Homework 9 - Part 1

Nicholas LaRosa

April 10, 2013

Autoconf

Before using Autoconf, Autoscan must be run, generating a configure.scan file. This file characterizes source code in the package being configured. After editing the configure.scan file to correlate with the custom package and renaming the file to configure.ac, the autoconf program can be run. This program generates a configure and automate.cache file; running the configure script, a config.status script is generated based on the build environment. This config.status script, when run, converts input files (commonly Makefile.in files) into Makefile output files, which in turn can be used to generate appropriate executables. Autoconf is thus used to configure a Makefile according to the environment in which the Makefile and package of interest is contained. Used in conjunction with Automake, Autoconf allows for portability with source code packages. The job of Autoconf is primarily to make sure package Makefiles are configured correctly according to their respective environment.

Automake

In order to generate suitable Makefiles, it is necessary to first write a Makefile.am file containing the directories necessary for installing the package. In addition to a top- level Makefile.am file, one is also necessary for each directory to account for subdirectories. The Automake program takes in these Makefile.am files as input and subsequently generates Makefile.in files. As mentioned, these files are converted to working Makefiles via the config.status script, which was previously generated based on the build environment by Autoconf. Automake is therefore used to actually produce the package Makefiles based on the results of Autoconf and user-defined dependencies. Through the make program, Automake allows for seemless dependency tracking, which involves rebuilding any files dependent upon a modified file. Lastly, Automake works with multiple directory levels, producing a Makefile.in for any nested Makefile.am files in the package of interest.

Libtool

Because shared libraries are handled differently among different environments, portability is an issue with compiled libraries. This is where Libtool comes in, providing a method of abstracting the library-creation process. Through this abstraction, differences in environment are hidden, giving static and dynamic libraries much-needed portability. Libtool would be used whenever a library is utilized in a package but loses functionality in different operating environments. Libtool's configure script allows for the same portability via Autocof and Automake, but instead of packages, Libtool works with libraries.

Gettext

Oftentimes program comments and error messages are written in a certain language, while the person utilizing the program may not speak this language. In order to allow for translation of program output, gettext can be used by first wrapping strings of interest with a gettext function call (inside of the program's source code). Then, using the msginit program and the necessary locale (ie. language), the translator generates a Portable Object file. Editing this file with the corresponding translation involves declaring the original string as msgid and the translation as msgstr. Finally, using the msgfmt program, the Portable Object file is compiled into binary and can be distributed along with the original source code. Setting the environment variable LC_MESSAGES, the user can choose their preferred language.

Pkg-config

The many different installed libraries on a system determine the services that this system can provide the user. The pkg-config program allows for quick access and knowledge concerning the different libraries on a system, allowing for easier implementation of source code and other packages. As such, the primary use of pkg-config is to provide details concerning a library to which the user is interested in linking a program. Data concerning each library is stored in a respective pkg-config file with extension .pc. These pkg-config files contain keyword fields of information relating to the package. Adding known library directories to the PKG_CONFIG_PATH environmental variable, the user can easily find these pkg-config files and retrieve necessary information using the pkg-config program.