Integration of GNU Autotools with Python Development Environments

Franklin E. Diaz

June 03, 2022

Abstract

The goal of this document is to detail my use of GNU Autotools with Python development. My effort to reduce the complexity of delivering Python heavy projects to customers and end users has led to useful ways of working that I share in this paper.

Click here to download latest version.

1 Integration of GNU Autotools with Python Development Environments

My effort to reduce the complexity and ease configuration issues when delivering Python heavy projects to customers and end users has led to useful ways of working that I share in this paper. Autotools are a well-maintained set of Open Source tools with a gentle learning curve and are included in the distribution of many Open Source packages that we all rely on daily, at least indirectly. Here I have assembled certain battle-tested methods that I encourage others to understand and adopt.

The list of tools for using this Autotools configuration paradigm is show in Table 1.

Tool	Description
autoconf	Generates a configure script from configure.ac
automake	Generates a system-specific Makefile based on Makefile.am template
make	X

Table 1: Tools used in this project

In an effort to maintain a tidy development environment, the ".gitignore" file is appended with the following entries. These entries will prevent intermediate build files from being checked in to revision control unnecessarily. Note that in the case of "aclocal" it has been intentionally omitted since there is a sub-folder containing many m4 macros specific to Latex document configuration and build.

1.1: Files to be excluded from revision control systems

```
Makefile
Makefile.in
configure
configure~
config/
config/install-sh
config/missing
#aclocal
aclocal.m4
autom4te.cache/
config.status
config.log
COPYING
INSTALL
libtool
```

The image shown in Figure 1 is from [1]. The image illustrates the relationship between the components mentioned in Table 1.

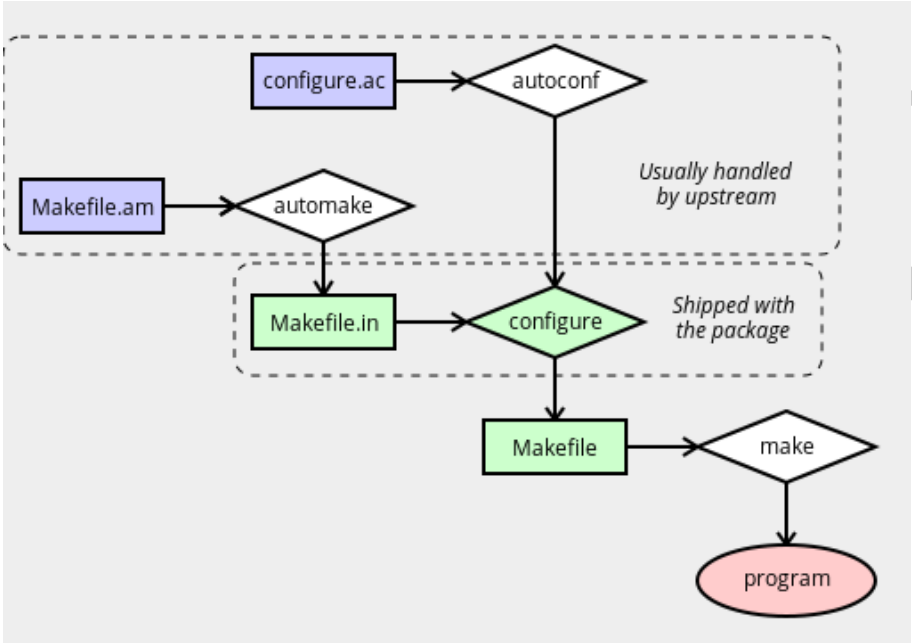

Figure 1: A basic overview of how the main autotools components fit together.

2 The bootstrap.sh script

The purpose of the "bootstrap.sh" script is to allow project maintainers to prepare the local environment for use of autotools. The boostrap script will attempt to guess if the local host is running a certain Linux distribution, or MacOS. There are execution paths for each of these respective scenarios.

At first execution, autotools will run libtool, automake, and configure. Autotools creates a file named "config.log". If the config.log file is found on subsequent runs of the script, fewer configuration steps will be performed since a certain system state is assumed. To get a "clean start" the maintainer can simply delete the config.log file and rerun the bootstrap.sh script.

A full working example of the bootstrap.sh script can be found in the source repository for this paper.

2.1: Steps to use Autotools

```
./bootstrap.sh
./configure
make python
. _build/bin/activate
```

3 The configure.ac file

A "./configure" script can be generated from a template named "configure.ac". This template is comprised of macros that are used to identify local system software and program locations. In our case, we are interested in identifying the location of the Python 3.x installation on the local system.

Per the automake documentation we can define the minimum acceptable version of Python and a variable that automake can use to refer to the version of Python discovered on the system. These declarative statements and the resultant values can then be referenced by automake and added explicitly to the "Makefile.am" template.

3.1: Declaring Python Autoconf macros in configure.ac

```
dnl the Python configuration
AM_PATH_PYTHON(3.9) # minimum version of Python
PY39="python$PYTHON_VERSION" # define the python interpreter
dnl LDFLAGS="$LDFLAGS -l$PY39"
AC_SUBST(PY39, python$PYTHON_VERSION)
```

A full working example of the configure.ac file can be found in the source repository for this paper.

4 The Makefile.am file

The "Makefile.am" file is a template that Automake uses to generate the full Makefile. Note that this project includes "Makefile.am" at the top level for setting up the Python virtual environment, and a second named "paper/Makefile.am" that contains directives on generating the PDF document you are reading now.

4.1: The Makefile.am for generating the top-level Makefile

```
CLOCAL_AMFLAGS = -I config/m4 -I aclocal
ACLOCAL_AMFLAGS = -I config/m4 -I aclocal

.PHONY: src tests

clean: ## Clean up your mess
	rm -rf _build *.egg-info
	@find . -name '*.pyc' | xargs rm -rf
	@find . -name '__pycache__' | xargs rm -rf
	@find . -name 'Makefile' | xargs rm -rf
	@find . -name 'Makefile.in' | xargs rm -rf
	@for trash in _build aclocal.m4 autom4te.cache config config.log config.status configure configure~ libtool; do\
		if [ -f $$trash ] || [ -d $$trash ]; then \
			echo "Removing $$trash" ;\
			rm -rf $$trash ;\
		fi ; \
	done

python:
	@$(PY39) -m venv _build
	( \
		source _build/bin/activate; \
		_build/bin/python -m pip install --upgrade pip; \
		_build/bin/python -m pip install -r src/requirements.txt; \
	)

test:
	@$(PY39) -m venv _build
	( \
		source _build/bin/activate; \
		_build/bin/python -m pip install --upgrade pip; \
		_build/bin/python -m pip install tox; \
		_build/bin/python -m pip install -r tests/requirements-test.txt; \
		_build/bin/python -m pip install -r tests/requirements-security.txt; \
	)
```

4.2: The Makefile.am for generating the paper Makefile

```
CLOCAL_AMFLAGS = -I config/m4 -I aclocal

clean:
	rm -rf _build *.egg-info
	@for trash in *.aux *.bbl *.blg *.lof *.log *.lot \
		*.out *.pdf *.synctex.gz *.toc ; do \
		if [ -f "$$trash" ]; then \
			rm -rf $$trash ; \
		fi ; \
	done

paper:
	latexmk -pdf -file-line-error -interaction=nonstopmode -synctex=1 \
		-shell-escape paper-autotools-python
	bibtex paper-autotools-python
```

5 Python requirements.txt files

My projects typically include three main requirements files, separated by functionality and named to foster a self-documenting paradigm.

Filename	Description
src/requirements.txt	Common Python modules required by the main application.
tests/requirements-test.txt	test, linting, syntax checking, formatting, etc.
tests/requirements-security.txt	scan and secure the main app

Table 2: Python requirements files

Revision History

Revision	Date	Author(s)	Description
v0.1	June 03, 2022	Franklin Diaz	Initial Draft

References

[1] Gentoo Authors. The basics of autotools, 2022.