Introduction to Linux – Part 2

Zhiyu (Drew) Li & Martin Cuma
Research Consulting & Faculty Engagement
Center for High Performance Computing
{zhiyu.li; martin.cuma}@utah.edu

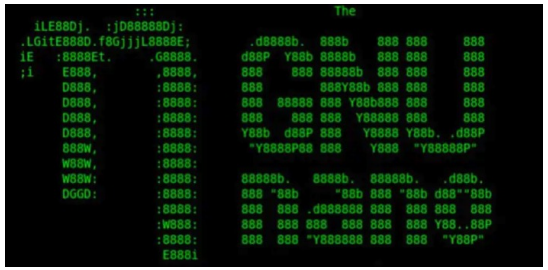
Linux Virtual Machine

- ❑ Get a temporary account (or use your own CHPC account)
- ❑ Virtual Machine FastX portal: https://linuxclass.chpc.utah.edu:3300
- ❑ Open a XFCE Terminal
  - ❑ Adjust Font size: Edit → Preferences → Appearance → Click on Font → adjust Font Size
- ❑ Use Bash shell (quick check: echo $SHELL);
  - ❑How to change to bash: /bin/bash
- ❑ Copy and Paste issue on Mac;
- ❑Prepare example data (if you missed Part 1)
  - ❑cd ~
  - ❑mkdir LinuxClass
  - ❑cd LinuxClass
  - ❑wget https://home.chpc.utah.edu/~u0028729/CHPCPresentation/shell-lesson-data.zip
  - ❑unzip shell-lesson-data.zip
  - ❑cd shell-lesson-data

Editors

There are many choices – a few are:

- nano



- vi/vim

- emacs

Nano Editor

To start either

nano

OR

nano filename

-- if filename exists, it will open file in editor; if it does not, this will be the name used when you save the file.

if you start nano without a filename it will prompt you for a name when you "WriteOut" using ^O (Ctrl + O)

^? → Ctrl + a specific Key; How to quit nano session: ^X → Ctrl + x

Nano cheat sheet: https://www.nano-editor.org/dist/latest/cheatsheet.html

Vi editor

- Another common choice
  - Start with the command **`vi`** or **`vi`** filename
  - **`vi`** at CHPC is actually **`vim`**, which is an improved version of **`vi`**
  - More feature rich, takes more time to learn
- Not going into detail but we do provide a vi cheat sheet and a vi graphical cheat sheet – linked on the presentation page https://www.chpc.utah.edu/presentations/IntroLinux3parts.php
- There is also a tutor program – start with command **`vimtutor`** which is a great tool to learn to use the program
- How to quit Vi session (discard changes): ESC → : (SHIFT + :) → q!

Loops

- Used when you want to preform the same action many times, such as on multiple files
- There are a number of ways you can do this
- One option
  - List multiple arguments for a command to act upon
  - Example (go to the `LinuxClass/shell-lesson-data/exercise-data/creatures` directory):
    - **`head -n 3`** basilisk.dat minotaur.dat unicorn.dat
- Another option – do a loop with a for/do statement

Loop Terminology

- in bash syntax a loop looks like:

```bash
for thing in list_of_things
do
    operation_using $thing    # Indentation within the loop is not required, but aids legibility
done
```

- In this loop **thing** is a **variable**. During execution **$thing** is set to the first item in the list, the operation(s) is done, then it goes to the second and repeats the operation(s), etc until it reaches the last item in the list. Then the loop is exited.
- You can choose anything for **thing** (eg mything, item, myfile…) – however, your choice of the what to use for thing should help a person reading the file understand what the loop is doing and what it is acting upon
- Examples of **list_of_thing**: 2 4 6 8 10; ← a list of numbers

  {2..10..2}; {start..end..step} ← A number range

  fileA fileB fileC fileD; ← a list of filenames

  file*; ← a list of filenames represented by wildcard

  $variable or $(command) representing a list: $(ls) ← variable or command

Exercise

- in bash syntax a loop looks like:

```bash
for thing in list_of_things
do
    operation_using $thing    # Indentation within the loop is not required, but aids legibility
done
```

- Go to the directory `shell-lesson-data/exercise-data/proteins` and write a loop that lists all of the pdb files
- Write a loop that prints all file names – can use **echo** command
- Build upon this loop by adding another command
  - Copy each file to a new name based on the existing name, <file>.pdb to orig-<file>.pdb
- Add another command to the loop to make a single file that has the content of all of the pdb files in this directory
  - can use redirect >, >>

```
for file in *.pdb
 do
    echo $file
    cp $file orig-$file
    cat $file >> all.pdb
 done
```

Exercise – Nelle's Data

- To process her data files Nelle will need to run an analysis on each of her sample files in **north-pacific-gyre**. The files to be processed have the consistent names of **NENExxxxA.txt** and **NENExxxxB.txt**
- The analysis requires running a script – more on this next time – written by her supervisor is called **goostats.sh**; this script acts on one sample file at a time; it requires two arguments – the input file name and the output file name.
  - **bash goostats.sh <INPUT> <OUTPUT>**
- Nelle decides to call the output file **stats-NENExxxA.txt** – prepending the filename with **stats-**
- She is being careful so she wants to test (using echo instead of running the script
- Hint – start with 1 file and run test to create the two arguments. You can prepend a v**ariable** with additional information: **stats-$variable**

Exercise – Nelle's Data Answer

- To make sure getting all of the files

```
for datafile in NENE*A.txt NENE*B.txt
do
   echo $datafile
done
```

- Test getting the output file name

```
for datafile in NENE*A.txt NENE*B.txt
do
   echo $datafile stats-$datafile
done
```

- Executing the script across all files

```
for datafile in NENE*A.txt NENE*B.txt
do
  echo $datafile
  bash goostats.sh $datafile stats-$datafile
done
```

Some other useful Linux commands

- **`cut`** – e.g. **`cut -d , -f 2,3 animals.csv`**
  - **`Example file: shell-lesson-data/exercise-data/animal-counts`**
  - **`-d (delimiter) -f (column ids)`**
  - Prints selected parts of lines from file to standard output (screen)
- **`du`** – e.g. **`du –h or du -sh`**
  - Scan a given file/directory (and subdirs) and report space usage; -s give summary of total usage, -h gives it in “human readable” format of K, M, G
- **`df`** – e.g. **`df –h`**
  - Overview of file system disk space usage (-h: human readable)
- **`ln`** – e.g. **`ln -s ~/bin/prog.exe prog1.exe`**
  - create a link between files (ln -s FILE LINK)

On your own – Use and explore options of these commands

Linux File Permissions

- Each Linux file belongs to a specific user (u) and a specific group (g)
- Shown with **`ls -l`**
  - -rw-rw-r-- 1 u0028729 chpc 86 Jul 30 02:41 notes.txt
  - -: file; d: directory
  - : user (u); : group (g); : others (o)
  - r: readable; w: writable; x: executable ("cd"-able for directory); -: no permission
    - examples: rwx; r-x; r--; ---;
- **`chmod`** – to change permissions of file or directory, can set
- Examples:
  - **`chmod u=rwx file`** ← **Set User permissions to Read Write and Exec**
  - **`chmod g+x file`** ← **Grant Group Executable permission**
  - **`chmod o-rwx *.c.`** ← **Remove all permissions for Others (not User, not Group)**
- Executable files (programs and scripts) must have executable permissions; directories must be executable in order to be able to cd into them
  - **`chmod +x *.sh`**

Login Scripts & Environment Variables

- In your home directory are a number of dot files - `.bashrc` and `.custom.sh`, `.tcshrc` and `.custom.csh` Depending on your shell choice, the appropriate pair of these are executed during login.
- These set the environment (as environment variables) needed for you to work on CHPC resources
- Commands to check your environment: `env` or `printenv`
- Some important variables
  - $USER
  - $HOME
  - $PATH – paths to search for commands
  - $LD_LIBRARY_PATH – paths to search for libraries when linking a program (more on that later)

Processes

- A Process is a running Linux program
  - Each process has a PID (Process ID)
- **top** displays processes and resource usage in real time (Ctrl + C to quit)
  - top –u <user>
  - Ctrl + C to quit
- **`ps`** reports a snapshot of current processes
  - **`ps, ps x`** Display ALL of your processes
  - **`ps ax`** Display ALL processes
  - **`ps aux`** Display ALL processes (more detailed)
- **`kill PID`** kills the process with the specified PID
- **`killall processname`** kills all process with the processname
- **`kill -9 PID`** kills the process with the specified PID if a kill does not work

Monitoring processes/usage

- **uptime** – how long the system has been running
- **free** – free –h, memory and swap usage
- enhanced **top**
  - **atop** (available on CHPC clusters)
  - **htop** (available on CHPC clusters)
- **sar** – historical system usage report (cpu, memory, I/O…)

Other Job Controls

Note: here “Job” means a local process

- Ctrl+C (^C) terminate the currently running process
- Ctrl-Z (^Z) suspends/pauses the currently running process
  - Example: ping www.byu.edu (check if a address is reachable)
- **jobs**: lists all jobs, with their number
- **fg %n**: bring a program back to the foreground (and continue/resume execution)
- **bg %n**: puts current or specified job (%n) in the background (and continue/resume execution)
- **&** (ampersand) runs the job in the background
  - Example: sleep 10 VS sleep 10 &

Moving files to/from CHPC

https://www.chpc.utah.edu/documentation/data_services.php

- Can mount CHPC file systems on your local machine (Windows, Mac or Linux), must be on campus or using the campus VPN
- Windows – there are graphical tools such as WinSCP
- Mac, Windows, cloud options – cyberduck, another graphical tool
- Linux
  - **`scp`** command (secure shell copy) – to copy files between linux systems
- **`wget`** – to download from web with URL
  - **`curl`** is another option
- For larger data sets – look into the Data Transfer Nodes (DTNs) and transfer tools such as Globus, see
  - https://www.globus.org/quickstart
  - https://www.chpc.utah.edu/documentation/data_services.php

Have Questions?

helpdesk@chpc.utah.edu