Auxilium Developer's Guide

Version 0.1.6

Contents

Auxilium Developer Introduction

This manual provides information about helping to maintain and/or extend the capabilities of the Auxilium open source project. The information on how to just get and use Auxilium is avalable at https://ericmotleybytes.github.io/auxilium/, which also has links to the latest Auxilium User Guide in both html and pdf formats. Before attempting Auxilium development, or getting too deep into this manual, it is probably wise to look over these end-user resources.

Auxilium Primary GitHub Repository

The original and primary Auxilium GitHub repository site, which is developer oriented, is at https://github.com/ericmotleybytes/auxilium.

Recommended Collaborative Development Process

The GitHub recommended collaborative development process is also supported and recommended for Auxilium. This is sometimes called the "fork and branch" workflow. The basic workflow is as follows.

1. Get a GitHub account if you do not already have one. The free account option is fine.
2. "Fork" the main Auxilium repository at https://github.com/ericmotleybytes/auxilium.
3. "Clone" the forked repository to your local system.
4. Add a Git remote for the original repository.
5. Create a feature branch in which to place your changes.
6. Make your changes to the new branch.
7. Commit the changes to the branch.
8. Push the branch to GitHub.
9. Open a pull request from the new branch to the original repo.
10. Clean up after your pull request is merged.

A good tutorial for this process is available at http://blog.scottlowe.org/2015/01/27/using-fork-branch-git-workflow/. GitHub also has good documentation such as their repository fork description at https://help.github.com/articles/fork-a-repo/.

Tools

All runtime Auxilium programs are written in pure Bash version 4.2 or higher. So to just use Auxilium, Bash is the only absolute prerequisite.

For development several additional tools are required to support development unit testing and documentation generation. These tools include:

- Bats. The Bash Automated Testing System. Avalable at https://github.com/sstephenson/bats.
- Pandoc. A universal document converter. See http://pandoc.org/ and https://github.com/jgm/pandoc.
- LaTeX. Used by Pandoc to generate PDF files. *TeX Live* is recommended. See http://www.tug.org/texlive/.

- Pandoc Preprocessor. The "pp" Pandoc-oriented text preprocessor. See http://cdsoft.fr/pp/ and https://github.com/CDSoft/pp.

A "make" driven process.

Auxilium uses the "make" utility to drive the development build and release process.

You can clone Auxilium to your local server the top level git root directrory is usually just called "auxilium", and in this document we will refer to it this way. Specifying "auxilium/bin" refers to the "bin" subdirectory under the auxilium git root directory. For brevity, sometime the "auxilium" part if left off, so "bin" also refers to the "bin" subdirectory under the auxilium root directory.

The master development Makefile is at "auxilium/Makefile". It includes several other Makefiles. They all work together. All important auxilium development build and release tasks are defined in this Makefile and its inclusions. This makefile has a number of different help screens to help you know what it can do. A master help screen, which explains how to see several additional help screens, can be viewed as follows:

```
$ cd auxilium
$ make help
Available help topics:
  make               # display this screen.
  make help          # display this screen.
  make helpbuild     # help building/cleaning sources.
  make helpinstall   # help install/uninstall.
  make helpgit       # help common git tasks.
  make helprel       # help common GitHub release tasks.
```

Building Auxilium

You can see the make options available to build Auxilium as follows.

```
$ cd auxilium
$ make helpbuild
Common building usage:
  make helpbuild     # display this screen.
  make build         # build docs and more.
  make test          # run tests.
  make checktest     # run tests and remember results.
Common build cleanup usage:
  make clean         # clean tests and docs.
  make cleantest     # copy test log files.
  make cleandocs     # delete working copy of gen'd docs.
```

The process to build auxilium from scratch is simple.

```
$ cd auxilium
$ make clean
...(output not shown)...
$ make build
...(output not shown)...
```

The process to run all the unit tests is also simple.

```
$ cd auxilium
$ make checktest
...(output not shown)...
```

The Makefile is very smart about dependencies, which makes incremental re-building easy. each "make build" only rebuilds what it things is needed. Once in a while, however, it is prudent to do a "make clean", which will force the next "make build" to rebuild everything.

Installing Auxilium

You can see the make options available to install Auxilium as follows.

```
$ cd auxilium
$ make helpinstall
Common installing usage:
  make install       # copy files to under ~/local.
  make sysinstall    # copy files to under /usr/local as root.
  make install PREFIX=<prefix> # custom copy files.
Common uninstalling usage:
  make uninstall     # delete files installed under ~/local.
  make sysuninstall  # delete files installed under /usr/local as root.
  make uninstall PREFIX=<prefix> # custom uninstall files.
```

It is easy to install to your own personal area.

```
$ cd auxilium
$ make install PREFIX=~/local
...(output not shown)...
```

If you want to install to a system area and you are not logged in as root, you might be able to use sudo to get the right access privileges.

```
$ cd auxilium
$ sudo make install PREFIX=/usr/local
...(output not shown)...
```

You can uninstall auxilium in a similar manner with the uninstall option.

```
$ cd auxilium
$ make uninstall PREFIX=~/local
...(output not shown)...
$ sudo make uninstall PREFIX=/usr/local
...(output not shown)...
```

Releasing Auxilium

Generally, as a contributing developer working on a linked and forked child repository, you should not perform any releases nor any of the tasks listed on the "make helprel" screen. Of course, if you decided to release your version of auxilium separately (which is not recommended) these release options could prove helpful.

Platforms

Auxilium programs are all written in Bash, the standard Linux shell language. Bash is an interpretive language, so as long as your platform has a relatively new version of Bash installed, it *should* be able to run the runtime Auxilium Bash scripts. There is no need of different runtime versions of Auxilium for different hardware platforms because there are no hard-compiled programs.

That being said, to date Auxilium has only been tested on Linux platforms. The author, however, would be very interested in user experience reports.

Authors

Eric Alan Christiansen
eric@motleybytes.com
https://www.linkedin.com/in/ericchristiansen/

Legal

Copyright

Licensing

See Also

- https://ericmotleybytes.github.io/auxilium/ : The Auxilium Information and Downloads site.
- https://github.com/ericmotleybytes/auxilium : The master auxilium GitHub repository. You can also report bugs, describe issues, and request enhancements here.
- http://blog.scottlowe.org/2015/01/27/using-fork-branch-git-workflow/ : A good tutorial on the GitHub "fork and branch" workflow.
- https://help.github.com/articles/fork-a-repo/ : GitHub documentation about how to "fork" a repository.
- https://github.com/sstephenson/bats : Bats, the Bash Automated Testing System.
- http://pandoc.org/ : Pandoc information page.
- https://github.com/jgm/pandoc : Pandoc GitHub page.
- http://www.tug.org/texlive/LaTeX : The 'TeX Live' LaTeX system.
- http://cdsoft.fr/pp/ : "pp", the pandoc preprocessor.
- https://github.com/CDSoft/pp : "pp", the pandoc preprocessor on GitHub.
- https://testanything.org/ : A description of TAP, the unit testing Test Anything Protocol.