UseLATEX.cmake: LaTeX Document Building Made Easy

Kenneth Moreland

Version 1.8.0

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.

clean To CMake's default clean target, the numerous files that LaTeX often generates are added.

auxclean Deletes the auxiliary files from LaTeX, but not the generated input files. Sometimes LaTeX gets itself in a bad state where the auxiliary files need to be deleted to successfully build again, and this target does that without also deleting other build files (such as converted image files or files from unrelated targets in the same directory).

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. For more details and hints on workarounds, see the "Why does UseLATEX.cmake have to copy my tex files?" frequently asked question in Section 4.3.

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.

Note that in order to ensure that the resulting image files are placed in the appropriate directory, you are required to give *relative* paths for images and image directories. For example, `IMAGE_DIRS ${CMAKE_CURRENT_SOURCE_DIR}/images` will fail. Use `IMAGE_DIRS images` instead.

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)
```

If you still want to use the latex program to compile your documents but by default want to create pdf files (that is, build the safepdf target by default), then use the `DEFAULT_SAFEPDF` option to `ADD_LATEX_DOCUMENT`:

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

3.4 SyncTeX-Enabled Editors

Some implementations of LaTeX compilers have a feature called SyncTeX that allows an editor or viewer to link between the compiled version of the document (such as a pdf) and the original LaTeX source code. The most common way to do this is to add -synctex=1 to the pdflatex command. This will create a file named ⟨*base-name*⟩.synctex.gz where each part of the final document points to the original LaTeX files.

However, there is a problem. UseLATEX.cmake copies all of the input LaTeX source files to an out-of-source build directory (see Section 4.3 for more information on why). But the LaTeX compiler does not know that. Thus, the created ⟨*base-name*⟩.synctex.gz will point to the temporary files in the build directory rather than your original source files.

UseLATEX.cmake can add commands to the make targets that "correct" the ⟨*base-name*⟩.synctex.gz. To add these commands, simply turn on the `LATEX_USE_SYNCTEX` in ccmake or equivalent CMake configuring tool. When this option is on, the -synctex=1 argument is added to the LaTeX compile commands (settable with the `LATEX_SYNCTEX_FLAGS` variable) and a command is added to targets that will find files in ⟨*base-name*⟩.synctex.gz and change their paths to the original files in the source directory.

3.5 Building Multiple LaTeX Documents

The most common 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`

```
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)
```

3.6 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.

[1] CMake version 2.4 as of this writing.

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

3.7 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 or xindy 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. It does it in a way similar to the makeglossary command, but in a more portable way. 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)
```

3.8 Nomenclature Support

The nomencl package provides a mechanism to collect nomenclature and print it together in a single section. The nomencl behaves very similarly to glossaries (described in Section 3.7) including running the makeindex command. However, the arguments to makeindex are a bit different (to avoid clashes with creating glossaries), and unfortunately nomencl provides no hints in the auxiliary file about it. Thus, UseLATEX.cmake provides a special `USE_NOMENCL` option to `ADD_LATEX_DOCUMENT` to add the necessary commands to build the nomenclature.

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

It should be noted that this feature only works with nomencl version 4.0 or later. The nomencl package changed how makeindex is called to make it compatible with indices and glossaries. The correct version of nomencl is easily identified as the one that uses the `\makenomenclature` and `\printnomenclature` commands (as opposed to the old `\makeglossary` and `\printglossary` commands). If you are using an older version of nomencl, you are best off to update for a number of reasons.

3.9 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.

3.10 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.

3.11 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

[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.

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.
```

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 How do I automate plot generation with command line programs?

LaTeX is often used in conjunction with plotting programs that run on the command line like gri or gnuplot. Although there is no direct support for these programs in UseLATEX.cmake, it is straightforward to use these programs.

One way to use a plotting program is simply to run it yourself to generate the plot and then incorporate the image file into your LaTeX document as you would any other image file (see Section 3.2). This the easiest way to incorporate these plots since it does not require additional CMake code. It also ensures consistency of how the plot looks (often the plots will look different if created on different platforms), and it provides the opportunity to edit the image to make it look better for publication.

Another way to use these plotting programs is to automatically run them when building the LaTeX document. This is convenient if you frequently change the data you are plotting or if you are creating many plots. To automate running the plotting program build one or more targets to generate these files and then add these targets as LaTeX dependencies (see Section 3.11 for information on adding dependencies). Here is an example of creating the targets for converting a directory of gri files and incorporating the resulting files in a LaTeX document.

```
# Set GRI executable
SET(GRI_COMPILE "/usr/bin/gri")
# Set the location of data files
SET(DATA_DIR data)
# Set the location of the directory for image files
SET(IMAGE_DIR graphics)

# Get a list of gri files
FILE(GLOB_RECURSE GRI_FILES "*.gri")

FOREACH(file ${GRI_FILES})
  GET_FILENAME_COMPONENT(basename "${file}" NAME_WE)
```

```
  # Replace stings in gri file so data files can be found
  FILE(READ
    ${CMAKE_CURRENT_SOURCE_DIR}/${IMAGE_DIR}/${basename}.gri
    file_contents
    )
  STRING(REPLACE "${DATA_DIR}" "${IMAGE_DIR}/${DATA_DIR}"
    changed_file_contents ${file_contents}
    )
  FILE(WRITE
    ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${basename}.gri
    ${changed_file_contents}
    )

  # Command to run gri
  IF(GRI_COMPILE)
    ADD_CUSTOM_COMMAND(
      OUTPUT
        ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${basename}.eps
      DEPENDS
        ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${basename}.gri
        ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${DATA_DIR}
      COMMAND
        ${GRI_COMPILE}
      ARGS
        -output
        ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${basename}.eps
        ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${basename}.gri
      )
  ENDIF(GRI_COMPILE)
  # Make a list of all gri files (for ADD_LATEX_DOCUMENT depend)
  SET(ALL_GRI_FILES ${ALL_GRI_FILES}
    ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${basename}.eps
    )
ENDFOREACH(file)

# Copy over all data files needed to generate gri graphs
ADD_CUSTOM_COMMAND(
  OUTPUT  ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${DATA_DIR}
  DEPENDS ${CMAKE_CURRENT_SOURCE_DIR}/${IMAGE_DIR}/${DATA_DIR}
  COMMAND ${CMAKE_COMMAND} -E copy_directory
          ${CMAKE_CURRENT_SOURCE_DIR}/${IMAGE_DIR}/${DATA_DIR}
          ${CMAKE_CURRENT_BINARY_DIR}/${IMAGE_DIR}/${DATA_DIR}
  )

ADD_LATEX_DOCUMENT(MyDoc.tex
  IMAGE_DIRS ${IMAGE_DIR}
```

```
  DEPENDS ${ALL_GRI_FILES}
  )
```

4.6 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.

4.7 How do I resolve \write 18 errors with pstricks or pdftricks?

A `\write18` command is LaTeX's obtuse syntax for running a command on your system. Commands in the pstricks and pdftricks packages may need to run commands on your system to, for example, convert graphics from one format to another.

Unfortunately, allowing LaTeX to run commands on your system can be considered a security issue. Some versions of LaTeX such as MikTeX disable the feature by default. Thus, in order to use packages that rely on `\write18`, you may have to enable the feature, typically with a command line option. For MikTeX, this command line option is `--enable-write18`.

You can instruct UseLATEX.cmake to pass any flag to LaTeX by adding it to the `LATEX_COMPILER_FLAGS` CMake variable. One way to do this is through the CMake GUI. Simply go to the advanced variables, find `LATEX_COMPILER_FLAGS`, and add `--enable-write18` (or equivalent flag) to the list of arguments.

You can also automatically add this flag by setting the flag in your CMakeLists.txt file. For example:

```
SET(LATEX_COMPILER_FLAGS
  "-interaction=nonstopmode --enable-write18"
  CACHE STRING "Flags passed to latex."
  )
INCLUDE(UseLATEX.cmake)
```

The disadvantage of this latter approach is the reduction of portability. This addition could cause a failure for any LaTeX implementation that does not support the `--enable-write18` flag (for which there are many).

5 Acknowledgments

Thanks to all of the following contributors.

Arnout Boelens Example of using gri in conjunction with LaTeX.

Mark de Wever Fixes for interactions between the makeglossaries and BibTeX commands.

Alin Elena Suggestions on removing dependence on makeglossaries command.

Myles English Support for the nomencl package.

Øystein S. Haaland Support for making glossaries.

Sven Klomp Help with SyncTeX support.

Thimo Langbehn Support for pstricks with the `--enable-write18` option.

Edwin van Leeuwen Fix for a bug when copying BibTeX files.

Lukasz Lis Workaround for problem with ImageMagick dropping the BoundingBox of eps files by using the ps2pdf program instead.

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.

Raymod Wan `DEFAULT_SAFEPDF` option.

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 is released as technical report 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 CMakeLists.txt would not be considered an
# input to the document, but in this special case of documenting
# UseLATEX.cmake the contents of this file is actually included
# in the document.
ADD_LATEX_DOCUMENT(UseLATEX.tex
  INPUTS CMakeLists.txt
  )
```