primal v1.0

Mike Hansen

May, 2016

outline

1. introduction
2. getting started
3. primal projects
4. man pages

primal makes LaTeX projects easier

- convenient compilation with `pdflatex` and `latex` → `dvi` → `pdf`
- automatic compilation of glossaries and BibTeX databases
- clean workspace and robust, repeatable compilation
- straightforward customization - primal is just a few bash scripts
- templates for journal formats, cover letters, paper rebuttals, etc.
- easy integration with popular LaTeX editors

clone from GitHub

From a terminal, enter

```
$ git clone https://github.com/michael-a-hansen/primal.git
```

which will clone the repository into a new directory named `primal`.

Here you can find:

- `doc/` - documentation
- `src/` - source code
- `templates/` - template documents
- `primal-configure.sh` - configure script

configure before running

Whenever you clone, pull, or make changes to your local repository, run the configure script from inside the `primal` directory.

```
$ cd primal
$ ./primal-configure.sh
```

This will build the `configured` directory that contains scripts for primal's main functionality and testing.

testing primal

primal may fail for a number of reasons, such as paths not being set correctly[1] or errors committed when customizing the scripts.

To test primal go to the `configured` directory and run `runtests.sh`.

```
$ cd configured
$ ./runtests.sh
```

This will confirm that each template in primal successfully assembles into a project and builds a pdf.

[1] 'Paths not being set correctly' means that your system can't find executables such as `pdflatex` or LaTeX packages or style files.

assemble a blank project

To make a blank primal project in directory `projname`, simply enter:

```
$ <primal-path>/configured/assemble-project.sh
--name=projname --template=blank
```

where `<primal-path>` is the `primal` directory from earlier.

structure of a project

The assembled project directory `projname` has four directories and a local configuration file:

- `projname/src/` - contains at least the main `tex` file
- `projname/output/` - `pdf`, `dvi`, `ps` output files are copied to this directory after document compilation
- `projname/log/` - the LaTeX log file is stored here after compilation
- `projname/tmp/` - temporaries from LaTeX and subprocesses (*e.g.* glossaries) are moved here after compilation
- `projname/primal-project-config` - see next

project configuration file

The project configuration file, `projname/primal-project-config`, sets the following variables necessary for document compilation:

- `mainname=` the main `tex` file
- `texdir=` path of the `pdflatex` executable
- `primalbasedir=` path of the cloned `primal` directory
- `texer=` the executable (`pdflatex` or `latex`)
- `pdfinsrc=` set to 'y' to keep a copy of the `pdf` in the `src` directory, necessary for editors that sync the `pdf` continously
- `tmpexts=` comma-separated (no spaces) list of extensions[2] belonging to the temporary files moved to `projname/tmp`

[2] The `glossaries` package may be used to create custom glossaries, each of which needs several extensions that should be added here.

primal ships with several templates

Templates have necessary class/style files, packages, and a document with placeholders (*e.g.* title, authors, abstract, keywords). Formats such as SIAM papers that require a `latex` → `dvi` → `pdf` build will automatically set the project configuration accordingly.

To assemble a project with a template below, simply use the `-template=` argument to `assemble-project.sh`.

- `blank` - a blank document
- `article` - simple document of `article` class
- `cover letter` - basic cover letter using the `letter` class
- `paper rebuttal` - document of `article` class with a package for writing responses to reviews
- **journal/conf. formats:** `elsevier-article`, `siam`, `combustion-theory-modelling`, `combustion-meeting`

custom templates

If you find yourself writing the same document over and over again, you can easily make it a new primal template.

1. Make a new folder in the `primal/templates/` directory
2. Add your `tex` files, class/style files, etc.
3. Change the name of the main `tex` document to `main.tex`

primal will automatically find the new template and include it in testing. You should run the tests after making a new template.

If your template requires `latex` → `dvi` → `pdf`, create an empty file named `USE_LATEX` in the new template directory. Projects of this type will then be configured to use `latex` → `dvi` → `pdf` instead of `pdflatex`.

building a pdf in primal project

To build a pdf with primal go to the `src` directory of the project and run the `write-project.sh` script.

If compilation is successful, the `pdf` file will be generated and copied to the output directory while the log and temporary files will be moved accordingly.

If compilation fails, primal will inform you through the console and find the offending line where LaTeX found an error.

```
$ cd projname/src
$ <primal-path>/configured/write-project.sh
```

building a pdf outside of a project

primal uses a standalone script to generate a `pdf` - `generate-pdf.sh`. You can use this script on a `tex` file anywhere on your system, it doesn't have to be part of a primal project.

The first argument is the name of the `tex` file *without extension*, and the second argument (optional - default `pdflatex`) is the executable to build with - `latex` or `pdflatex`. No cleanup is performed with `generate-pdf.sh`.

```
$ <primal-path>/configured/generate-pdf.sh <file> <exec>
```

Making aliases for the assembly & write scripts

Remembering the path to primal for commands like `<primal-path>/configured/write-project.sh` can be annoying.

Aliases to the assembly and writing scripts can be added to your profile automatically during configuration. Call the configure script with the path of your `aliases` file or `profile`:

```
$ ./primal-configure.sh --profile=<path-to-profile>
```

This will add (unless they already exist) two aliases to your profile:

```
alias pa='.../configured/assemble-project.sh'
alias pw='.../configured/write-project.sh'
```

man page: `primal-configure.sh`

This configures primal for your system and populates the `configured` directory with scripts for assembling projects, running tests, and building documents.

Usage:

```
$ ./primal-configure.sh --profile=<profpath>
--texshop=<tdir>
```

- `--profile=` builds aliases in the profile at the given path
- `--texshop=` builds a symbolic link to `primal-write.sh` in TeXShop's `Engines` directory so that primal can be used as a TeXShop build script (restart TeXShop after building this link the first time). Here you supply the path of the Engines directory, typically something like `/Users/<user-name>/Library/TeXShop/Engines`.

man page: `generate-pdf.sh`

This generates the `pdf` and is called by `write-project.sh`. It can be run from anywhere in the The first argument, the name of the `tex` file without extension, must be supplied while the second argument, the executable used to build the document, is optional (default: `pdflatex`).

Usage:

```
$ <primal-path>/configured/generate-pdf.sh <file> <exec>
```

man page: `runtests.sh`

This tests assembly and compilation of each template. It must be run from within the `configured` directory of primal. It will build a new directory, `primal-test-project`, into which each test is assembled. It takes no arguments.

Usage:

```
$ ./runtests.sh
```

man page: `assemble-project.sh`

This assembles a primal project given a name and a template type.

Usage:

```
 $ <primal-path>/configured/assemble-project.sh
--name=<project-name> --template=<template-name>
```

Adding `--help` to the command will display a list of options and available templates. Nothing will be assembled if `--help` is in the command.

man page: `write-project.sh`

This generates the `pdf` and keeps a primal project clean. It must be run from within the `src` directory of a project. It takes no arguments.

Usage:

```
$ <primal-path>/configured/write-project.sh
```