Compiling Software on UNIX

System Administration Decal
Spring 2008
Lecture #4
Joshua Kwan

Today

- How to turn source code into programs that run on Linux?
- What if that software needs other software to function?
- What if there's a problem during the build?
- Set up final project groups and assign virtual servers

Clarifications

- If you've had problems doing labs because of account problems, use your shiny new virtual server! (later on)
- Running programs; the $PATH variable.
  - check with "echo $PATH"
  - PATH does not include current dir by default
  - to run "foo" from your current directory, must type "./foo" to tell the shell where it is.

Administrivia

All labs up to and including **Lab 3** are now due **next Tuesday, 10/14.** If you fail to finish by then you will have to do that and an extra lab of my choosing, or risk an NP (per the original course rules.)

Types of Software Packages

- *Programs* – things you can run off the command line.
- *Libraries* – software that other source code can use the functions from.
- *Modules* – "extension" code written specifically to work with a certain program
- *Script libraries* – code archives in languages like Python, Perl, Ruby for various purposes.

The Three-Step Procedure

- Step 0: Download and unpack source
  - Generally, using the `tar` application. e.g.
    ```
    tar -xvzf MyProg-1.0.tar.gz
    tar -xvjf MyProg-1.0.tar.bz2
    ```
- Step 1: Run `./configure`
  - Prepares source for building on your particular system
- Step 2: Run `make`
  - Compiles source files to binaries (if applicable)
- Step 3: Run `make install`
  - Installs programs and data into system

The Three-Step Procedure

- This works in the majority (70-75%) of cases
- Many other software environments (e.g. scripting languages) have own system
- For example..
  - Python: `python setup.py install`
  - Perl: `perl Makefile.PL; make …`
- When in doubt, look for an INSTALL text file or a README

Patching Software

- When released software has issues, a **code patch** is released instead of a new version
- Generally come in the **unified diff** format, which the “patch” utility understands
- You should apply patches *before* you build, obviously - hence mentioning this here.

Patch Example

```
--- maildirtree-0.6/maildirtree.c~  2008-10-07 14:19:42 -0700
+++ maildirtree-0.6/maildirtree.c   2008-10-07 14:19:48 -0700
@@ -103,7 +103,7 @@
     {
       case 'h':
         puts(usage);
-        exits(0);
+        exit(0);

       case 's':
         summary = true;
```

- Example: `patch -p1 < fix.diff`
- `-p1`: If fix.diff wants to look for a/b/test.c, actually modify b/test.c.
  `-p2`: fix.diff looks for a/b/test.c, actually modifies ./test.c
- 99% of patches: Enter the source directory, then use `-p1`

What is make?

- Powerful build system! You'll be using the "GNU" version of make in this class
- Lets you specify what to build, how to build (compiler and arguments), order to build in
- Includes strong dependency system
- "Don't build my_program without having libprogram.a built already"
- "If I update foo.h, rebuild foo.c"

Configuration Options

- Configure script generally has options; try `./configure --help`
- You can enable features, point it to library install paths that it needs, use different compiler, etc.
- Reacting to a configure/build error often involves trying to find an option that will fix things.

Build Problems

- Missing library:
  - Download, build and install the needed library
- Missing compiler:
  - Install your OS's compiler distribution (e.g. Xcode or gcc package on Linux)
  - Make sure to install the C development headers!
- Compilation error:
  - Is your operating system supported by the author?
  - You could try and fix it… then submit your solution to the author!

Dependency Hell

- What if your program depends on libfoo?
  - Download libfoo source and try to build
  - libfoo depends on libbar!
  - Download libbar source and try to build
  - … *ad infinitum* …
- We call this “Dependency hell”
- Package systems in Linux distributions or ‘ports’ in BSD-type distributions can help.

Distribution Package Systems

- Want to install ‘program’ on Ubuntu?
  - Easy! `apt-get install program`
  - ‘program’ *and its dependencies* will be installed from binary packages.
  - Thus, `apt-get install program` is equivalent to `apt-get install program libfoo libbar`
- But: You can’t customize; desired program may not have a package; no learning involved.
- *Not allowed* to do this for your final project ☺

Final Project

- We're giving you a virtual Linux server
- Your goal: combine several pieces of open source software to make a cool **service**.
- Groups of three or four. Find some partners!
- Your "lab" this week includes setting up user and administrator access for all group members.

Final Project Ideas

- IRC server and services (NickServ, ChanServ)
- Open-source game server (bzflag, OpenArena)
- Multiple OS network boot server (PXE/DHCP)
- or any number of database-backed web applications... (using Apache/MySQL/etc.)

Virtual Servers

- If your server is “iXY”, login by doing:
  `ssh –p 2XY22 root@decal.ocf.berkeley.edu`
- Change your root password *immediately* using **passwd**
- The lab will walk you through getting everything else set up.
- Talk to me about port forwarding for any network servers you want to run for the proj.