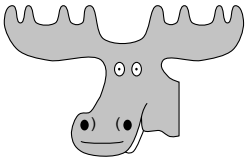
MooseTEX manual



Charles Deledalle

February 25, 2015

MooseTEX is open-source software for UNIX-like systems (such as Linux and MacOS-X) to make the most of your LATEX project by helping you generate high quality documents in PDF format, and if required DVI and PS formats. MooseTEX aims to be used for any kind of project such as articles, letters, reports, theses, presentations or posters.

Contents

1 License

This software is governed by the CeCILL license under French law and abiding by the rules of distribution of free software. You can use, modify and/ or redistribute the software under the terms of the CeCILL license as circulated by CEA, CNRS and INRIA at the following URL "http://www.cecill.info".

As a counterpart to the access to the source code and rights to copy, modify and redistribute granted by the license, users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the successive licensors have only limited liability.

In this respect, the user's attention is drawn to the risks associated with loading, using, modifying and/or developing or reproducing the software by the user in light of its specific status of free software, that may mean that it is complicated to manipulate, and that also therefore means that it is reserved for developers and experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the software's suitability as regards their requirements in conditions enabling the security of their systems and/or data to be ensured and, more generally, to use and operate it in the same conditions as regards security.

The fact that you are presently reading this means that you have had knowledge of the CeCILL license and that you accept its terms.

2 Description of MooseTEX

Based on the technology of Makefile(s), the purpose of MooseTEX is "to determine automatically how (re)generate all pieces of a (large) LaTeX project in the fastest way". For doing so, MooseTEX also includes a suite of conversion tools to generate each of such pieces. More specifically, MooseTEX:

- Provide (or aim at providing) human readable error messages when compilation errors occur,
- Generate references, citations and tables of contents automatically in a single run of MooseTEX,
- Regenerate files only when required by detecting the dependencies with/between
  - TeX files (with extension `.tex`),
  - Styles files (with extension `.sty`),
  - BibTeX files (with extension `.bib`) and
  - images
- Convert images to `.pdf` or `.eps` from the following native formats
  - Blender (with extension `.blend`)
  - Dia (with extension `.dia`)
  - Geogebra (with extension `.ggb`)
  - Gimp (with extension `.xcf`)
  - GLE (with extension `.gle`)
  - Gnuplot (with extension `.gnuplot`)
  - Inkscape (with extension `.svg`)
  - Python (with extension `.py` or `.pickle`)
  - Matlab (with extension `.m` or `.fig`)
  - Octave (with extension `.oct` or `.octave`)
  - Umbrello (with extension `.xmi`)
  - R (with extension `.R`)
  - Scilab (with extension `.sce` or `.scg`)
  - XFig (with extension `.fig`)
  - all formats dealt with LibreOffice
  - all formats dealt with ImageMagick

  while it ensures (or aims at ensuring) that:
  - Fonts are embedded,
  - Boundingbox are correctly defined,

– Vectorial images remain vectorial,

– Quality of bitmap images and encapsulated bitmap images is preserved.

Note that MooseTEX is non-intrusive. It does not change the way you use LATEX and is, as a consequence, compatible with your older projects. You can also use MooseTEX within collaborative LATEX projects without imposing the use of MooseTEX to other collaborators.

The following describes MooseTEX and how to install, configure and use it. This document does not teach LATEX. For documentation about LATEX, the reader can consult [Knuth, 1984, Lamport and Bibby, 1986].

2.1 Prerequisites

In order to work properly, MooseTEX requires that your LATEX project be composed of one or several main `.tex` files starting with the command `\documentclass`. These main `.tex` files should be located in the root directory of your project, denoted by `<PROJECT>`. They may include other `.tex` files located in `<PROJECT>` or in any of its sub-directories. All your source image files should be in a single directory, denoted by `<IMAGESSRCDIR>` (or potentially placed in sub-directories of `<IMAGESSRCDIR>`), which cannot be the same as `<PROJECT>`. The directory denoted by `<IMAGESDSTDIR>`, where generated images will be placed, can be any directory except `<IMAGESSRCDIR>` (in particular, it can be `<PROJECT>`).

2.2 Basics of MooseTEX

When running MooseTEX, with the command `moosetex` it will first analyze all `.tex` and `.sty` files placed in `<PROJECT>` (or its sub-directories) and images placed in `<IMAGESDIR>` (or its sub-directories) such that it will

- Detect unused images,
- Detect the dependencies of `.tex` and `.sty` files to other `.tex` and `.sty` files as well as image files,
- Detect the dependencies of `.svg` files to other image files.

Then, for each `.tex` file starting with the command `\documentclass`, it will generate a `.pdf` file with the same name and in the same directory by generating or updating all required images in advance. There are two ways to generate the `.pdf` file:

1. by asking MooseTEX to use successively the commands `latex`, `dvips`, `ps2pdf`, or
2. by asking MooseTEX to use the command `pdflatex`.

If you choose method 1, images will be generated in `.eps` format and your documents will also be available in `.dvi` and `.ps`. If you choose method 2, images will be generated in `.pdf` format and your documents will be available only in `.pdf`.

All generated images from images in `<IMAGESSRCDIR>` will be placed into `<IMAGESDSTDIR>`. Images placed in sub-directories of `<IMAGESSRCDIR>` will be placed into sub-directories of `<IMAGESDSTDIR>` with the same sub-directory name. For instance if you have a file `<IMAGESSRCDIR>/foo/bar.svg`, MooseTEX will generate `<IMAGESDSTDIR>/foo/bar.xxx` (where either `xxx = pdf` or `xxx = eps`). In this case, your `.tex` file should be of the form

```
...
\usepackage{graphicx}
\graphicspath{{<IMAGESDSTDIR>/},{<IMAGESPREBUILTDIR>/}}
...
\includegraphics{foo/bar}
...
```

Be careful, do not precise extensions for images, do not add subdirectories in your graphicspath, and do not prefix images by `<IMAGESDSTDIR>/`. For instance

```
...
\usepackage{graphicx}
\graphicspath{{<IMAGESDSTDIR>/}}              % Correct
\graphicspath{{<IMAGESDSTDIR>/foo}}           % Incorrect
...
\includegraphics{foo/bar}                     % Correct
\includegraphics{foo/bar.svg}                 % Incorrect
\includegraphics{foo/bar.pdf}                 % Incorrect
\includegraphics{foo/bar.eps}                 % Incorrect
\includegraphics{bar}                         % Incorrect
\includegraphics{<IMAGESDSTDIR>/foo/bar}      % Incorrect
...
```

2.3 Image generation: directives and command line

MooseTEX converts an image to `.pdf` or `.eps` by using its own conversion tool available in command line as `moosetex convert`. You can use this tool independently of a LATEX project to convert specific image formats into `.pdf` or `.eps` formats (see Section 4.2).

MooseTEX automatically ensure that fonts are embedded and the bounding box is the smallest possible. In some particular cases, you may want to give MooseTEX some extra directives for the post-processing of these files. These directives can be given as optional argument of the conversion command line tool or placed in the file `<IMAGESSRCDIR>/directives.conf`. There are directives to control the "resolution", the margins, the generation pipeline... In each line, the file `<IMAGESSRCDIR>/directives.conf` should contain a regular expression, corresponding to the file names for which the directive applies, and the corresponding command line options. For instance, if `<IMAGESSRCDIR>/directives.conf` contains the following

```
foo/bar1                --size=512x512
foo/bar2                --margin=-10,10,-10,10
foo/bar[2-3]            --matlab-print-to=svg
```

the files `<IMAGESSRCDIR>/foo/bar1.yyy`, ..., `<IMAGESSRCDIR>/foo/bar4.yyy` will be automatically generated as `<IMAGESDSTDIR>/foo/bar1.xxx`, ..., `<IMAGESDSTDIR>/foo/bar5.xxx` (where `yyy` is any available format and either `xxx = pdf` or `xxx = eps`) equivalently as if you have manualy typed

```
moosetex convert --size=512x512 \
                           <IMAGESSRCDIR>/foo/bar1.yyy <IMAGESDSTDIR>/foo/bar1.xxx
moosetex convert --margin=-10,10,-10,10 --matlab-print-to=svg \
                           <IMAGESSRCDIR>/foo/bar2.yyy <IMAGESDSTDIR>/foo/bar2.xxx
moosetex convert --matlab-print-to=svg \
                           <IMAGESSRCDIR>/foo/bar3.yyy <IMAGESDSTDIR>/foo/bar3.xxx
moosetex convert <IMAGESSRCDIR>/foo/bar4.yyy <IMAGESDSTDIR>/foo/bar4.xxx
```

In the above example, `<IMAGESDSTDIR>/foo/bar1.xxx` will have a resolution of 512×512 (unit: pt), `<IMAGESDSTDIR>/foo/bar2.xxx` will have margins $(-100, 100, -100, 100)$ (unit: `pt`). Note that the unit can be specified in squared brackets (see Sextion 4.2). For some software such as Gnuplot, Octave, Matlab, R, Scilab, the resolution is set before the image is cropped, but next MooseTEX rescales the image after cropping to fit the required one. If no resolution directive is given, MooseTEX preserves the original one for image formats dealt with ImageMagick otherwise the default size is set to 400×300 pt.

The `--matlab-print-to=svg` directive is dedicated to Matlab. It forces Matlab to first export the image in `.svg` (using `plot2svg`) before generating it into the targeted format. This can be usefull since exports in `.eps` or `.pdf` might sometimes be unsatisfactory with Matlab (but also can sometimes be in `.svg`). In the same vein, you can use the directive `--matlab-print-to=eps[export_fig]` which will force Matlab to first export the image in `.eps` (using `export_fig`). Do not hesitate to play with this option to improve the rendering of your Matlab figures.

Note that changing a directive will force the regeneration of all images whose names match the regular expression.

Note that when using MooseTEX independently of a LATEX project (to convert all such formats to `.pdf` or `.eps` format), you can specify in command line to use a specific directive file (see Section 4.2).

2.4 Fix missing dependencies or make your own dependencies

If you have specific dependencies that MooseTEX does not manage yet, you can specify to MooseTEX the way to take into account these dependencies. To this end, you can add a file `Makefile.deps` in the directory `<PROJECT>/.moosetex` with dependencies described in the form of a standard `Makefile` (see "The GNU Make Manual"). There are two typical examples:

1. for dependencies not detected by MooseTEX ,
2. for formats not dealt by MooseTEX ,

but they are many other situations for which you can use the `Makefile.deps` feature.

Missing dependencies. It might happen that MooseTEX does not detect a dependency. For instance, when defining non-trivial commands as in the following `document.tex` file

```
...
\newcommand{\myfig}[2]{\includegraphics{#1#2}}
...
\myfig{holidays}{2012}
...
```

which refers to an image `<IMAGESDSTDIR>/holidays2012.pdf` or `<IMAGESDSTDIR>/holidays2012.eps` that will not be generated by MooseTEX . MooseTEX indeed does not detect the dependency because it was expected to find the sequence of characters "`holidays2012`" in `document.tex`. Hence, when compiling, the following error will occur

```
Following LaTeX/BibTeX warnings/errors occured
==============================================

./document.tex:36: LaTeX Error: File ‘<IMAGESDSTDIR>/holidays2012.pdf’ not found.
See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...
l.42
I could not locate the file with any of these extensions:
.png,.pdf,.jpg,.mps,.jpeg,.jbig2,.jb2,.PNG,.PDF,.JPG,.JPEG,.JBIG2,.JB2
Try typing  <return>  to proceed.
--
More details in ./document.log
```

Assume that this images should be generated from a `.svg` image `<IMAGESSRCDIR>/holidays2012.svg`, then, with the following `Makefile.deps` file

```
$(TEX_CACHE)/document.tex: <IMAGESDSTDIR>/holidays2012.$(TARGET_FORMAT)
```

MooseTEX will generate the `.pdf` image `<IMAGESDSTDIR>/holidays2012.pdf` or the `.eps` image `<IMAGESDSTDIR>/holidays2012.eps` from `<IMAGESSRCDIR>/holidays2012.svg` using the suitable generation rule. However, at this step, MooseTEX will still believe you are not using this image and will generate the following warning message:

```
Following MooseTeX warnings occured
===================================
```

```
Unused image <IMAGESDSTDIR>/holidays2012.svg
Type 'moosetex vacuum' to move the useless images to the trash directory
```

By placing a file `used` in the directory `<PROJECT>/.moosetex` with the following entry

```
...
<IMAGESDSTDIR>/holidays2012.svg
...
```

it will prevent MooseTEX to show such warning message.

User specific formats. Let say that you are dealing with files of format FOO with file extension `.foo` (that are not dealt by MooseTEX). With the following `Makefile.deps` file

```
<IMAGESDSTDIR>/myimage.$(TARGET_FORMAT): foo/myimmage.foo
        @mkdir -p <IMAGESDSTDIR>
        @foo2$(TARGET_FORMAT) $< $@

$(TEX_CACHE)/document.tex: <IMAGESDSTDIR>/myimage.$(TARGET_FORMAT)
```

MooseTEX will generate the images `<IMAGESDSTDIR>/myimage.pdf` or `<IMAGESDSTDIR>/myimage.eps` from `foo/myimage.foo` using your own program `foo2pdf` or respectively `foo2eps`. It is important that your FOO files are not placed in `<IMAGESSRCDIR>` or its sub-directories that are reserved for images that MooseTEX deals with.

2.5 Vacuuming useless images

MooseTEX auto-detects images that are never used, i.e., placed in `<IMAGESSRCDIR>` and that never appear in LaTEX commands. This often occurs, to have a bunch of figures and images that you were using but you do not need anymore. MooseTEX has a vacuum function that allows you to move all unused images to the "trash" directory `<IMAGESTRASHDIR>`.

2.6 Prebuilt directory

When working collaboratively, some collaborators may not have all pieces of software that you have. For instance, on your system, MooseTEX uses Matlab to convert Matlab figures to `.pdf` or `.eps` formats. Your collaborators may not have Matlab installed, but they still want to generate the document using MooseTEX . Of course, you could share the directory `<IMAGESDSTDIR>` with them but this would be too intensive for your versioning program, and it could break the dependencies. Instead, just share `<IMAGESSRCDIR>` and `<IMAGESPREBUILTDIR>` in which you have placed only figures that they cannot regenerate. When MooseTEX cannot find a program to generate a `.pdf` or `.eps` file, it looks at this last directory to see if this target is present.

2.7 Recommendations

We encourage users of MooseTEX to respect the following convention

Directory	Name
`<IMAGESSRCDIR>`	`srcimages`
`<IMAGESDSTDIR>`	`images`
`<IMAGESTRASHDIR>`	`trashimages`
`<IMAGESPREBUILTDIR>`	`prebuiltimages`

3 Installation of MooseTEX

First of all, get the latest version X.X of MooseTEX. You can either

- Download the archive `moosetex-X.X.tar.gz` from http://www.math.u-bordeaux1.fr/~cdeledal/moosetex.php, decompress it and go to the directory `moosetex-X.X` by typing in a terminal:

```
> tar -zxvf moosetex-X.X.tar.gz
> cd moosetex-X.X
```

- OR, clone the git repository by typing in a terminal:

```
> git clone https://bitbucket.org/charles_deledalle/moosetex.git
> cd moosetex
```

  If you already cloned the repository in the past and you want to update MooseTEX to its latest version, go back to the local repository and update it by typing

```
> cd moosetex
> git pull
```

- OR, if you are a Debian/Ubuntu user, download the package `moosetex_X.X_arch.deb` from http://www.math.u-bordeaux1.fr/~cdeledal/moosetex.php, and install it with GDebi or the Ubuntu Software Center. By doing so, MooseTEX will be automatically installed, your next step is so to go to Section 4.

After checking dependencies as described in Section 3.1, if you are a super user (meaning you can be `root` on your system) go to Section 3.2 otherwise go to Section 3.3

In the following, we are assuming that you are using `bash`. In this case, make sure that the configuration file `.bashrc` is sourced when opening a new shell instance. This is the default behavior, but not every-time (for instance on Mac OS X). In this case, add at the end of the file `$HOME/.bash_profile` the following line:

```
source $HOME/.bashrc
```

3.1 Software dependencies

First check the following software are installed:

command	software	feature	status
bash	GNU FSF	shell interpreter used for homemade commands	R
gcc	GNU FSF.	compile some homemade commands	R
latex	TeX Live	generate `.dvi` file from `.tex` file	R
dvips	TeX Live	generate `.ps` file from `.dvi` file	R
ps2pdf	GhostScript	generate `.pdf` file from `.ps` file	R
pdflatex	TeX Live	generate `.pdf` file from `.tex` file	R
bibtex	TeX Live	generate `.bbl` file form `.aux` and `.bib`	R
mispipe	Moreutils	pipe used for homemade commands	R
pdftops	Poppler	crop and embed fonts	R
pdfcrop	pdfcrop	crop `.pdf` files	R
convert	ImageMagick	convert various image types	R
epstool	epstool	conversion tool used for octave	C
flock	Util-linux	lock program used for multi-threading	C
inkscape ($\geq$0.47)	Inkscape	convert `.svg` files (for geogebra/matlab/scilab)	C
Xvfb	Xvfb	silent GUI conversion	C
–	–	(for geogebra/gle/umbrello/matlab/python/scilab)	C

Legend: (R) required, (C) recommended

Next install the following as needed

command	software	feature	status
blender	Blender	convert `.blend` files	O
dia	Dia	convert `.dia` files	O
geogebra (≥4.4)	Geogebra	convert `.ggb` files (requires Xvfb & Inkscape)	O
gimp	Gimp	convert `.xcf` files	O
gle	GLE	convert `.gle` files	O
gnuplot	Gnuplot	convert `.gnuplot` files	O
octave	Octave	convert `.octave` files (≥3.8 requires epstool)	O
python	Pyhton	generation from figure `plt.gcf()` in .pickle file	O
–	–	generation from figure `plt.gcf()` in .py scripts	O
matlab	Matlab	convert Matlab's `.fig` files	O
–	–	generation from figure `gcf` in .m scripts	O
umbrello	Umbrello	convert `.xmi` files (requires Xvfb)	O
R	R	convert `.R` files	O
scilab (≥5.5)	Scilab	convert `.scg` files (requires Inkscape)	O
–	–	generation from `.sce` scripts (requires Inkscape)	O
fig2dev	XFig	convert XFig's `.fig` files	O
soffice	LibreOffice	convert various document types	O

Legend: (O) optional

The above software have to be located in the directories that appear in your environment variable `PATH`. This is mandatory for required commands. For optional commands, this is mandatory if you want to enable their associated features. Recommended commands are optional commands that we recommend for improved work flow and stability. For instance, if you want to enable the Matlab figure conversion, typing in your shell:

```
> which matlab
```

should give you the path where Matlab is installed (e.g., `/Applications/MATLAB_R2013a.app/bin/matlab`). If it is not the case, edit the file `$HOME/.bashrc` and add at the end of the file something like:

```
export PATH=/Applications/MATLAB_R2013a.app/bin/:$PATH
```

If you are using aliases (if you do not know what is aliases, skip this step), you can instead put the following line in your `$HOME/.bashrc`:

```
export PATH=$(dirname `echo ${BASH_ALIASES[matlab]}`):$PATH
```

Then, save the file, and type in your shell:

```
> source $HOME/.bashrc
```

3.1.1 On Ubuntu (tested on Quantal Quetzal, version 12.10)

In order to install the required dependencies, just type in a terminal:

```
> sudo apt-get install gcc texlive-base texlive-latex-extra texlive-extra-utils
                       poppler-utils imagemagick moreutils
```

To enable an optional feature, for instance conversion of Gimp's `.xcf` files, type something like:

```
> sudo apt-get install gimp
```

Some applications might not be available with `apt-get`. For instance, you will have to install Matlab on your own and make sure it is accessible by command line by updating your environment variable `PATH`.

3.1.2 On MacOS-X

Using XCode, Mactex and Macports (tested on Lion, version 10.7.5): We recommend MacOS-X users to install XCode (available at https://developer.apple.com/xcode/), Macports (available at http://www.macports.org/) and Mactex (available at http://www.tug.org/mactex/). Then, in order to install the required dependencies, just type in a terminal:

```
> sudo port install poppler moreutils imagemagick
```

To enable an optional feature, for instance conversion of Gimp's `.xcf` files, type something like:

```
> sudo port install gimp2
```

Using XCode and Fink (tested on Lion, version 10.7): Another solution consist in installing XCode (available at https://developer.apple.com/xcode/) and Fink (available at http://www.finkproject.org/). Then, in order to install the required dependencies, just type in a terminal:

```
> sudo fink install texlive-base tetex-macosx xpdf moreutils
```

To enable an optional feature, for instance conversion of Gimp's `.xcf` files, type something like:

```
> sudo fink install gimp2
```

Some applications might not be available with neither Macports or Fink. For instance, you may install Blender directly from `www.blender.org` and install Matlab on your own. Make sure they are accessible by command line by updating your environment variable `$PATH`.

If you use Geogebra, we recommend to install an X11 based Java (for instance using SoyLatte http://landonf.bikemonkey.org/static/soylatte), otherwise Geogebra's GUI might show up during the generation (because the native Mac OS X Java uses the Mac display that cannot be used with Xvfb). Next create a file `geogebra` available through your `$PATH` containing something like

```
#!/bin/bash

/some_path_to_soylatte/bin/java -Xms32m -Xmx1024m -jar \
         /Applications/GeoGebra.app/Contents/Java/geogebra.jar "$@"
```

and provide the execution rights as

```
> chmod 755 geogebra
```

Once you have checked your dependencies, you can compile and install MooseTEX in two ways: as a super user or as a non super user.

3.2 Installation for super users

Configure and compile MooseTEX by typing in a terminal:

```
> ./configure
> make
> sudo make install
```

This will install MooseTEX in `/usr/`.

3.3 Installation for non super users

Configure and compile MooseTEX by typing in a terminal:

```
> ./configure --prefix=<PREFIX>
> make
> make install
```

This will install MooseTEX in a directory denoted by `<PREFIX>`.

You will need to update your environment path variable by adding the following line in `$HOME/.bashrc` (create this file if it does not exist already):

```
export PATH=<PREFIX>/bin:$PATH
```

Then, save the file, and type in your shell:

```
> source $HOME/.bashrc
```

If you want to enable bash completions, you might need to add the following line in `$HOME/.bash_completion` (create this file if it does not exist already):

```
. <PREFIX>/share/moosetex/bash_completion.sh
```

Then, save the file, and type in your shell:

```
> source $HOME/.bash_completion
```

4 Generate documents with MooseTEX

4.1 Configuration

If you have never used MooseTEX for your LATEX project, first run MooseTEX from the directory `<PROJECT>` of your project by typing in a terminal:

```
> cd <PROJECT>
> moosetex
```

The following will appear

```
 /\/\__/\_/\      /\_/\__/\/\         Welcome to MooseTeX
 \          \____/          /         author Charles Deledalle
  '____-_____       ____-___'
            | 0 0  \_
            |                         The easy way to use LaTeX
          _/     /
         /o)  (o/l \_                 CeCILL license
         \=====//

Configuration of your current project
=====================================

Configuration of LaTeX generation

Do you want to compile using:
 1. latex -> dvips -> ps2pdf, or
 2. pdflatex.
Please enter your answer (1/2) [2]:
```

Answer a few questions concerning the way you want MooseTEX to compile your LaTeX project. More details about these options are given in Section 2. In brackets are given the default options that will be used if you just press `ENTER` by leaving the answer blank. Choosing default options requires that your project be organized as it is suggested in Section 2.7. Once you have finished answering the questions, you can proceed to the generation of your documents as described in the next section.

Note that if you want to reconfigure your project, you can do it easily by typing

```
> moosetex configure
```

4.2 Utilization

We assume in this section that you have already configured MooseTEX as described in the previous section. MooseTEX works natively in command line. You may nevertheless use MooseTEX from your favorite editors as described in the following subsections.

4.2.1 In a terminal

In order to generate all documents and linked images in the desired format as described in Section 2, go to the directory of your project and type:

```
> moosetex
```

If everything generate well, you might see something like

```
Document generation
===================

Loading...

Generated files           Comments
================================================================================
image.svg                 Start generation
doc.tex                   Start generation
doc.aux                   Generate bibliography
doc.tex                   Bibliography, references or table of contents generation
doc.tex                   Citations generation
doc.tex                   => bibtex 1 run(s)
doc.tex                   => latex 3 run(s)
*                         Compilation succeed
================================================================================
```

MooseTEX has other options that can be listed by typing `moosetex help`:

```
Usage: moosetex [options] [subcommand] [file1.tex ... fileN.tex]

Generate all LaTeX documents placed in the current directory
(or only file1.tex ... fileN.tex if specified).

Available commands:
> moosetex             Run generation.
> moosetex configure   (Re)configure.
> moosetex vacuum      Move the unused images to the trash directory
> moosetex clean       Remove intermediate files.
> moosetex distclean   Remove all generated files.
> moosetex purge       Remove all generated files and configuration files.
> moosetex convert [options] input-file output-file
                       Convert one or several images to eps or pdf format.
```

```
                        For more information type "moosetex convert --help".
> moosetex update       Update list of available software.
> moosetex version      Print the version number and exit.
> moosetex help         Print this message and exit.

Available options:
  --backward|-b         Change before doing anything to the first backward
                        directory being a MooseTex project.
                        Current directory is selected if none are found.
  -C DIRECTORY          Change of directory before doing anything.
  --default|-d          Use default answer whatever the question.
  --force|-f FILE       Force regenerating FILE (if used).
  --nocolor             Suppress color output.

Enjoy!
```

We invite the reader to refer to Section 2 for more details about these options.

In order to convert a specific image to `.eps` or `.pdf` format as described in Section 2, type:

```
> moosetex convert input-image.yyy output-image.xxx
```

where `yyy` is the format of the input image and either `xxx = pdf` or `xxx = eps`. You can also convert all files of a directory in either `xxx = pdf` or `xxx = eps` to a destination directory by typing

```
> ls input_directory/* | moosetex convert --batch-input-to=xxx output_directory/
```

The `convert` command has other options that can be listed by typing `moosetex convert --help`:

```
Usage: moosetex convert [options] input-file output-file
       moosetex convert [options] --batch-input-to=FORMAT output-dir

Convert one or several images to eps or pdf format.

Available options:
  --batch-input-to=FORMAT    Convert each file given on standard input to
                             FORMAT=eps|pdf and place the result in output-dir.
  --matlab-print-to=FORMAT   Specify to convert via
                             FORMAT=eps|pdf|svg|eps[export_fig]|pdf[export_fig]
                             using Matlab's print function for eps and pdf,
                             Juerg Schwizer's plot2svg function for svg and
                             Oliver Woodford's export_fig function for
                             eps/pdf[export_fig] (default eps).
  --margin=L,T,R,B           Specify targeted margins (default 0,0,0,0).
  --prebuiltdir=DIR          Look for pre-built output in DIR in case of failure.
  --size=WxH                 Specify targeted size.
  --use-directives=FILE      Provide a MooseTeX's image directive file.
  --help                     Print this message and exit

Examples:
  moosetex convert example.dia example.eps
  ls *.xcf | moosetex convert --batch-input-to=pdf results/

Enjoy!
```

We invite the reader to refer to Section 2 for more details about these options.

4.2.2 From GNU/Emacs (tested on versions 23.3.1 and 23.4.1)

Press `alt` or `escape` followed by `x`, type `compile` and validate by pressing `enter` (in short, `M-x compile`). Insert next to the following command:

```
Compile command: moosetex
```

You can also use MooseTEX options using the command lines described in the previous section. If you want to configure MooseTEX from GNU/Emacs, you should use `C-u M-x compile` instead to enable the interactive mode.

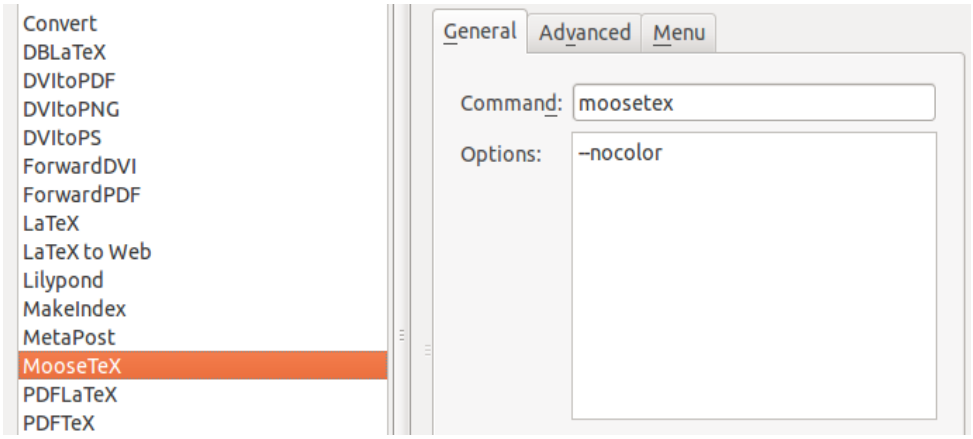
This can be made simpler/automatic if you add to your configuration file `$HOME/.emacs` (typically at the end) the following commands:

```
;; MooseTeX
(add-hook 'latex-mode-hook
  (lambda ()
    (unless (or (file-exists-p "makefile")
                (file-exists-p "Makefile"))
      (set (make-local-variable 'compile-command) "moosetex -b"))))
(add-hook 'LaTeX-mode-hook
  (lambda ()
    (unless (or (file-exists-p "makefile")
                (file-exists-p "Makefile"))
      (set (make-local-variable 'compile-command) "moosetex -b"))))
(defadvice compile (before ad-compile-smart activate)
  (when (or (derived-mode-p 'latex-mode) (derived-mode-p 'LaTeX-mode))
    (ad-set-arg 1 t)))
(defun recompile ()
  (interactive)
  (set (make-local-variable 'compilation-read-command-state)
       'compilation-read-command)
  (set (make-local-variable 'compilation-read-command) nil)
  (call-interactively 'compile)
  (set (make-local-variable 'compilation-read-command)
       'compilation-read-command-state))
(global-set-key [f5] 'compile)
(global-set-key [f6] 'recompile)
(global-set-key [f7] 'next-error)
```

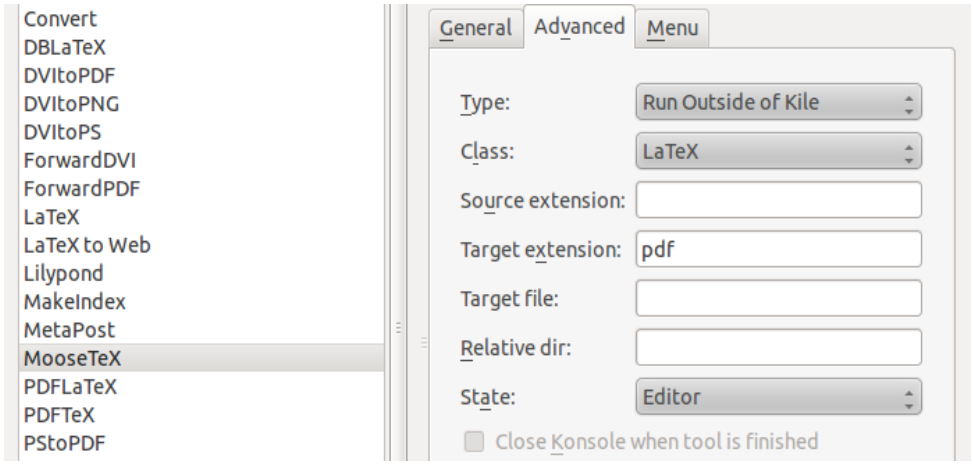
This will directly enter the `moosetex` command when typing `M-x compile`, allow you to configure MooseTEX without typing instead `C-u M-x compile` and provide you three shortcuts to compile, recompile and move to the next error by pressing respectively the keys `[f5]`, `[f6]` and `[f7]`.

4.2.3 From Kile (tested on version 2.1.2)

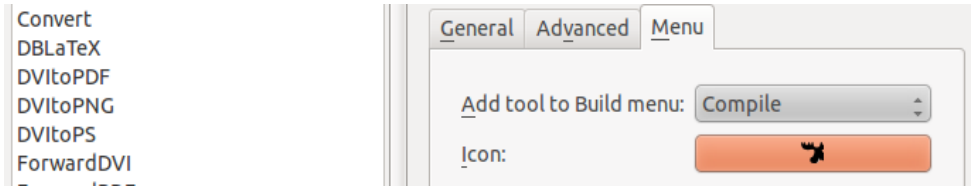
Go to the tab `Settings` and select `'Configure Kile...'`. In the section `'Tools > Build'` create a new tool with a short descriptive name `MooseTeX` and leave the default behavior to `<Custom>`. Next, enter the following:



switch to the tab `'Advanced'`:



and switch to the tab `'Menu'`:

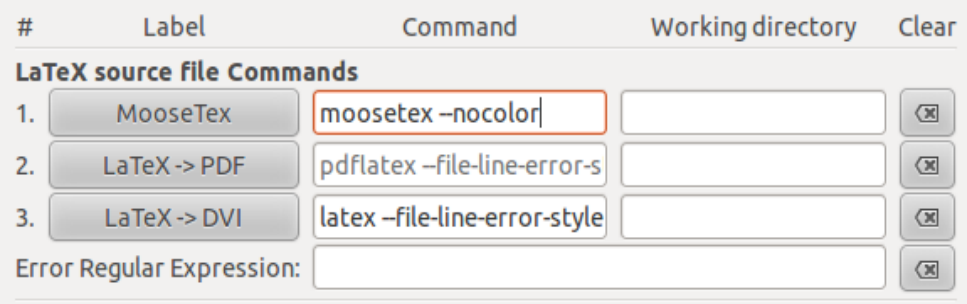


If MooseTEX was installed with default configuration, you should find the icons in `/usr/share/pixmaps/`, otherwise they are placed in `<PREFIX>/share/pixmaps/`. Press next the button `'OK'`.

You can now generate your LATEXdocuments with MooseTEX by selecting or clicking on `MooseTeX` in the `Compile` menu:

4.2.4 From Geany (tested on version 1.22)

Go to the tab `Build` and select `'Set Build Commands'`. In the section `'LaTeX source file Commands'` modify the entries as follows



Press next the button `'Compile the current file'` in the toolbar or press `F8`.

5 Related projects

- autolatex (http://www.arakhne.org/autolatex/)
- latexmk (http://users.phys.psu.edu/~collins/software/latexmk-jcc/)
- pretty print latex (http://www.stefant.org/web/projects/software/pplatex.html)
- rubber (https://launchpad.net/rubber/)

6 Known issues

- There is no guarantee that the pipelines 1 and 2 produce the same `.pdf` file.
- The quality of generated images strongly depends on the versions of the used programs. Please, try as much as possible to update your system.
- Depending on their versions, Scilab, Octave and Umbrello might have some instabilities (with potential crash).
- Could not find an implementation of `flock` for MacOS-X (hence, prevents the use of multi-threading). Apparently `flock` might be available with XCode 5 (but I haven't tried yet)

References

[Knuth, 1984] Knuth, D. (1984). The texbook. *Reading, Mass.*

[Lamport and Bibby, 1986] Lamport, L. and Bibby, D. (1986). *LATEX: A document preparation system*, volume 260. Citeseer.