UNIVERSITY OF CAMBRIDGE

MRC | Cognition and Brain Sciences Unit

Introduction to Linux at the CBSU Part 3 Advanced Concepts and Scripting

Jeff Berry

MRC Cognition and Brain Sciences Unit, Computing group

Previously on Intro to Linux ...

- **PuTTY**
- **CLI**
- **VNC/X2GO**
- **Basic Commands**
    - cd
    - pwd
    - ls
    - man
    - (and so on)

Files, Permissions, and Owners, Oh My!

- **Users and Groups**
  - Unique Username and User ID (UID)
    - *whoami; id –un; id -u*
  - May be member of multiple groups
    - *id -gn; id -G*

- **Files and Directories**
  - Single owner and single group
    - altered with *chown*
  - Three sets of permissions and three kinds of permissions
    - owner, group, and world(all)
    - read (r), write (w), execute (x)

Files Permissions and Owners

ls –n

```
                 uid   gid        timestamp     name
-rwxr-xr-x+ 1 6831  222  214   Oct 25  2016    test.csh
permissions                 size
```

number of links (usually 1 for files, for directories is a count of subdirectories and current directory)

More complicated permissions and ownerships are possible but beyond the scope of this class.

Files Permissions and Owners

```
-    rwx    r-x    r-x    +
```

special extra information

world permissions

group permissions (- means no permission)

owner permissions

file type: l = link, d = directory, - = file

Permissions often referred to as octal numbers

- read = 4, write = 2, execute = 1
- rwx = 7, r-x = 5, rw- = 6
- 755 and 664 are common
- anything world-writable is scary

Programs and Executables

- **Executables must have executable permission set**
  - group execute is sufficient if user is member of group
  - file must also be something which can be executed
    - a script
    - a compiled program

- **The shell must be able to find the code**
    - in the path ($PATH)
    - or given explicit path
      - fully qualified – starting at /
      - relative – eg ./*executable* for current directory

Why Bother?

Accuracy and Ease

- **Shell Scripts Can Contain lists of commands**
    - **Example**
        - You want to set up datestamped working directory in your imaging space. One way is to change directory to your imaging space, figuring out the day and then use mkdir to create the directory.
            - cd /imaging/${LOGNAME}
            - set DATE = `date +%y%m%d`
            - set WORKDIR = "work"${DATE}
            - mkdir $WORKDIR
        - The file is called 'makework' in your imaging space

Why Bother?

Speed and Efficiency

- **Loops and Iteration**
  - **Example**
    - You’ve run some code (DataGen.csh) and generated a lot of output files
      - *ls SampleOutputDir*
      - How many files? Hint: use *ls* (with a flag), *wc*, and a pipe.)
    - Now you want to re-run the code with a slight change
      - save old data
      - look for changes
    - Script it up!
      - retain existing files, date-stamped in some way
      - run the code
      - compare the new output with the old
        - Different files?
        - Differences in files with the same name?

Why Bother?

Speed and Efficiency

- **Example (cont.)**
  - Rerun.csh should to the job
    - *cat Rerun.csh | more* (or *more Rerun.csh*)
  - set variables – change them to address changing circumstances
  - loop over files
  - creates a shell with working directories, environment, and so forth
    - Remember environment variables propagate – so your shell environment will propagate down, but …
    - if you are submitting a job remotely to the cluster, the environment will be different, so …
    - good practice is to set whatever you know you need in the script itself.

Have a go …

Try this:

- make a directory in your imaging space called ‘SampleOutputDir’
- run Datagen.csh
- have a look in SampleOutputDir
- change the names of a few files, or remove a few, or add new ones (explore the ‘touch’ command), copy one file over another
- run Rerun.csh

Compilation and Interpretation

There are (broadly speaking) two kinds of executable files

- scripts, which usually means interpretable files
    - scripts are evaluated as they are executed
    - single step process
    - sometimes line by line
    - slow but portable
- programs, which usually means compiled files
    - programs are compiled before they are executed
    - two step process – compile and run
    - fast but need to be compiled on the platform where they will be run.

Scripting languages

So many options:

Scripts can be written using only shell commands, or can be written in one of any number of scripting languages such as:

- perl
- python
- php
- even (shudder) PowerShell
- and many more

Each has advantages, disadvantages, proponents, and detractors.

Simple example

Example:

test.csh, test.pl, and test.py all do the same thing.
Each does similar things in different ways internally.

For simple tasks, a simple shell script is good - the commands are just linux commands in a file. (Mostly.)

For more complex tasks, a more fully featured scripting language is often preferable, although it will require some programming skill.

The more you want to manipulate data, the more likely it is that a scripting language is the right choice.

A Mostly Real World Example

There was a directory on our website with pictures of staff with names in the format firstname.lastname.pdf – a new system wanted the format of firstnamelastname.pdf – ie without the middle '.'

It could be done by hand using mv to rename the files (or cp to copy them with the new name), but for a couple of hundred files that is both time consuming and error-prone.

The directory FixFiles in the imaging space replicates this problem.

A Mostly Real World Example

A python script called ‘genfiles.py’ is used to generate a bunch of dummy filenames, all ending with .dwarf

The simple shell script in the src directory, fixname.sh, does what we want it to do.

Further Information

- **Linux:**
  - Basics:
    - http://imaging.mrc-cbu.cam.ac.uk/methods/unixsurvivalguide
    - http://www.ee.surrey.ac.uk/Teaching/Unix
  - Scripting:
    - http://tldp.org/LDP (both Beginner and Advanced)