build all of the documents, you can add them yourself with custom targets that depend on the targets created by `ADD_LATEX_DOCUMENT`

[1] CMake version 2.4 as of this writing.

```
ADD_CUSTOM_TARGET(dvi)
ADD_DEPENDENCIES(MyDoc1_dvi MyDoc2_dvi)
ADD_CUSTOM_TARGET(pdf)
ADD_DEPENDENCIES(MyDoc1_pdf MyDoc2_pdf)
ADD_CUSTOM_TARGET(ps)
ADD_DEPENDENCIES(MyDoc1_ps MyDoc2_ps)
```

3.5 Making an Index

You can make an index in a LaTeX document by using the makeidx package. However, this package requires you to run the makeindex command. Simply add the `USE_INDEX` option anywhere in the `ADD_LATEX_DOCUMENT` arguments, and makeindex will automatically be added to the build.

```
ADD_LATEX_DOCUMENT(MyDoc.tex BIBFILES MyDoc.bib
                   IMAGE_DIRS images
                   USE_INDEX)
```

3.6 Making a Glossary

There are multiple ways to make a glossary in a LaTeX document, but the glossaries package provides one of the most convenient ways of doing so. Like the makeidx package, glossaries requires running makeindex for building auxiliary files. However, building the glossary files can be more complicated as there can be different sets of glossary files with different extensions. UseLATEX.cmake will handle that for you. Simply add the `USE_GLOSSARY` option anywhere in the `ADD_LATEX_DOCUMENT` arguments, and the glossary creating will be handled for you.

```
ADD_LATEX_DOCUMENT(MyDoc.tex BIBFILES MyDoc.bib
                   IMAGE_DIRS images
                   USE_GLOSSARY)
```

3.7 Multipart LaTeX Files

Often, it is convenient to split a LaTeX document into multiple files and use the LaTeX `\input` or `\include` command to put them back together. To do this, all the files have to be located together. UseLATEX.cmake can take care of that, too. Simply add the `INPUTS` argument to `ADD_LATEX_DOCUMENT` to copy these files along with the target tex file. Build dependencies to these files is also established.

```
ADD_LATEX_DOCUMENT(MyDoc.tex
  INPUTS Chapter1.tex Chapter2.tex Chapter3.tex Chapter4.tex
  BIBFILES MyDoc.bib
  IMAGE_DIRS images
  USE_INDEX
  )
```

As far as UseLATEX.cmake is concerned, input files do not necessarily have to be tex files. For example, you might be including the contents of a text file into your document with the `\VerbatimInput` command of the fancyvrb package. In fact, you could also add graphic files as inputs, but you would not get the extra conversion features described in Section 3.2.

3.8 Configuring LaTeX Files

Sometimes it is convenient to control the build options of your tex file with CMake variables. You can achieve this by using the `CONFIGURE` argument to `ADD_LATEX_DOCUMENT`.

```
ADD_LATEX_DOCUMENT(MyDoc.tex
  INPUTS Chapter1.tex Chapter2.tex Chapter3.tex Chapter4.tex
  CONFIGURE MyDoc.tex
  BIBFILES MyDoc.bib
  IMAGE_DIRS images
  USE_INDEX
  )
```

In the above example, in addition to copying MyDoc.tex to the binary directory, UseLATEX.cmake will configure MyDoc.tex. That is, it will find all occurrences of `@`*`VARIABLE`*`@` and replace that string with the current CMake variable *`VARIABLE`*.

With the `CONFIGURE` argument you can list the target tex file (as shown above) as well as any other tex file listed in the `INPUTS` argument.

```
ADD_LATEX_DOCUMENT(MyDoc.tex
  INPUTS Ch1Config.tex Ch1.tex Ch2Config.tex
         Ch2.tex Ch3Config Ch3.tex
  CONFIGURE Ch1Config.tex Ch2Config.tex Ch3Config.tex
  BIBFILES MyDoc.bib
  IMAGE_DIRS images
  USE_INDEX
  )
```

Be careful when using the `CONFIGURE` option. Unfortunately, the `@` symbol is used by LaTeX in some places. For example, when establishing a tabular environment, an `@` is used to define the space between columns. If you use it more than once, then UseLATEX.cmake will erroneously replace part of the definition of your columns for a macro (which is probably an empty string). This can be particularly troublesome to debug as LaTeX will give an error in a place that, in the original document, is legal. Hence, it is best to only configure tex files that contain very little text of the actual document and instead are mostly setup and options.

3.9 Identifying Dependent Files

In some circumstances, CMake's configure mechanism is not sufficient for creating input files. Input LaTeX files might be auto-generated by any number of other mechanisms.

If this is the case, simply add the appropriate CMake commands to generate the input files, and then add that file to the DEPENDS option of `ADD_LATEX_DOCUMENT`. To help you build the CMake commands to place the generated files in the correct place, you can use the LATEX_GET_OUTPUT_PATH convenience function to get the output path.

```
LATEX_GET_OUTPUT_PATH(output_dir)

ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/generated_file.tex
  COMMAND tex_file_generate_exe
  ARGS ${output_dir}/generated_file.tex
  )

ADD_LATEX_DOCUMENT(MyDoc.tex DEPENDS generated_file.tex)
```

4 Frequently Asked Questions

This section includes resolutions to common questions and issues concerning use of UseLATEX.cmake and with LaTeX in general.

4.1 How do I process LaTeX files on Windows?

I have successfully used two different ports of LaTeX for windows: the cygwin port (http://www.cygwin.com/) and the MikTeX port (http://www.miktex.org/).

If you use the cygwin port of LaTeX, you must also use the cygwin port of CMake, make, and ImageMagick. If you use the MikTeX port of LaTeX, you must use the CMake from http://www.cmake.org/HTML/Download.html, the ImageMagick port from http://www.imagemagick.org/script/index.php,

and a native build tool like MSVC or the GNU make port at http://unxutils.sourceforge.net/. *Do not use the "native" CMake program with any cygwin programs or the cygwin CMake program with any non-cygwin programs.* This issue at hand is that the cygwin ports create and treat filenames differently then other windows programs.[2]

Also be aware that if you have images in your document, there are numerous problems that can occur on Windows with the ImageMagick convert program. See Section 4.4 for more information.

4.2 How do I process LaTeX files on Mac OS X?

Using LaTeX on Mac OS X is fairly straightforward because this OS is built on top of Unix. By using the Terminal program or X11 host, you can run LaTeX much like any other Unix variant. The only real issue is that LaTeX and some of the supporting programs like CMake and ImageMagick are not typically installed (whereas on Linux they often are).

Most applications port fairly easily to Mac OS so long as you are willing to use them as typical Unix or X11 programs. To make things even easier, I recommend taking advantage of a Mac porting project to make this process even easier. MacPorts (http://www.macports.org) is a good tool providing a comprehensive set of tool ports including LaTeX, CMake, and ImageMagick. The fink project and FinkCommander (http://finkcommander.sourceforge.net/) is a similar although less active project.

4.3 Why does UseLATEX.cmake have to copy my tex files?

UseLATEX.cmake cannot process your tex file without copying it. As explained in Section 3, LaTeX is very picky about file locations. The relative locations of files that your input files point to, and all but the most simple LaTeX files point to other files, must remain consistent.

UseLATEX.cmake will often have to modify at least one file either through configurations or image format and size conversions. When creating new files, UseLATEX.cmake will have to copy either all of the files or none of the files. Since configuring and writing over an original file is unacceptable, UseLATEX.cmake forces you to configure it such that LaTeX builds in a different directory than where you have placed the original. If you do not specify a seperate directory, you get an error like the following.

```
CMake Error at UseLATEX.cmake:377 (MESSAGE):
  LaTeX files must be built out of source or you must set
  LATEX_OUTPUT_PATH.
```

[2] If you are careful, you can use the cygwin version of make with the windows ports of CMake, LaTeX, and ImageMagick. It is an easy way around the problems described in Section 4.4.

The best way around this problem is do an "out of source" build, which is really the preferred method of using CMake in general. To do an out of source build, create a new build directory, go to that directory, and run cmake from there, pointing to the source directory.

If for some reason an out of source build is not feasable or desireable, you can set the `LATEX_OUTPUT_PATH` variable to a directory other than . (the local directory). If you are building a LaTeX document in the context of a larger project for which you wish to support in source builds, consider pragmatically setting the `LATEX_OUTPUT_PATH` CMake cache variable from within your CMakeLists.txt.

4.4 Why is convert failing on Windows?

Assuming that you have correctly downloaded and installed an appropriate version of ImageMagick (as specified in Section 4.1), there are several other problems that users can run into the created build files attempt to run the convert program.

A common error seen is

```
Invalid Parameter - filename
```

This is probably because CMake has found the wrong convert program. Windows is installed with a program named convert in %SYSTEMROOT%\system32. This convert program is used to change the filesystem type on a hard drive. Since the windows convert is in a system binary directory, it is usually found in the path before the installed ImageMagick convert program. (Don't get me started about the logic behind this.) Make sure that the `IMAGEMAGICK_CONVERT` CMake variable is pointing to the correct convert program.

Another common error is that convert not finding a file that is clearly there.

```
convert: unable to open image 'filename'
```

If you notice that the drive letter is stripped off of the filename (i.e. C:), then you are probably mixing the Cygwin version of convert with the non-cygwin CMake. The cygwin version of convert uses the colon (:), as a directory separator for inputs. Thus, it assumes the output file name is really two input files separated by the colon. Switch to the non-cygwin port of ImageMagick to fix this.

If you are using nmake, you may also see the following error:

```
convert.exe: unable to open image 'C:': Permission denied.
```

I don't know what causes this error, but it appears to have something to do with some strange behavior of nmake when quoting the convert executable. The easiest solution is to use a different build program (such as make or MSVC's IDE or a unix port of make). If anyone finds away around this problem, please contribute back.

4.5 Why does make stop after each image conversion?

There is a bug in the ImageMagick convert version 6.1.8 that inappropriatly returns a failure condition even when the image convert was successful. The problem might also occur in other ImageMagick versions. Try updating your installation of ImageMagick.

5 Acknowledgments

Thanks to all of the following contributors.

Alin Elena Suggestions on removing dependence on makeglossaries command.

Øystein S. Haaland Support for making glossaries.

Eric Noulard Support for any file extension on LaTeX input files.

Theodore Papadopoulo DEPENDS parameter for `ADD_LATEX_DOCUMENT` and help in identifying some dependency issues.

This work was primarily done at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

This document released as document SAND 2008-2743P.

A Sample CMakeLists.txt

Following is a sample listing of CMakeLists.txt. In fact, it is the CMakeLists.txt that is used to build this document.

```
PROJECT(UseLATEX_DOC NONE)

CMAKE_MINIMUM_REQUIRED(VERSION 2.4)

INCLUDE(UseLATEX.cmake)

# Note that normally neither CMakeLists.txt nor UseLATEX.cmake
# would be considered inputs to the document, but in this
# special case (of documenting UseLATEX.cmake) the contents of
# these files are actually included in the document.
ADD_LATEX_DOCUMENT(UseLATEX.tex
  INPUTS CMakeLists.txt UseLATEX.cmake
  )
```

B UseLATEX.cmake Listing

```
# File: UseLATEX.cmake
# CMAKE commands to actually use the LaTeX compiler
# Version: 1.7.1
# Author: Kenneth Moreland (kmorel at sandia dot gov)
#
# Copyright 2004 Sandia Corporation.
# Under the terms of Contract DE-AC04-94AL85000, there is a non-exclusive
# license for use of this work by or on behalf of the
# U.S. Government. Redistribution and use in source and binary forms, with
# or without modification, are permitted provided that this Notice and any
# statement of authorship are reproduced on all copies.
#
# The following MACROS are defined:
#
# ADD_LATEX_DOCUMENT(<tex_file>
#                       [BIBFILES <bib_files>]
#                       [INPUTS <input_tex_files>]
#                       [IMAGE_DIRS] <image_directories>
#                       [IMAGES] <image_files>
#                       [CONFIGURE] <tex_files>
#                       [DEPENDS] <tex_files>
#                       [USE_INDEX] [USE_GLOSSARY]
#                       [DEFAULT_PDF] [MANGLE_TARGET_NAMES])
#       Adds targets that compile <tex_file>.  The latex output is placed
#       in LATEX_OUTPUT_PATH or CMAKE_CURRENT_BINARY_DIR if the former is
#       not set.  The latex program is picky about where files are located,
#       so all input files are copied from the source directory to the
#       output directory.  This includes the target tex file, any tex file
#       listed with the INPUTS option, the bibliography files listed with
#       the BIBFILES option, and any .cls, .bst, and .clo files found in
#       the current source directory.  Images found in the IMAGE_DIRS
#       directories or listed by IMAGES are also copied to the output
#       directory and coverted to an appropriate format if necessary.  Any
#       tex files also listed with the CONFIGURE option are also processed
#       with the CMake CONFIGURE_FILE command (with the @ONLY flag.  Any
#       file listed in CONFIGURE but not the target tex file or listed with
#       INPUTS has no effect. DEPENDS can be used to specify generated files
#       that are needed to compile the latex target.
#
#       The following targets are made:
#               dvi: Makes <name>.dvi
#               pdf: Makes <name>.pdf using pdflatex.
#               safepdf: Makes <name>.pdf using ps2pdf.  If using the default
#                       program arguments, this will ensure all fonts are
#                       embedded and no lossy compression has been performed
#                       on images.
#               ps: Makes <name>.ps
#               html: Makes <name>.html
#               auxclean: Deletes <name>.aux.  This is sometimes necessary
#                       if a LaTeX error occurs and writes a bad aux file.
#
#       The dvi target is added to the ALL.  That is, it will be the target
#       built by default.  If the DEFAULT_PDF argument is given, then the
#       pdf target will be the default instead of dvi.
#
```

```
#       If the argument MANGLE_TARGET_NAMES is given, then each of the
#       target names above will be mangled with the <tex_file> name.  This
#       is to make the targets unique if ADD_LATEX_DOCUMENT is called for
#       multiple documents.  If the argument USE_INDEX is given, then
#       commands to build an index are made.  If the argument USE_GLOSSARY
#       is given, then commands to build a glossary are made.
#
# History:
#
# 1.7.1 Fixed some dependency issues.
#
# 1.7.0 Added DEPENDS options (thanks to Theodore Papadopoulo).
#
# 1.6.1 Ported the makeglossaries command to CMake and embedded the port
#       into UseLATEX.cmake.
#
# 1.6.0 Allow the use of the makeglossaries command.  Thanks to Oystein
#       S. Haaland for the patch.
#
# 1.5.0 Allow any type of file in the INPUTS lists, not just tex file
#       (suggested by Eric Noulard).  As a consequence, the ability to
#       specify tex files without the .tex extension is removed.  The removed
#       function is of dubious value anyway.
#
#       When copying input files, skip over any file that exists in the
#       binary directory but does not exist in the source directory with the
#       assumption that these files were added by some other mechanism.  I
#       find this useful when creating large documents with multiple
#       chapters that I want to build separately (for speed) as I work on
#       them.  I use the same boilerplate as the starting point for all
#       and just copy it with different configurations.  This was what the
#       separate ADD_LATEX_DOCUMENT method was supposed to originally be for.
#       Since its external use is pretty much deprecated, I removed that
#       documentation.
#
# 1.4.1 Copy .sty files along with the other class and package files.
#
# 1.4.0 Added a MANGLE_TARGET_NAMES option that will mangle the target names.
#
#       Fixed problem with copying bib files that became apparent with
#       CMake 2.4.
#
# 1.3.0 Added a LATEX_OUTPUT_PATH variable that allows you or the user to
#       specify where the built latex documents to go.  This is especially
#       handy if you want to do in-source builds.
#
#       Removed the ADD_LATEX_IMAGES macro and absorbed the functionality
#       into ADD_LATEX_DOCUMENT.  The old interface was always kind of
#       clunky anyway since you had to specify the image directory in both
#       places.  It also made supporting LATEX_OUTPUT_PATH problematic.
#
#       Added support for jpeg files.
#
# 1.2.0 Changed the configuration options yet again.  Removed the NO_CONFIGURE
#       Replaced it with a CONFIGURE option that lists input files for which
#       configure should be run.
#
```

```
#       The pdf target no longer depends on the dvi target.  This allows you
#       to build latex documents that require pdflatex.  Also added an option
#       to make the pdf target the default one.
#
# 1.1.1 Added the NO_CONFIGURE option.  The @ character can be used when
#       specifying table column separators.  If two or more are used, then
#       will incorrectly substitute them.
#
# 1.1.0 Added ability include multiple bib files.  Added ability to do copy
#       sub-tex files for multipart tex files.
#
# 1.0.0 If both ps and pdf type images exist, just copy the one that
#       matches the current render mode.  Replaced a bunch of STRING
#       commands with GET_FILENAME_COMPONENT commands that were made to do
#       the desired function.
#
# 0.4.0 First version posted to CMake Wiki.
#

#############################################################################
# Find the location of myself while originally executing.  If you do this
# inside of a macro, it will recode where the macro was invoked.
#############################################################################
SET(LATEX_USE_LATEX_LOCATION ${CMAKE_CURRENT_LIST_FILE}
  CACHE INTERNAL "Location of UseLATEX.cmake file." FORCE
  )

#############################################################################
# Generic helper macros
#############################################################################

# Helpful list macros.
MACRO(LATEX_CAR var)
  SET(${var} ${ARGV1})
ENDMACRO(LATEX_CAR)
MACRO(LATEX_CDR var junk)
  SET(${var} ${ARGN})
ENDMACRO(LATEX_CDR)

MACRO(LATEX_LIST_CONTAINS var value)
  SET(${var})
  FOREACH (value2 ${ARGN})
    IF (${value} STREQUAL ${value2})
      SET(${var} TRUE)
    ENDIF (${value} STREQUAL ${value2})
  ENDFOREACH (value2)
ENDMACRO(LATEX_LIST_CONTAINS)

# Parse macro arguments.
MACRO(LATEX_PARSE_ARGUMENTS prefix arg_names option_names)
  SET(DEFAULT_ARGS)
  FOREACH(arg_name ${arg_names})
    SET(${prefix}_${arg_name})
  ENDFOREACH(arg_name)
  FOREACH(option ${option_names})
    SET(${prefix}_${option})
  ENDFOREACH(option)
```

```
  SET(current_arg_name DEFAULT_ARGS)
  SET(current_arg_list)
  FOREACH(arg ${ARGN})
    LATEX_LIST_CONTAINS(is_arg_name ${arg} ${arg_names})
    IF (is_arg_name)
      SET(${prefix}_${current_arg_name} ${current_arg_list})
      SET(current_arg_name ${arg})
      SET(current_arg_list)
    ELSE (is_arg_name)
      LATEX_LIST_CONTAINS(is_option ${arg} ${option_names})
      IF (is_option)
        SET(${prefix}_${arg} TRUE)
      ELSE (is_option)
        SET(current_arg_list ${current_arg_list} ${arg})
      ENDIF (is_option)
    ENDIF (is_arg_name)
  ENDFOREACH(arg)
  SET(${prefix}_${current_arg_name} ${current_arg_list})
ENDMACRO(LATEX_PARSE_ARGUMENTS)

# Match the contents of a file to a regular expression.
MACRO(LATEX_FILE_MATCH variable filename regexp default)
  # The FILE STRINGS command would be a bit better, but it's not supported on
  # older versions of CMake.
  FILE(READ ${filename} file_contents)
  STRING(REGEX MATCHALL "${regexp}"
    ${variable} ${file_contents}
    )
  IF (NOT ${variable})
    SET(${variable} "${default}")
  ENDIF (NOT ${variable})
ENDMACRO(LATEX_FILE_MATCH)

##############################################################################
# Macros that perform processing during a LaTeX build.
##############################################################################
MACRO(LATEX_MAKEGLOSSARIES)
  MESSAGE("*************************** In makeglossaries")
  IF (NOT LATEX_TARGET)
    MESSAGE(SEND_ERROR "Need to define LATEX_TARGET")
  ENDIF (NOT LATEX_TARGET)

  IF (NOT MAKEINDEX_COMPILER)
    MESSAGE(SEND_ERROR "Need to define MAKEINDEX_COMPILER")
  ENDIF (NOT MAKEINDEX_COMPILER)

  SET(aux_file ${LATEX_TARGET}.aux)

  IF (NOT EXISTS ${aux_file})
    MESSAGE(SEND_ERROR "${aux_file} does not exist.  Run latex on your target file.")
  ENDIF (NOT EXISTS ${aux_file})

  LATEX_FILE_MATCH(newglossary_lines ${aux_file}
    "@newglossary[ \t]*{([^}]*)}{([^}]*)}{([^}]*)}{([^}]*)}"
    "@newglossary{main}{glg}{gls}{glo}"
    )
```

```
  LATEX_FILE_MATCH(istfile_line ${aux_file}
    "@istfilename[ \t]*{([^}]*)}"
    "@istfilename{${LATEX_TARGET}.ist}"
    )
  STRING(REGEX REPLACE "@istfilename[ \t]*{([^}]*)}" "\\1"
    istfile ${istfile_line}
    )

  FOREACH(newglossary ${newglossary_lines})
    STRING(REGEX REPLACE
      "@newglossary[ \t]*{([^}]*)}{([^}]*)}{([^}]*)}{([^}]*)}"
      "\\1" glossary_name ${newglossary}
      )
    STRING(REGEX REPLACE
      "@newglossary[ \t]*{([^}]*)}{([^}]*)}{([^}]*)}{([^}]*)}"
      "${LATEX_TARGET}.\\2" glossary_log ${newglossary}
      )
    STRING(REGEX REPLACE
      "@newglossary[ \t]*{([^}]*)}{([^}]*)}{([^}]*)}{([^}]*)}"
      "${LATEX_TARGET}.\\3" glossary_out ${newglossary}
      )
    STRING(REGEX REPLACE
      "@newglossary[ \t]*{([^}]*)}{([^}]*)}{([^}]*)}{([^}]*)}"
      "${LATEX_TARGET}.\\4" glossary_in ${newglossary}
      )
    MESSAGE("${MAKEINDEX_COMPILER} ${MAKEGLOSSARIES_COMPILER_FLAGS} -s ${istfile} -t ${glossary_log} -o ${glossa
    EXEC_PROGRAM(${MAKEINDEX_COMPILER} ARGS ${MAKEGLOSSARIES_COMPILER_FLAGS}
      -s ${istfile} -t ${glossary_log} -o ${glossary_out} ${glossary_in}
      )
  ENDFOREACH(newglossary)
ENDMACRO(LATEX_MAKEGLOSSARIES)

#############################################################################
# Helper macros for establishing LaTeX build.
#############################################################################

MACRO(LATEX_NEEDIT VAR NAME)
  IF (NOT ${VAR})
    MESSAGE(SEND_ERROR "I need the ${NAME} command.")
  ENDIF(NOT ${VAR})
ENDMACRO(LATEX_NEEDIT)

MACRO(LATEX_WANTIT VAR NAME)
  IF (NOT ${VAR})
    MESSAGE(STATUS "I could not find the ${NAME} command.")
  ENDIF(NOT ${VAR})
ENDMACRO(LATEX_WANTIT)

MACRO(LATEX_SETUP_VARIABLES)
  SET(LATEX_OUTPUT_PATH "${LATEX_OUTPUT_PATH}"
    CACHE PATH "If non empty, specifies the location to place LaTeX output."
    )

  FIND_PACKAGE(LATEX)

  MARK_AS_ADVANCED(CLEAR
```

```
  LATEX_COMPILER
  PDFLATEX_COMPILER
  BIBTEX_COMPILER
  MAKEINDEX_COMPILER
  DVIPS_CONVERTER
  PS2PDF_CONVERTER
  LATEX2HTML_CONVERTER
  )

LATEX_NEEDIT(LATEX_COMPILER latex)
LATEX_WANTIT(PDFLATEX_COMPILER pdflatex)
LATEX_NEEDIT(BIBTEX_COMPILER bibtex)
LATEX_NEEDIT(MAKEINDEX_COMPILER makeindex)
LATEX_WANTIT(DVIPS_CONVERTER dvips)
LATEX_WANTIT(PS2PDF_CONVERTER ps2pdf)
LATEX_WANTIT(LATEX2HTML_CONVERTER latex2html)

SET(LATEX_COMPILER_FLAGS "-interaction=nonstopmode"
  CACHE STRING "Flags passed to latex.")
SET(PDFLATEX_COMPILER_FLAGS ${LATEX_COMPILER_FLAGS}
  CACHE STRING "Flags passed to pdflatex.")
SET(BIBTEX_COMPILER_FLAGS ""
  CACHE STRING "Flags passed to bibtex.")
SET(MAKEINDEX_COMPILER_FLAGS ""
  CACHE STRING "Flags passed to makeindex.")
SET(MAKEGLOSSARIES_COMPILER_FLAGS ""
  CACHE STRING "Flags passed to makeglossaries.")
SET(DVIPS_CONVERTER_FLAGS "-Ppdf -G0 -t letter"
  CACHE STRING "Flags passed to dvips.")
SET(PS2PDF_CONVERTER_FLAGS "-dMaxSubsetPct=100 -dCompatibilityLevel=1.3 -dSubsetFonts=true -dEmbedAllFonts=tru
  CACHE STRING "Flags passed to ps2pdf.")
SET(LATEX2HTML_CONVERTER_FLAGS ""
  CACHE STRING "Flags passed to latex2html.")
MARK_AS_ADVANCED(
  LATEX_COMPILER_FLAGS
  PDFLATEX_COMPILER_FLAGS
  BIBTEX_COMPILER_FLAGS
  MAKEINDEX_COMPILER_FLAGS
  MAKEGLOSSARIES_COMPILER_FLAGS
  DVIPS_CONVERTER_FLAGS
  PS2PDF_CONVERTER_FLAGS
  LATEX2HTML_CONVERTER_FLAGS
  )
SEPARATE_ARGUMENTS(LATEX_COMPILER_FLAGS)
SEPARATE_ARGUMENTS(PDFLATEX_COMPILER_FLAGS)
SEPARATE_ARGUMENTS(BIBTEX_COMPILER_FLAGS)
SEPARATE_ARGUMENTS(MAKEINDEX_COMPILER_FLAGS)
SEPARATE_ARGUMENTS(MAKEGLOSSARIES_COMPILER_FLAGS)
SEPARATE_ARGUMENTS(DVIPS_CONVERTER_FLAGS)
SEPARATE_ARGUMENTS(PS2PDF_CONVERTER_FLAGS)
SEPARATE_ARGUMENTS(LATEX2HTML_CONVERTER_FLAGS)

FIND_PROGRAM(IMAGEMAGICK_CONVERT convert
  DOC "The convert program that comes with ImageMagick (available at http://www.imagemagick.org)."
  )
IF (NOT IMAGEMAGICK_CONVERT)
  MESSAGE(SEND_ERROR "Could not find convert program.  Please download ImageMagick from http://www.imagemagick
```

```
  ENDIF (NOT IMAGEMAGICK_CONVERT)

  OPTION(LATEX_SMALL_IMAGES
    "If on, the raster images will be converted to 1/6 the original size.  This is because papers usually requir
    OFF)
  IF (LATEX_SMALL_IMAGES)
    SET(LATEX_RASTER_SCALE 16)
    SET(LATEX_OPPOSITE_RASTER_SCALE 100)
  ELSE (LATEX_SMALL_IMAGES)
    SET(LATEX_RASTER_SCALE 100)
    SET(LATEX_OPPOSITE_RASTER_SCALE 16)
  ENDIF (LATEX_SMALL_IMAGES)

  # Just holds extensions for known image types.  They should all be lower case.
  SET(LATEX_DVI_VECTOR_IMAGE_EXTENSIONS .eps)
  SET(LATEX_DVI_RASTER_IMAGE_EXTENSIONS)
  SET(LATEX_DVI_IMAGE_EXTENSIONS
    ${LATEX_DVI_VECTOR_IMAGE_EXTENSIONS} ${LATEX_DVI_RASTER_IMAGE_EXTENSIONS})
  SET(LATEX_PDF_VECTOR_IMAGE_EXTENSIONS .pdf)
  SET(LATEX_PDF_RASTER_IMAGE_EXTENSIONS .png .jpeg .jpg)
  SET(LATEX_PDF_IMAGE_EXTENSIONS
    ${LATEX_PDF_VECTOR_IMAGE_EXTENSIONS} ${LATEX_PDF_RASTER_IMAGE_EXTENSIONS})
  SET(LATEX_IMAGE_EXTENSIONS
    ${LATEX_DVI_IMAGE_EXTENSIONS} ${LATEX_PDF_IMAGE_EXTENSIONS})
ENDMACRO(LATEX_SETUP_VARIABLES)

MACRO(LATEX_GET_OUTPUT_PATH var)
  SET(${var})
  IF (LATEX_OUTPUT_PATH)
    IF ("${LATEX_OUTPUT_PATH}" STREQUAL "${CMAKE_CURRENT_SOURCE_DIR}")
      MESSAGE(SEND_ERROR "You cannot set LATEX_OUTPUT_PATH to the same directory that contains LaTeX input files
    ELSE ("${LATEX_OUTPUT_PATH}" STREQUAL "${CMAKE_CURRENT_SOURCE_DIR}")
      SET(${var} "${LATEX_OUTPUT_PATH}")
    ENDIF ("${LATEX_OUTPUT_PATH}" STREQUAL "${CMAKE_CURRENT_SOURCE_DIR}")
  ELSE (LATEX_OUTPUT_PATH)
    IF ("${CMAKE_CURRENT_BINARY_DIR}" STREQUAL "${CMAKE_CURRENT_SOURCE_DIR}")
      MESSAGE(SEND_ERROR "LaTeX files must be built out of source or you must set LATEX_OUTPUT_PATH.")
    ELSE ("${CMAKE_CURRENT_BINARY_DIR}" STREQUAL "${CMAKE_CURRENT_SOURCE_DIR}")
      SET(${var} "${CMAKE_CURRENT_BINARY_DIR}")
    ENDIF ("${CMAKE_CURRENT_BINARY_DIR}" STREQUAL "${CMAKE_CURRENT_SOURCE_DIR}")
  ENDIF (LATEX_OUTPUT_PATH)
ENDMACRO(LATEX_GET_OUTPUT_PATH)

# Makes custom commands to convert a file to a particular type.
MACRO(LATEX_CONVERT_IMAGE output_files input_file output_extension convert_flags
    output_extensions other_files)
  SET(input_dir ${CMAKE_CURRENT_SOURCE_DIR})
  LATEX_GET_OUTPUT_PATH(output_dir)

  GET_FILENAME_COMPONENT(extension "${input_file}" EXT)

  STRING(REGEX REPLACE "\\.[^.]*\$" ${output_extension} output_file
    "${input_file}")

  LATEX_LIST_CONTAINS(is_type ${extension} ${output_extensions})
  IF (is_type)
    IF (convert_flags)
```

```
      ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${output_file}
        COMMAND ${IMAGEMAGICK_CONVERT}
        ARGS ${input_dir}/${input_file} ${convert_flags}
          ${output_dir}/${output_file}
        DEPENDS ${input_dir}/${input_file}
        )
      SET(${output_files} ${${output_files}} ${output_dir}/${output_file})
    ELSE (convert_flags)
      # As a shortcut, we can just copy the file.
      ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${input_file}
        COMMAND ${CMAKE_COMMAND}
        ARGS -E copy ${input_dir}/${input_file} ${output_dir}/${input_file}
        DEPENDS ${input_dir}/${input_file}
        )
      SET(${output_files} ${${output_files}} ${output_dir}/${input_file})
    ENDIF (convert_flags)
  ELSE (is_type)
    SET(do_convert TRUE)
    # Check to see if there is another input file of the appropriate type.
    FOREACH(valid_extension ${output_extensions})
      STRING(REGEX REPLACE "\\.[^.]*\$" ${output_extension} try_file
        "${input_file}")
      LATEX_LIST_CONTAINS(has_native_file "${try_file}" ${other_files})
      IF (has_native_file)
        SET(do_convert FALSE)
      ENDIF (has_native_file)
    ENDFOREACH(valid_extension)

    # If we still need to convert, do it.
    IF (do_convert)
      ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${output_file}
        COMMAND ${IMAGEMAGICK_CONVERT}
        ARGS ${input_dir}/${input_file} ${convert_flags}
          ${output_dir}/${output_file}
        DEPENDS ${input_dir}/${input_file}
        )
      SET(${output_files} ${${output_files}} ${output_dir}/${output_file})
    ENDIF (do_convert)
  ENDIF (is_type)
ENDMACRO(LATEX_CONVERT_IMAGE)

# Adds custom commands to process the given files for dvi and pdf builds.
# Adds the output files to the given variables (does not replace).
MACRO(LATEX_PROCESS_IMAGES dvi_outputs pdf_outputs)
  LATEX_GET_OUTPUT_PATH(output_dir)
  FOREACH(file ${ARGN})
    IF (EXISTS "${CMAKE_CURRENT_SOURCE_DIR}/${file}")
      GET_FILENAME_COMPONENT(extension "${file}" EXT)
      SET(convert_flags)

      # Check to see if we need to downsample the image.
      LATEX_LIST_CONTAINS(is_raster extension
        ${LATEX_DVI_RASTER_IMAGE_EXTENSIONS}
        ${LATEX_PDF_RASTER_IMAGE_EXTENSIONS})
      IF (LATEX_SMALL_IMAGES)
        IF (is_raster)
          SET(convert_flags -resize ${LATEX_RASTER_SCALE}%)
```

```
        ENDIF (is_raster)
      ENDIF (LATEX_SMALL_IMAGES)

      # Make sure the output directory exists.
      GET_FILENAME_COMPONENT(path "${output_dir}/${file}" PATH)
      MAKE_DIRECTORY("${path}")

      # Do conversions for dvi.
      LATEX_CONVERT_IMAGE(${dvi_outputs} "${file}" .eps "${convert_flags}"
        "${LATEX_DVI_IMAGE_EXTENSIONS}" "${ARGN}")

      # Do conversions for pdf.
      IF (is_raster)
        LATEX_CONVERT_IMAGE(${pdf_outputs} "${file}" .png "${convert_flags}"
          "${LATEX_PDF_IMAGE_EXTENSIONS}" "${ARGN}")
      ELSE (is_raster)
        LATEX_CONVERT_IMAGE(${pdf_outputs} "${file}" .pdf "${convert_flags}"
          "${LATEX_PDF_IMAGE_EXTENSIONS}" "${ARGN}")
      ENDIF (is_raster)
    ELSE (EXISTS "${CMAKE_CURRENT_SOURCE_DIR}/${file}")
      MESSAGE("Could not find file \"${CMAKE_CURRENT_SOURCE_DIR}/${file}\"")
    ENDIF (EXISTS "${CMAKE_CURRENT_SOURCE_DIR}/${file}")
  ENDFOREACH(file)
ENDMACRO(LATEX_PROCESS_IMAGES)

MACRO(ADD_LATEX_IMAGES)
  MESSAGE("The ADD_LATEX_IMAGES macro is deprecated.  Image directories are specified with LATEX_ADD_DOCUMENT.")
ENDMACRO(ADD_LATEX_IMAGES)

MACRO(LATEX_COPY_GLOBBED_FILES pattern dest)
  FILE(GLOB file_list ${pattern})
  FOREACH(in_file ${file_list})
    GET_FILENAME_COMPONENT(out_file ${in_file} NAME)
    CONFIGURE_FILE(${in_file} ${dest}/${out_file} COPYONLY)
  ENDFOREACH(in_file)
ENDMACRO(LATEX_COPY_GLOBBED_FILES)

MACRO(LATEX_COPY_INPUT_FILE file)
  LATEX_GET_OUTPUT_PATH(output_dir)

  IF (EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/${file})
    GET_FILENAME_COMPONENT(path ${file} PATH)
    FILE(MAKE_DIRECTORY ${output_dir}/${path})

    LATEX_LIST_CONTAINS(use_config ${file} ${LATEX_CONFIGURE})
    IF (use_config)
      CONFIGURE_FILE(${CMAKE_CURRENT_SOURCE_DIR}/${file}
        ${output_dir}/${file}
        @ONLY
        )
      ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${file}
        COMMAND ${CMAKE_COMMAND}
        ARGS ${CMAKE_BINARY_DIR}
        DEPENDS ${CMAKE_CURRENT_SOURCE_DIR}/${file}
        )
    ELSE (use_config)
      ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${file}
```

```
        COMMAND ${CMAKE_COMMAND}
        ARGS -E copy ${CMAKE_CURRENT_SOURCE_DIR}/${file} ${output_dir}/${file}
        DEPENDS ${CMAKE_CURRENT_SOURCE_DIR}/${file}
        )
    ENDIF (use_config)
  ELSE (EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/${file})
    IF (EXISTS ${output_dir}/${file})
      # Special case: output exists but input does not.  Assume that it was
      # created elsewhere and skip the input file copy.
    ELSE (EXISTS ${output_dir}/${file})
      MESSAGE("Could not find input file ${CMAKE_CURRENT_SOURCE_DIR}/${file}")
    ENDIF (EXISTS ${output_dir}/${file})
  ENDIF (EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/${file})
ENDMACRO(LATEX_COPY_INPUT_FILE)

#############################################################################
# Commands provided by the UseLATEX.cmake "package"
#############################################################################

MACRO(LATEX_USAGE command message)
  MESSAGE(SEND_ERROR
    "${message}\nUsage: ${command}(<tex_file>\n           [BIBFILES <bib_file> <bib_file> ...]\n           [INPU
    )
ENDMACRO(LATEX_USAGE command message)

# Parses arguments to ADD_LATEX_DOCUMENT and ADD_LATEX_TARGETS and sets the
# variables LATEX_TARGET, LATEX_IMAGE_DIR, LATEX_BIBFILES, LATEX_DEPENDS, and
# LATEX_INPUTS.
MACRO(PARSE_ADD_LATEX_ARGUMENTS command)
  LATEX_PARSE_ARGUMENTS(
    LATEX
    "BIBFILES;INPUTS;IMAGE_DIRS;IMAGES;CONFIGURE;DEPENDS"
    "USE_INDEX;USE_GLOSSARY;USE_GLOSSARIES;DEFAULT_PDF;MANGLE_TARGET_NAMES"
    ${ARGN}
    )

  # The first argument is the target latex file.
  IF (LATEX_DEFAULT_ARGS)
    LATEX_CAR(LATEX_MAIN_INPUT ${LATEX_DEFAULT_ARGS})
    LATEX_CDR(LATEX_DEFAULT_ARGS ${LATEX_DEFAULT_ARGS})
    GET_FILENAME_COMPONENT(LATEX_TARGET ${LATEX_MAIN_INPUT} NAME_WE)
  ELSE (LATEX_DEFAULT_ARGS)
    LATEX_USAGE(${command} "No tex file target given to ${command}.")
  ENDIF (LATEX_DEFAULT_ARGS)

  IF (LATEX_DEFAULT_ARGS)
    LATEX_USAGE(${command} "Invalid or depricated arguments: ${LATEX_DEFAULT_ARGS}")
  ENDIF (LATEX_DEFAULT_ARGS)

  # Backward compatibility between 1.6.0 and 1.6.1.
  IF (LATEX_USE_GLOSSARIES)
    SET(LATEX_USE_GLOSSARY TRUE)
  ENDIF (LATEX_USE_GLOSSARIES)
ENDMACRO(PARSE_ADD_LATEX_ARGUMENTS)

MACRO(ADD_LATEX_TARGETS)
  LATEX_GET_OUTPUT_PATH(output_dir)
```

```
PARSE_ADD_LATEX_ARGUMENTS(ADD_LATEX_TARGETS ${ARGV})

# Set up target names.
IF (LATEX_MANGLE_TARGET_NAMES)
  SET(dvi_target      ${LATEX_TARGET}_dvi)
  SET(pdf_target      ${LATEX_TARGET}_pdf)
  SET(ps_target       ${LATEX_TARGET}_ps)
  SET(safepdf_target  ${LATEX_TARGET}_safepdf)
  SET(html_target     ${LATEX_TARGET}_html)
  SET(auxclean_target ${LATEX_TARGET}_auxclean)
ELSE (LATEX_MANGLE_TARGET_NAMES)
  SET(dvi_target      dvi)
  SET(pdf_target      pdf)
  SET(ps_target       ps)
  SET(safepdf_target  safepdf)
  SET(html_target     html)
  SET(auxclean_target auxclean)
ENDIF (LATEX_MANGLE_TARGET_NAMES)

# For each directory in LATEX_IMAGE_DIRS, glob all the image files and
# place them in LATEX_IMAGES.
FOREACH(dir ${LATEX_IMAGE_DIRS})
  FOREACH(extension ${LATEX_IMAGE_EXTENSIONS})
    FILE(GLOB files ${CMAKE_CURRENT_SOURCE_DIR}/${dir}/*${extension})
    FOREACH(file ${files})
      GET_FILENAME_COMPONENT(filename ${file} NAME)
      SET(LATEX_IMAGES ${LATEX_IMAGES} ${dir}/${filename})
    ENDFOREACH(file)
  ENDFOREACH(extension)
ENDFOREACH(dir)

SET(dvi_images)
SET(pdf_images)
LATEX_PROCESS_IMAGES(dvi_images pdf_images ${LATEX_IMAGES})

SET(make_dvi_command
  ${CMAKE_COMMAND} -E chdir ${output_dir}
  ${LATEX_COMPILER} ${LATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT})
SET(make_pdf_command
  ${CMAKE_COMMAND} -E chdir ${output_dir}
  ${PDFLATEX_COMPILER} ${PDFLATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT})

SET(make_dvi_depends ${LATEX_DEPENDS} ${dvi_images})
SET(make_pdf_depends ${LATEX_DEPENDS} ${pdf_images})
FOREACH(input ${LATEX_MAIN_INPUT} ${LATEX_INPUTS})
  SET(make_dvi_depends ${make_dvi_depends} ${output_dir}/${input})
  SET(make_pdf_depends ${make_pdf_depends} ${output_dir}/${input})
ENDFOREACH(input)

IF (LATEX_BIBFILES)
  SET(make_dvi_command ${make_dvi_command}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${BIBTEX_COMPILER} ${BIBTEX_COMPILER_FLAGS} ${LATEX_TARGET})
  SET(make_pdf_command ${make_pdf_command}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${BIBTEX_COMPILER} ${BIBTEX_COMPILER_FLAGS} ${LATEX_TARGET})
  FOREACH (bibfile ${LATEX_BIBFILES})
```

```
    SET(make_dvi_depends ${make_dvi_depends} ${output_dir}/${bibfile})
    SET(make_pdf_depends ${make_pdf_depends} ${output_dir}/${bibfile})
  ENDFOREACH (bibfile ${LATEX_BIBFILES})
ENDIF (LATEX_BIBFILES)

IF (LATEX_USE_INDEX)
  SET(make_dvi_command ${make_dvi_command}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${LATEX_COMPILER} ${LATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${MAKEINDEX_COMPILER} ${MAKEINDEX_COMPILER_FLAGS} ${LATEX_TARGET}.idx)
  SET(make_pdf_command ${make_pdf_command}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${PDFLATEX_COMPILER} ${PDFLATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${MAKEINDEX_COMPILER} ${MAKEINDEX_COMPILER_FLAGS} ${LATEX_TARGET}.idx)
ENDIF (LATEX_USE_INDEX)

IF (LATEX_USE_GLOSSARY)
  SET(make_dvi_command ${make_dvi_command}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${LATEX_COMPILER} ${LATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${CMAKE_COMMAND}
    -D LATEX_BUILD_COMMAND=makeglossaries
    -D LATEX_TARGET=${LATEX_TARGET}
    -D MAKEINDEX_COMPILER=${MAKEINDEX_COMPILER}
    -D MAKEGLOSSARIES_COMPILER_FLAGS=${MAKEGLOSSARIES_COMPILER_FLAGS}
    -P ${LATEX_USE_LATEX_LOCATION}
    )
  SET(make_pdf_command ${make_pdf_command}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${PDFLATEX_COMPILER} ${PDFLATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT}
    COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
    ${CMAKE_COMMAND}
    -D LATEX_BUILD_COMMAND=makeglossaries
    -D LATEX_TARGET=${LATEX_TARGET}
    -D MAKEINDEX_COMPILER=${MAKEINDEX_COMPILER}
    -D MAKEGLOSSARIES_COMPILER_FLAGS=${MAKEGLOSSARIES_COMPILER_FLAGS}
    -P ${LATEX_USE_LATEX_LOCATION}
    )
ENDIF (LATEX_USE_GLOSSARY)

SET(make_dvi_command ${make_dvi_command}
  COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
  ${LATEX_COMPILER} ${LATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT}
  COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
  ${LATEX_COMPILER} ${LATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT})
SET(make_pdf_command ${make_pdf_command}
  COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
  ${PDFLATEX_COMPILER} ${PDFLATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT}
  COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
  ${PDFLATEX_COMPILER} ${PDFLATEX_COMPILER_FLAGS} ${LATEX_MAIN_INPUT})

# Add commands and targets for building dvi outputs.
ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${LATEX_TARGET}.dvi
  COMMAND ${make_dvi_command}
```

```
    DEPENDS ${make_dvi_depends}
    )
  IF (LATEX_DEFAULT_PDF)
    ADD_CUSTOM_TARGET(${dvi_target}
      DEPENDS ${output_dir}/${LATEX_TARGET}.dvi)
  ELSE (LATEX_DEFAULT_PDF)
    ADD_CUSTOM_TARGET(${dvi_target} ALL
      DEPENDS ${output_dir}/${LATEX_TARGET}.dvi)
  ENDIF (LATEX_DEFAULT_PDF)

  # Add commands and targets for building pdf outputs (with pdflatex).
  IF (PDFLATEX_COMPILER)
    ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${LATEX_TARGET}.pdf
      COMMAND ${make_pdf_command}
      DEPENDS ${make_pdf_depends}
      )
    IF (LATEX_DEFAULT_PDF)
      ADD_CUSTOM_TARGET(${pdf_target} ALL
        DEPENDS ${output_dir}/${LATEX_TARGET}.pdf)
    ELSE (LATEX_DEFAULT_PDF)
      ADD_CUSTOM_TARGET(${pdf_target}
        DEPENDS ${output_dir}/${LATEX_TARGET}.pdf)
    ENDIF (LATEX_DEFAULT_PDF)
  ENDIF (PDFLATEX_COMPILER)

  IF (DVIPS_CONVERTER)
    ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${LATEX_TARGET}.ps
      COMMAND ${CMAKE_COMMAND} -E chdir ${output_dir}
        ${DVIPS_CONVERTER} ${DVIPS_CONVERTER_FLAGS} -o ${LATEX_TARGET}.ps ${LATEX_TARGET}.dvi
      DEPENDS ${output_dir}/${LATEX_TARGET}.dvi)
    ADD_CUSTOM_TARGET(${ps_target}
      DEPENDS ${output_dir}/${LATEX_TARGET}.ps)
    IF (PS2PDF_CONVERTER)
      # Since both the pdf and safepdf targets have the same output, we
      # cannot properly do the dependencies for both.  When selecting safepdf,
      # simply force a recompile every time.
      ADD_CUSTOM_TARGET(${safepdf_target}
        ${CMAKE_COMMAND} -E chdir ${output_dir}
        ${PS2PDF_CONVERTER} ${PS2PDF_CONVERTER_FLAGS} ${LATEX_TARGET}.ps ${LATEX_TARGET}.pdf
        )
      ADD_DEPENDENCIES(${safepdf_target} ${ps_target})
    ENDIF (PS2PDF_CONVERTER)
  ENDIF (DVIPS_CONVERTER)

  IF (LATEX2HTML_CONVERTER)
    ADD_CUSTOM_TARGET(${html_target}
      ${CMAKE_COMMAND} -E chdir ${output_dir}
      ${LATEX2HTML_CONVERTER} ${LATEX2HTML_CONVERTER_FLAGS} ${LATEX_MAIN_INPUT}
      )
    ADD_DEPENDENCIES(${html_target} ${LATEX_MAIN_INPUT} ${LATEX_INPUTS})
  ENDIF (LATEX2HTML_CONVERTER)

  ADD_CUSTOM_TARGET(${auxclean_target}
    ${CMAKE_COMMAND} -E remove ${output_dir}/${LATEX_TARGET}.aux ${output_dir}/${LATEX_TARGET}.idx ${output_dir}
    )
ENDMACRO(ADD_LATEX_TARGETS)
```

```
MACRO(ADD_LATEX_DOCUMENT)
  LATEX_GET_OUTPUT_PATH(output_dir)
  IF (output_dir)
    PARSE_ADD_LATEX_ARGUMENTS(ADD_LATEX_DOCUMENT ${ARGV})

    LATEX_COPY_INPUT_FILE(${LATEX_MAIN_INPUT})

    FOREACH (bib_file ${LATEX_BIBFILES})
      CONFIGURE_FILE(${CMAKE_CURRENT_SOURCE_DIR}/${bib_file}
        ${output_dir}/${bib_file}
        COPYONLY)
      ADD_CUSTOM_COMMAND(OUTPUT ${output_dir}/${bib_file}
        COMMAND ${CMAKE_COMMAND}
        ARGS -E copy ${CMAKE_CURRENT_SOURCE_DIR}/${bib_file} ${output_dir}/${bib_file}
        DEPENDS ${CMAKE_CURRENT_SOURCE_DIR}/${bib_file}
        )
    ENDFOREACH (bib_file)

    FOREACH (input ${LATEX_INPUTS})
      LATEX_COPY_INPUT_FILE(${input})
    ENDFOREACH(input)

    LATEX_COPY_GLOBBED_FILES(${CMAKE_CURRENT_SOURCE_DIR}/*.cls ${output_dir})
    LATEX_COPY_GLOBBED_FILES(${CMAKE_CURRENT_SOURCE_DIR}/*.bst ${output_dir})
    LATEX_COPY_GLOBBED_FILES(${CMAKE_CURRENT_SOURCE_DIR}/*.clo ${output_dir})
    LATEX_COPY_GLOBBED_FILES(${CMAKE_CURRENT_SOURCE_DIR}/*.sty ${output_dir})

    ADD_LATEX_TARGETS(${ARGV})
  ENDIF (output_dir)
ENDMACRO(ADD_LATEX_DOCUMENT)

#############################################################################
# Actually do stuff
#############################################################################

IF (LATEX_BUILD_COMMAND)
  SET(command_handled)

  IF ("${LATEX_BUILD_COMMAND}" STREQUAL makeglossaries)
    LATEX_MAKEGLOSSARIES()
    SET(command_handled TRUE)
  ENDIF ("${LATEX_BUILD_COMMAND}" STREQUAL makeglossaries)

  IF (NOT command_handled)
    MESSAGE(SEND_ERROR "Unknown command: ${LATEX_BUILD_COMMAND}")
  ENDIF (NOT command_handled)

ELSE (LATEX_BUILD_COMMAND)
  # Must be part of the actual configure (included from CMakeLists.txt).
  LATEX_SETUP_VARIABLES()
ENDIF (LATEX_BUILD_COMMAND)
```