Introduction to Make
and
Python Scripting

CS 35L
Spring 2018 - Lab 3

Getting Started

- Login to seasnet
- Download coreutils to a temporary directory
  - how can we download a file?
- Untar\Unzip it
  - How do you unzip a file?
  - man tar
  - *cd* into the newly created coreutils folder

Tar commands

- `tar –cvf <tarfilename.tar> <target directories>` - creates tar file.
- `tar –tvf <tarfilename.tar>` - list tar file contents
- `tar –xvf <tarfilename.tar>` - extracts tar file
- Can add `–z` flag for newer LINUX distros with gnuzip for automatic compress/decompress (.gz suffux).
- Otherwise try compress command (`.Z` suffix)
- USAGE:
  - Always create tarfile in target directory (relative file/directory names)
  - Always list tarfile before extracting (insure relative file names)
  - Always extact tarfile in target directory (relative file/directory names)
- Example:

```
        tar –xzvf ~/bb-1_3a_tar.gz
```

Compiling from scratch

- Common scenario
  - You download a utility from the internet to your unix machine
  - There are no binaries, but source code and makefile is available
  - Compile and build to install it
  - Reading text files in the program folder gives clues how to install the program
    - Usually INSTALL, README, readme.txt, install.txt and so on

Compiling from scratch

- Common scenario
  - The order of compilation is usually:
    - **`./configure`**
    - **`make`**
    - **`make install`**
  - man make for more details
  - view Makefile in the programs folder for details
    - e.g. search for “install:”

Configure script

- Designed to aid in developing a program to be run on a wide number of different computers
- `configure` is application specific
  - software provides it’s own configure script
- Creates the Makefile
  - Can change default behavior with options
  - `./configure --` help for more info

Makefile and make

- We need a file that instructs make how to compile and link a program. Called a Makefile.
- The make program allows you to use macros, which are similar to variables to codify how to compile a set of source code.
  - Macros are assigned as BASH variable:
    - CFLAGS= -O -systype bsd43
    - LIBS = "-lncurses -lm -lsdl“
- Makefile is invoked with
  - make <target_name>

Standard “targets”

- People have come to expect certain targets in Makefiles. You should always browse first, but it's reasonable to expect that the targets all (or just make), install, and clean will be found
  - **`make`** - compile the default target
  - **`make all`** - should compile everything so that you can do local testing before installing things.
  - **`make install`** - should install things in the right places. But watch out that things are installed in the right place for your system.
  - **`make clean`** - should clean things up. Get rid of the executables, any temporary files, object files, etc.
- Link to more info on makefile

Makefile example

- Makefile:

```
SHELL = /bin/sh
MAKE = make
CC = g++
LIBS=
CFLAGS=-DSIGSETJMP -O

hello: main.o hello.o factorial.h
        ${CC} ${CFLAGS} -o $@  main.o hello.o ${LIBS}

clean:
        rm -f *.o
```

- Run as `make; make clean`

Compiling coreutils

- Go into coreutils directory. This is what you just unzipped.
- Read the INSTALL file on how to configure “make,” particularly the --prefix flag.
- Run the configure script so that when everything is done, coreutils will be installed into your temporary directory
- Compile it: make
- Install it: make install

Bug appears with newly built coreutils

```
[User:~)@lnxsrv07 ~/cs35L/lab3/coreutils/bin]$ ls -l /bin/bash
-rwxr-xr-x 1 root root 960376 Jul  8  2015 /bin/bash


[User:~)@lnxsrv07 ~/cs35L/lab3/coreutils/bin]$ ./ls -l /bin/bash
-rwxr-xr-x 1 root root 960376 2015-07-08 04:11 /bin/bash
```

Notice the difference between invoking ls commands above

Applying the Patch

- Read the patch bug report
  - lists.gnu.org/archive/html/bug-coreutils/2009-09/msg00410.html
- Understand what part of the code is being fixed

Applying the Patch

```
diff --git a/src/ls.c b/src/ls.c
   index 1bb6873..4531b94 100644
   --- a/src/ls.c
   +++ b/src/ls.c
   @@ -2014,7 +2014,6 @@ decode_switches (int argc, char **argv)
                 break;

               case long_iso_time_style:
   -           case_long_iso_time_style:
                 long_time_format[0] = long_time_format[1] = "%Y-%m-%d %H:%M";
                 break;

   @@ -2030,13 +2029,8 @@ decode_switches (int argc, char **argv)
                      formats.  If not, fall back on long-iso format.  */
                   int i;
                   for (i = 0; i < 2; i++)
   -                 {
   -                   char const *locale_format =
   -                     dcgettext (NULL, long_time_format[i], LC_TIME);
   -                   if (locale_format == long_time_format[i])
   -                     goto case_long_iso_time_style;
   -                   long_time_format[i] = locale_format;
   -                 }
   +                 long_time_format[i] =
   +                   dcgettext (NULL, long_time_format[i], LC_TIME);
                 }
               }
           /* Note we leave %5b etc. alone so user widths/flags are honored.  */
```

Apply the Patch

- Just use an editor (eg. emacs, vim).
- Recompile it: make
- A new executable ls file is created
- Compare results between new ‘correct’ executable and ‘buggy’ installed version
- Did the patch fix the bug?

General tarball/make example

```
tar -vxzf <gzipped-tar-file>
cd <dist-dir>
./configure
make
make install
make clean
```

Homework

Python Scripting