Linux command-line for HPC workshop

9/14/17

NORTHERN ARIZONA UNIVERSITY

Outline

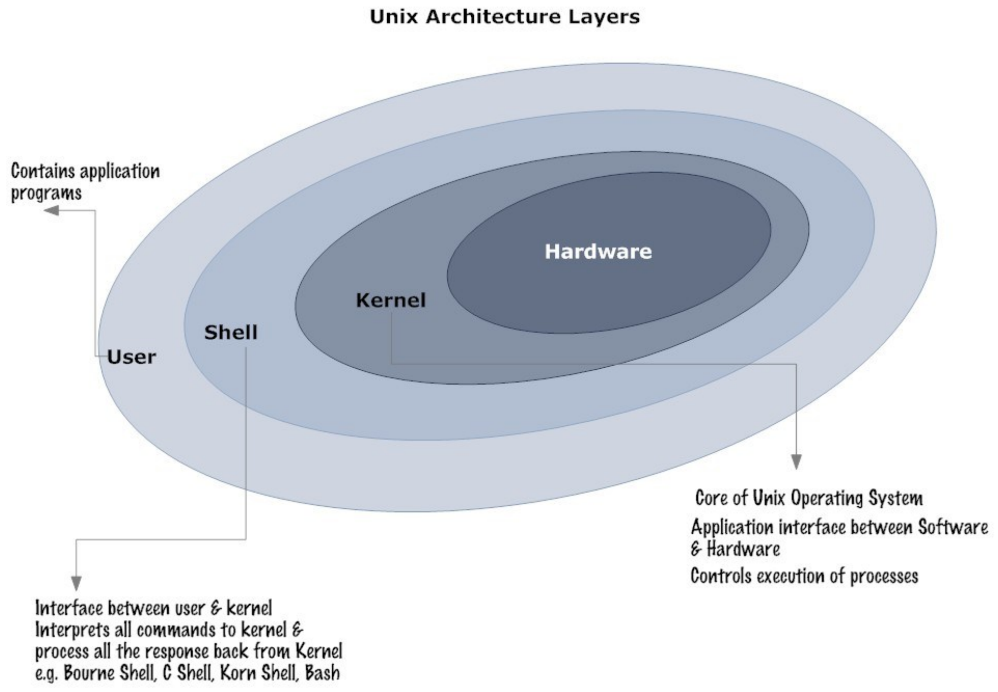
- Introductions
- What is Linux?
  - Brief info, and statistics
- The command-line, there's no GUI’s here!
- Intro to the shell
- Navigating “the system”
- Choose your editor
- Moving data around
  - Locally
  - Remotely
- In line scripting foo
- Monitoring system processes

Introductions

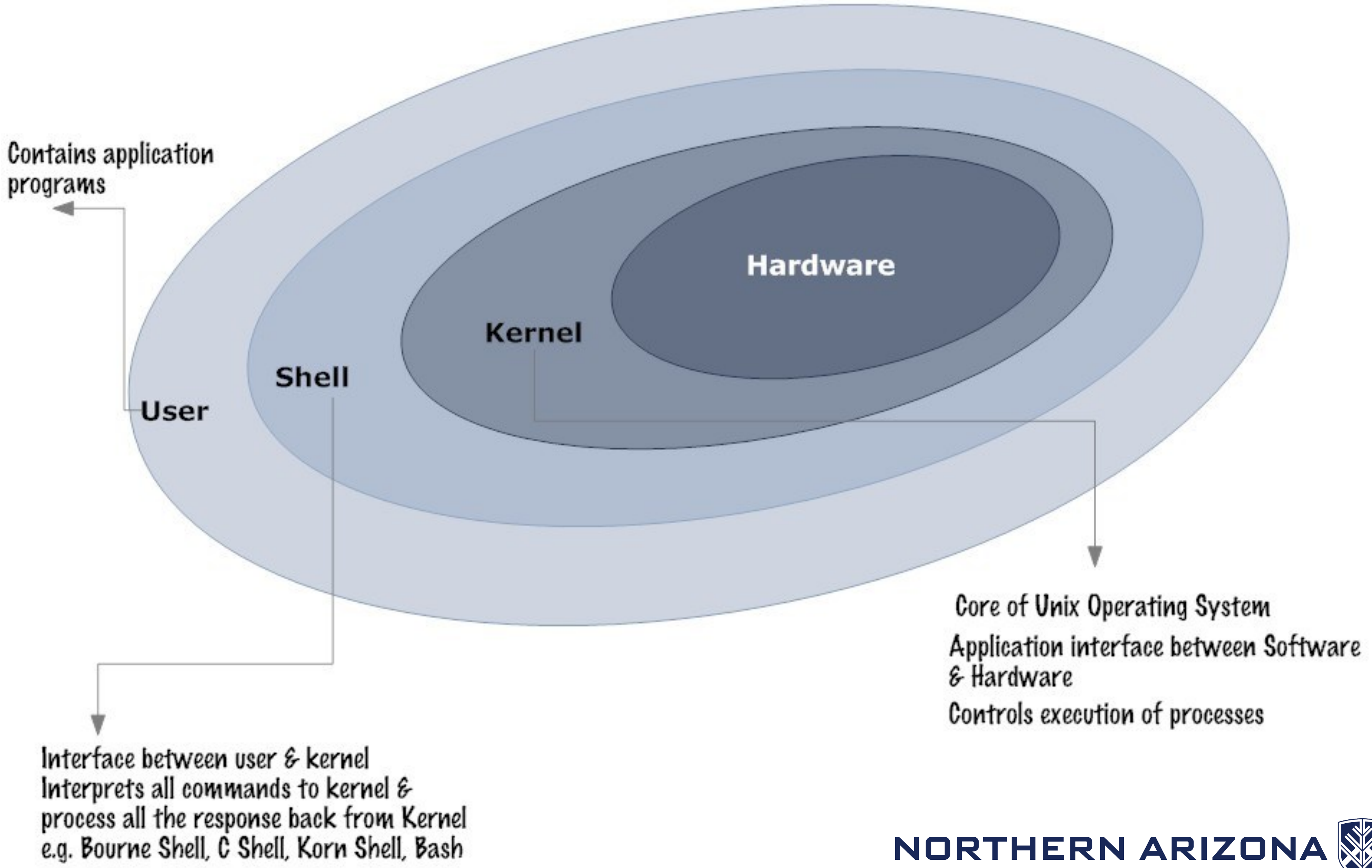
- Introduce yourself
  - Name
  - Department / Group
  - Linux or Unix experience

What is Linux?

- Linux is a computer operating system similar to windows. It is an open-source operating system where the defining piece is the Linux kernel which was developed by Linus Torvalds in 1991.
- The Linux operating system is:
  - Linux kernel
  - GNU applications
  - Open-source

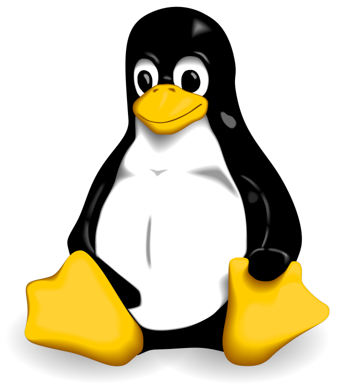

Unix Architecture Layers
Contains application programs
Hardware
Kernel
Shell
User
Core of Unix Operating System
Application interface between Software & Hardware
Controls execution of processes
Interface between user & kernel
Interprets all commands to kernel & process all the response back from Kernel
e.g. Bourne Shell, C Shell, Korn Shell, Bash

Unix Architecture Layers



Contains application programs
User
Shell
Kernel
Hardware
Core of Unix Operating System
Application interface between Software & Hardware
Controls execution of processes
Interface between user & kernel
Interprets all commands to kernel & process all the response back from Kernel
e.g. Bourne Shell, C Shell, Korn Shell, Bash

Linux continued



- Linux powers businesses, universities, the internet, and HPC clusters.
- Linux powers 99% of the top 500 HPC clusters in the world, the other 1% is, a flavor of Unix
- http://www.top500.org/statistics/details/osfam/1
- HPC is the future of computing, and a look at what will be on your desktop, or your hand in 10 years
- If HPC is a look at the future, and is powered by linux, shouldn't you possess the necessary skills to do your research on linux?

Lets get started

- You are used to a GUI
- We will use a text-based shell in this workshop
  - Mac OSX – already has this satisfied (terminal)
  - Windows – use putty, download here: http://www.putty.org/
  - Linux – already has this satisfied, of course!
- We use the shell to interface with the operating system
- `Shell commands will be colored like this`
- Information that will vary for each person will be entered in `<pointy brackets>`

Many types of shells

- The shell is how the user interacts with the operating system
- The user types commands into the shell which interprets them, and sends to the kernel where the work is done
- The standard Linux shell (used in this course) is bash, or “Bourne-again” shell
- Other shells
  - **sh** Bourne shell, original AT&T Unix shell
  - **tcsh** “Enhanced C shell”
- See what shell you are using
  - `echo $SHELL`

Logging in

- You must first be on the NAU network, or NAU VPN
- Login to monsoon with ssh
  - Mac (with terminal):
    - `ssh <nauid>@monsoon.hpc.nau.edu`
  - Windows (via Putty):
    - In the host name box insert, <nauid>@monsoon.hpc.nau.edu
    - Create a session name, hit save
    - Finally, "open", or double click your session
- You are prompted to accept a SSH key, type Y. This happens only once per initial connection
- Type your LOUIE password, there is no feedback (no stars or characters printed)
- Hit enter

I’m logged in, now what?

- You should see something like this:
  - [cbc@wind ~]$
- You are now logged into one of monsoon’s login nodes, in this case “wind”
- Note that the login nodes are not meant for processing, they are **solely** for:
  - Editing / submitting job scripts
  - Compiling programs
  - Moving data to/from monsoon
  - Debug work (short tests of <=30 min)

NORTHERN ARIZONA UNIVERSITY

Navigating with the shell

There is no mouse, so we must use the keyboard keys!

- Arrow keys to move back and forth, up and down to move through bash history (previous commands)
- TAB to auto-complete a command, file, or directory
- Control-c to interrupt any program
- Control-z to suspend programs
- Control-r to search bash history
- Enter to execute the command

Try some commands out

`pwd`	- print working directory (where you currently sit)
`id`	- this is your user id, and the groups you belong to
`ls`	- list files in the current directory, variations "-la", "-h"
`w`	- who's logged in today, system load, and uptime
`quota`	- report your quota
`df`	- list mounted filesystems
`date`	- get the current date
`cal`	- a handy calendar!
`echo`	- print a message, e.g. echo "hello world"

- Try using the up arrow to check out your history!

More commands

`man <command>`	- pull up the manual page for the command
`cat <file>`	- print contents of a file to the screen
`more <file>`	- page through files
`less <file>`	- less is more!
`file <file>`	- determine the type of file: txt, data, exe
`mkdir <dirname>`	- create a directory name ”dirname”
`rmdir <dirname>`	- remove a directory named “dirname”
`rm <filename>`	- remove a file
`cd <dirname>`	- change directory to “dirname”
`touch <file>`	- create an empty file, or update modified time stamp

Command Options (Flags)

- Most commands have extra options or “flags”
- Specify options with a dash “-”, and “--”
- Multiple options usually go directly next to each other, order may not matter
- Examples
  - `ls -a` (list all files, including hidden ones)
  - `ls -alrt` (list all files, long listing, sort reverse by time)
  - `top -bn1` (run top in batch mode)

Navigating the system

Everything in linux is a file and all of the files and filesystems are laid out in one huge hierarchical directory structure. Note, this is a bit different from windows where there is a separate directory structure for each storage device connected to the system.

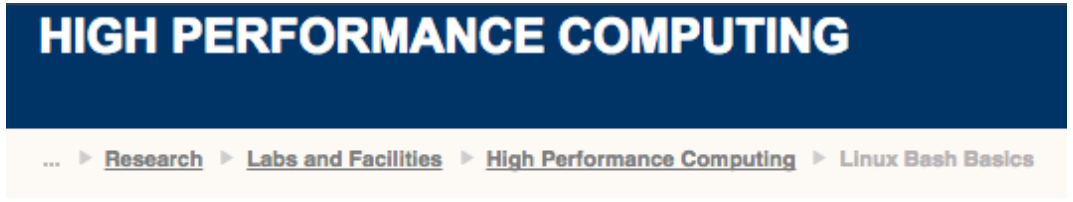
Quick whiteboard demo

To navigate the system we will use:

- `pwd` – print working directory
- `ls` – list files in current directory
- `cd` – change directory
  - `cd ..` , `cd ~` , `cd <dir>` , `cd <dir/subdir>`

Pathnames

- A Path is similar to web ”breadcrumbs”
- Absolute vs Relative Paths



HIGH PERFORMANCE COMPUTING

... ▸ Research ▸ Labs and Facilities ▸ High Performance Computing ▸ Linux Bash Basics

```
[cbc@wind ~/linux_workshop ]$ pwd
        /home/cbc/linux_workshop
[cbc@wind ~/linux_workshop ]$ ls
        file  file2
```

```
Absolute: cat /home/cbc/linux_workshop/file
        hello world!
Relative: cat file
        hello world!
```

Same result in this case, using relative is a shorthand, but it can cause problems because it’s ambiguous.

** Note that Linux filesystems are CASE SENSITIVE in regards to files, which includes directories!!*

** Also, not recommended to have spaces in filenames and directory names! It can be a pain*

List the files in your directory

```
[cbc@wind ~/linux_workshop ]$ ls
   hello
[cbc@wind ~/linux_workshop ]$ ls -l
total 4
-rw-r--r-- 1 billy ITS-HPC-coffey 8 Sep 22 13:56 hello

[cbc@wind ~/linux_workshop ]$ ls -la
total 48
drwx------   2 billy root           4096 Sep 22 13:56 .
drwxr-xr-x 234 root  root           8192 Sep 16 13:06 ..
-rw-r--r--   1 billy cluster         213 Jul 11  2014 .bashrc
-rw-r--r--   1 billy ITS-HPC-coffey    8 Sep 22 13:56 hello
```

NORTHERN ARIZONA UNIVERSITY

Listing files

Note the funny looking files there:

```
drwx------   2 billy root           4096 Sep 22 13:56 .
drwxr-xr-x 234 root  root           8192 Sep 16 13:06 ..
-rw-r--r--   1 billy cluster         213 Jul 11  2014 .bashrc
```

File names that begin with a "." are hidden.

- "." is the current directory
- ".." is the parent directory
- .bashrc is a hidden bash configuration file

Lab 1 – directory structure

- Print your working directory, where you are currently (`pwd`)
- List the contents of the directory you are in (`ls`)
- Create a directory in your home directory named “linux” (`mkdir`)
- Change directory to the new directory “linux” (`cd`)
- Create a directory named “is” inside of the linux directory (`mkdir`)
- Change directory to the “is” directory (`cd`)
- Create a file in the ”is” directory named “awesome” (`touch`)
- Change directory back to your home (`cd ~`)
- Do a recursive listing on the “linux” directory: `ls –lR linux`

Lab 1 - Solution

```
[cbc@wind ~ ]$ ls -lR linux
linux:
total 0
drwxr-xr-x 2 cbc clusteradm 28 Sep 21 14:20 is

linux/is:
total 0
-rw-r--r-- 1 cbc clusteradm 0 Sep 21 14:20 awesome
```

File permissions

drwxr-xr-x	2	cbc	clusteradm	28	Sep 21 14:20	is
1	*2*	*3*	*4*	*5*	*6*	*7*

1. The mode and type of the file, in this case a directory (d), mode 755

- From left to right: Type, User, Group, Other
- Type is directory (d) → could also be – (file), l (link), and others
- User has read (r), write (w), and execute (x) → r (4) + w (2) + x (1) = 7
- Group has read (r) and execute (x) → r (4) + x (1) = 5
- Other has read (r) and execute (x) → r (4) + x (1) = 5

2. Number of hardlinks (you can kinda forget about this)

3. The owner

4. The group

5. Size of the file in bytes

6. The date, of last modified

7. The name of the file or directory

Changing file permissions

- Default permissions for files and directories:
  - File: rw- for owner, r-- for group, and r-- for others
  - Directory: rwx for owner, r-x for group, and r-x for others
- Change owner
  - `chown bob file`
- Change group
  - `chgrp ninjas file`
- Change mode (permissions)
  - `chmod g+rw file` – add read and write for group
  - `chmod +x file` – add execute to a file, for user,group,other

Managing files

`cp file new`	- copy a file to “new” file, or “new” directory
`cp –f file new`	- force copy a file
`mv file file2`	- move a file to a directory “file2” or rename file
`touch file`	- create empty file, or update time stamp
`rm file`	- remove file
`rm -f file`	- force removal of file
`rm -r dir`	- recursively remove a directory
`rm –rf dir`	- force remove recursively (CAUTION!!!!)
`rmdir dir`	- remove directory
`mkdir dir`	- make directory

Editors

- We need a text editor to edit, and create text files
- Lots of editors:
  - Nano - simple to use
  - Emacs - powerful
  - Vi - powerful

- Start out using nano, then change it up and use a better editor later.

Lab 2 – Editing/moving files

- Change directory to the linux/is directory (`cd`)
- Rename the file “awesome” to be “best” (`mv`)
- Make a directory in the “is” directory named “the” (`mkdir`)
- Move the “best” file to the “the” directory (`mv`)
- Edit the “best” file, with contents “of course!” (`nano, emacs, vi`)
- Copy the “best” file to the “is” directory, naming it “fun” (`cp`)
- Find out what type of file ”fun” is (`file`)
- Print the contents of the “fun” file (`cat`)
- *EXTRA* Create a file called ”a_secret” within the “is” directory with contents, “Professor plum, with the lead pipe”. This file should only be readable by you! (hint … chmod)
- *BONUS* Make the file hidden

Lab 2 – Editing/moving files Solutions

This is how it should look!

```
[tct49@wind ~ ]$ cd linux/is
[tct49@wind ~/linux/is ]$ mv awesome best
[tct49@wind ~/linux/is ]$ mkdir the
[tct49@wind ~/linux/is ]$ mv best the
[tct49@wind ~/linux/is ]$ cd the
[tct49@wind ~/linux/is/the ]$ nano best
[tct49@wind ~/linux/is/the ]$ cd ..
[tct49@wind ~/linux/is ]$ cp best fun
[tct49@wind ~/linux/is ]$ file fun
fun: ASCII text
[tct49@wind ~/linux/is ]$ cat fun
of course
```

Archiving, compression, and disk usage

- In windows, and mac you've worked with archives before. Maybe they were zip files, or possibly tar files. Tar (tape archive) files are similar to zip files. They are an archive file that holds many files. Making transport of many files easy.
- Normally in linux, datasets and source file bundles are packaged in tar files and compressed a number of ways. You may also still see zip files from time to time.
- Tar files are compressed most often times with:
  - gzip
  - bzip2

Archiving, compression, and disk usage

- Archives may take the form of:

data.tar	- uncompressed tar file
data.tar.gz	- tar file compressed with gzip
data.tgz	- same, but different extension
data.tar.bz2	- tar file compressed with bzip2
data.zip	- zip file

- Can always check with the handy “file” command:

```
file R-3.1.0.tar.gz
```

```
R-3.1.0.tar.gz: gzip compressed data, from Unix, last modified: Thu Apr
10 00:10:53 2014
```

Archiving, compression, and disk usage

`tar cvf file.tar directory`	- create archive of the directory “directory”
`tar xvf file.tar`	- expand the file.tar archive into current dir
`tar tvf file.tar`	- view the contents without extracting
`tar xvzf file.tar.gz`	- expand a compressed archive
`tar tvzf file.tar.gz`	- view the contents of a compressed archive
`gzip file.tar`	- compress a tar file with gzip
`gunzip file.tar.gz`	- uncompress a tar file with gunzip
`bzip2 file.tar`	- compress a tar file with bzip2
`bunzip2 file.tar.bz`	- uncompress a tar file with bunzip2
`du file`	- show the disk usage of a file, or directory
`du -h file`	- in human readable format
`du –sh dir`	- like above, but the sum of all files in a dir
`ls -h`	- list sizes of files in human readable format

Lab 3 – File Compression and disk usage

- Cd ~
- Create a tar archive named linux.tar of your "linux" directory, and store the archive in the root of your home directory (`tar`)
- Run the file command on your new archive (`file`)
- How much space is taken up by this archive? (`du`)
- Compress your archive file with gzip (`gzip`)
- How much space is taken up by the compressed archive? (`du`)
- Run the file command on your new compressed archive (`file`)
- While in the root of your home directory, expand the archive:
  - /common/contrib/workshops/linux/linux_data.tar.gz (`tar, gunzip`)
- Inspect the contents (ls)
- Find out how much space is taken up by the linux_data directory (`du`)

NORTHERN ARIZONA UNIVERSITY

Wildcards

- While in the shell, you can select files/directories based on wildcards on standard regex
- [0-9] - select elements with numbers
- [a-z] - selects elements with letters
- ^ - selects everything but, e.g. [^0-9] (not numbers)
- ? - matches any 1 character
- * - 0 or more

Wildcard Examples

- While in the shell, you can select files/directories based on wildcards on standard regex (regular expression)
- `ls *.txt` - lists all files/folders that end in ".txt"
- `ls lin*` - lists all files/folders that start with "lin"
- `ls *[0-9]*` - lists all files/folders that have a number in them
- `ls [a-z]*` - lists all files/folders that begin with a letter
- `ls [^a-z]*` - lists all files/folders that don't begin with a letter
- `ls ??[0-9]*` - lists all files/folders that have a number at the 3rd position

Lab 4 – Wildcards

- Switch to the `/common/contrib/tutorials/sc15` directory
- List all the files that end with ".pdf"
- List all the files that have the exact string "ADIOS" in them

Lab 4 – Wildcards Solutions

```
[mkg52@wind ~ ]$ cd /common/contrib/tutorials/sc15
[mkg52@wind /common/contrib/tutorials/sc15 ]$ ls *.pdf
tut106s3--Catalyst.pdf                          tut143s3--Debugging.pdf
tut111s3--MCDRAM.pdf                            tut145s3--LiveProgramming.pdf
tut112s3--PracticalFaultTolerance.pdf           tut148s3--ADIOS.pdf
tut115s3--Debugging.pdf                         tut150s3--AVX-512.pdf
tut116s3--AdvancedOpenMP.pdf                    tut157s3--NodeLevelPerfEng.pdf
tut120s3--Parallel-IO.pdf                       tut162s3--RADICAL.pdf
tut124s3--PortableHeterogeneousPrograms.pdf     tut167s3--Performance101.pdf
tut126s3--VisIt.pdf                             tut168s3--InfiniBand-HighSpeedEthernet-for-Dummies.pdf
tut130s3--FaultToleranceTheoryAndPractice.pdf   tut170s3--OpenFabrics.pdf
tut134s3--AutotuningWithPeriscope.pdf           tut171s3--BigDataOnClusters.pdf
tut137s3--MPI-plus-X.pdf                        tut173s3--PowerAwareHPC.pdf
[mkg52@wind /common/contrib/tutorials/sc15 ]$ ls *ADIOS*
tut148s3--ADIOS.pdf
[mkg52@wind /common/contrib/tutorials/sc15 ]$ cd /common/contrib/databases/genbank/
[mkg52@wind /common/contrib/databases/genbank ]$ ls ???[^0-9]*
chrontab.txt  human_hits.txt  mouse_hits.txt  my_update_blastdb.pl  nr.pal  nt.gz  nt.gz.md5  nt.nal
```

NORTHERN ARIZONA UNIVERSITY

Redirecting Input and Output

- System Defaults
  - stdin – Standard Input (File Descriptor of 0)
  - stdout – Standard Output (File Descriptor of 1)
  - stderr – Standard Error (File Descriptor of 2)
- > Redirects output to another file, overwriting if it exists
- >> Appends to a file
- 2>&1 Redirects error messages to standard output
- &> Redirects stdout, and stderr to a file
- | (vertical bar) Redirects or “pipes” output from one program to another’s input (*more on this later*)

Redirection Examples

- `ls > out.txt` - sends output from ls to “out.txt” file,
  `ls >> out.txt` - appends output from ls to “out.txt”
  `ls foo 2> error.txt` - sends only errors to ”error.txt”
- `ls foo &> out.txt` - writes output and errors to out.txt
- ls | wc –l - send output from ls to the wc (wordcount) program and counts lines

Processes

- `top` – Real-time view all running processes, akin to task manager in windows (for color, try htop)
- `kill <process id>` - Terminates a running process (if you are the owner of the process)
- `ps` - Shows current processes
- `ctrl-z` – sends a process to the background
- `ctrl-c` – ends a running process in the foreground
- `bg` – lists processes running in the background
- `fg` – brings background processes to the front

Guided Exercise – Processes Part 1

- Lets create a process. Run the command `sleep 500`. This will cause your cursor to disappear until the process is finished.
- Press `ctrl-z` to put the sleep process in the background, allowing you to regain control of your terminal.
- Type `jobs`. You should see your sleep program listed as a current background process in the stopped state.
- Type `bg`, your process will then begin running in the background
- Type `jobs`. You should see your sleep program listed as a current background process in the running state.
- Type `fg`. Your sleep process will now regain control of your terminal. Press ctrl-c to end the current process.
- Press Ctrl-z and bg to return your process to the background.

Guided Exercise – Processes Part 2

- Run the `top` program to view all processes currently running. Alternatively, you can run `ps` for a one-time snapshot, and top –u <userid>.
- Look for your sleep process in the list. Specifically, look at the first column labelled "PID". This means "process id". Take note of your sleep process's PID.
- Press `q` to quit top and get back to the terminal.
- Type `kill` <PID> where PID is your sleep process PID. This will end the sleep process.
- Press `jobs` again to view background processes. There should be none.

Lab 5 – Editing files, guided

- `cd ~/linux`
- `nano grepfile`

  Hello world! (press Enter to go to next line)

  The world is a big place. (press Enter)

  Save the file with cntrl-x , and yes
- nano commafile

  apples,oranges,oh,my

Lets do some simple data mining

- grep
  - Use grep to search a file and return all instances of a string of text
    - `grep world grepfile`
  - Notice, each line that has world, is returned, versus …
    - `grep place grepfile`
- Use grep recursively to look for instances of a word in nested directories and files
- From our "linux" directory, we can find all instances of the word "course" we wrote earlier…

```
[mkg52@wind ~/linux ]$ grep -r course *
is/the/best:of course!
is/fun:of course!
```

More data mining

- Remember the | symbol (pipe)?
    - We can redirect the output of one command to the input of another
- Let’s add a few lines to our grepfile so it looks like this:

```
Hello world!
The world is a big place
test 1
test 2
testing 3
```

- We can grep for test, and pipe the output to grep for the character “2”

```
[cbc@wind ~/linux ]$ grep test grepfile
test 1
test 2
testing 3
[cbc@wind ~/linux ]$ grep test grepfile |grep 2
test 2
```

More data mining

- With grep you can invert your selection as well

```
[cbc@wind ~/linux ]$ grep world grepfile
Hello world!
The world is a big place
```

- Vs

```
[cbc@wind ~/linux ]$ grep -v big grepfile
Hello world!
```

Let’s take a look at what happened… demo

Lab 6 – Pipes and Processes

- Start a new process by running sleep for 500 seconds (`sleep`)
- Put the process in the background (`ctrl-z`)
- Find the PID of your sleep process using `ps` and `grep`
- Kill your sleep process (`kill`)
- Verify your process is gone by running your previous `ps` and `grep` command
- Do a long recursive listing of your linux directory, filter the results so only filenames with the word "best" are returned, and send the output to a file called results.txt (`ls, grep, >`)
- List the contents of "results.txt" to verify the results (`cat`)
- Remove the file "results.txt" (`rm`)

Lab 6 – Pipes and Processes Solutions

```
mkg@ucc759:~/linux$ sleep 500
^Z
[1]+  Stopped                       sleep 500
mkg@ucc759:~/linux$ ps |grep sleep
  46396 pts/0    00:00:00 sleep
mkg@ucc759:~/linux$ kill 49396
[1]+  Terminated: 15                sleep 500
mkg@ucc759:~/linux$ ps |grep sleep
mkg@ucc759:~/linux$ ls -alR | grep best > results.txt
mkg@ucc759:~/linux$ cat results.txt
-rw-r--r--  1 mkg  staff    11B Oct 22 11:45 best
mkg@ucc759:~/linux$ rm results.txt
```

NORTHERN ARIZONA UNIVERSITY

Variables

- Linux makes use of variables in almost every program
- `env` command lists all current variables defined in your environment
- Set your own variables with the `set` command
- Set your own variables for sub processes opened from your shell with the `export` command
- Remove variables with the `unset` command
- Use existing variables by prepending them with `$`
- Standard is to use all CAPS for constants and exported variables
- `echo` command can display variable contents

Variables - Example

```
mkg@ucc759:~/linux$ MYVAR="this is my variable"
mkg@ucc759:~/linux$ echo $MYVAR
this is my variable
mkg@ucc759:~/linux$ unset MYVAR
mkg@ucc759:~/linux$ echo $MYVAR

mkg@ucc759:~/linux$
```

Controlling Your Environment

- ~/.bashrc file controls your bash environment settings
  - Note that due to the dot at the beginning, this is a hidden file
  - Contents of the file get run each time you start a terminal session
  - NOTE: This means that edits to the file will not go into effect until you "source" the script or log off and then log back in
- Allows you to set persistent variables for every session
  - `export MYVAR=10`
- Allows you to create short custom names or "aliases" for complex or long commands
  - `alias sq="squeue"`
- Now when you type `sq`, you will get the same results as typing `squeue`

  - `alias sj="scontrol show job"`

Command substitution

- It is possible to imbed command results within a command

- file `which cat`

```
[cbc@wind ~/linux ]$ file `which cat`
/bin/cat: ELF 64-bit LSB executable, x86-64, version 1 (SYSV) …
```

- ldd $(which cat)

```
[cbc@wind ~/linux ]$ ldd $(which cat)
linux-vdso.so.1 =>  (0x00007fff6d5d4000)
libc.so.6 => /lib64/libc.so.6 (0x0000003c20000000)
/lib64/ld-linux-x86-64.so.2 (0x0000003c1f800000)
```

NORTHERN ARIZONA UNIVERSITY

Loops

- Need to perform an action(s) over a list of items? Use a loop!
- Perform action over many items

```
[cbc@wind ~ ]$ for i in red blue green; do echo $i is a color; done
red is a color
blue is a color
green is a color
```

- Displaying consecutive numbers

```
[cbc@wind ~/linux ]$ for i in {1..3}; do echo $i; done     (also for a range: `seq 1 3`)
1
2
3
```

- Create ranges of letters or numbers using curly braces
  - {a..z} – loops over all LOWERCASE letters of the alphabet
  - {1..10} – loops over the numbers 1 through 10
- Use a custom variable, e.g. `i` and reference it with the `$` sign, just like any other variable

Loops Continued

```
[cbc@wind /common/contrib/databases/genbank ]$ for i in `ls nr.44.p*`; do du -h $i ;done
31M nr.44.phd
712K nr.44.phi
647M nr.44.phr
14M nr.44.pin
37M nr.44.pnd
148K nr.44.pni
6.9M nr.44.pog
14M nr.44.ppd
56K nr.44.ppi
169M nr.44.psd
3.9M nr.44.psi
650M nr.44.psq
```

Loops Continued

```
[cbc@wind ~]$ for i in `cat gems`; do echo $i ;done
```

```
for i in `ls *.seq`; do gzip $i; du –h $i;done
```

```
while true ; do sleep 30; squeue -u uid; done
```

```
for i in `seq 1 40`; do analysis.sh file_$i ; echo "done with file_$i" | mailx –s "file_$i done" a@b.com; done && echo "done with all processing” | mail -s “all done” a@b.com
```

```
echo "The following user id’s are logged in today: " && echo "userid  idle" && w | awk '{print $1 "\t" $5}' |grep -v USER|grep ^[^0-9] |sort -u -k1,1
```

Loops Continued

- While loops run until a condition is no longer true
- Good for running commands on a time interval
- Example: Do a long listing of your directory every second indefinitely

```
mkg@ucc759:~/linux$ while true; do ls -al; sleep 1; done
total 16
drwxr-xr-x   6 mkg  staff   204B Oct 27 15:48 ./
drwxr-xr-x+ 34 mkg  staff   1.1K Oct 22 12:22 ../
-rw-r--r--   1 mkg  staff    24B Oct 22 11:45 commafile
-rw-r--r--   1 mkg  staff    62B Oct 22 11:53 grepfile
drwxr-xr-x   5 mkg  staff   170B Oct 22 11:45 is/
drwxr-xr-x  12 mkg  staff   408B Oct 22 11:45 regex/
```

- Use `Ctrl-C` to end the process

Lab 7 – Variables and Loops

- Make a variable called GREETING that stores the string “Hello there” (`export`)
- Print your GREETING variable to the screen (`echo, $`)
- Create an alias called ”greet” that prints your GREETING variable to the screen (`alias`)
- What happens when you type greet at the command line?
- Write a for loop that calls your new greet alias 5 times (`for, {}, greet`)
- Write a while loop that indefinitely calls your greet alias and sleeps for 2 seconds (`while, greet, sleep`)

Lab 7 – Variables and Loops Solutions

```
mkg@ucc759:~/linux$ export GREETING="Hello there"
mkg@ucc759:~/linux$ echo $GREETING
Hello there
mkg@ucc759:~/linux$ alias greet="echo $GREETING"
mkg@ucc759:~/linux$ greet
Hello there
mkg@ucc759:~/linux$ for i in {1..5}; do greet; done
Hello there
Hello there
Hello there
Hello there
Hello there
mkg@ucc759:~/linux$ while true; do greet; sleep 2; done
Hello there
Hello there
Hello there
^C
mkg@ucc759:~/linux$
```

Links

- One thing you're sure to see and need at some point are links
- There are two types of links: *soft*, and *hard*
- We will focus on soft links for now, just know there are hard links too

Hard links

```
[cbc@wind ~/linux ]$ ls -li
total 0
12886566884 -rw-r--r-- 2 cbc clusteradm  0 Sep 22 10:14 file
   58536428 drwxr-xr-x 3 cbc clusteradm 38 Sep 21 15:58 is
12886566884 -rw-r--r-- 2 cbc clusteradm  0 Sep 22 10:14 link
```

- Adding a string of text to the file shows you that both file sizes update
- Also notice that the inode is the same for both files

```
[cbc@wind ~/linux ]$ ls -li
total 8
12886566884 -rw-r--r-- 2 cbc clusteradm 13 Sep 22 10:23 file
   58536428 drwxr-xr-x 3 cbc clusteradm 38 Sep 21 15:58 is
12886566884 -rw-r--r-- 2 cbc clusteradm 13 Sep 22 10:23 link
```

Soft links

- Soft links aka symbolic links are the most common type of links that you will use
  - `ln -s file symlink`
  - A soft link is a pointer to the file entry (*not the data*) in the filesystem

```
[cbc@wind ~/linux ]$ ls -li
total 8
-rw-r--r-- 2 cbc clusteradm 13 Sep 22 10:23 file
drwxr-xr-x 3 cbc clusteradm 38 Sep 21 15:58 is
lrwxrwxrwx 1 cbc clusteradm  4 Sep 22 10:15 symlink -> file
```

- Notice the inode is unique, and the size of the file is different. The size of the file is 4KB, the smallest size of file.

Simple Text manipulation

- sed is a command that allows you to perform manipulations to text without the need of a text editor and the patience to tediously do the edits yourself.
- To perform a text substitution:
  - `sed ‘s/oranges/apples/’ commafile (changes go to stdout)`
  - `sed –i ‘s/oranges/apples/’ commafile (in place)`
  - `sed ‘s/oranges/apples/’ commafile > output.txt (redirect to new file)`
- Notice that it replaced all instances of oranges with apples!
- Additionally, sed can take a stream of files as input and perform the same operations over all the files -- yet another advantage over your standard text editor.
  - `sed ‘s/oranges/apples’ commafile1 commafile2 commafile3`

Simple Text formatting

- awk is a command that allows you to filter/manipulate a stream of text. Great at manipulating rows and columns of data
- grab two columns from a stream of text

```
[cbc@wind ~/linux ]$ echo "image1 has value of .069" | awk '{print $1 " " $5}'
image1 .069
```

- Grab columns from a comma delimited stream

```
[cbc@wind ~/linux ]$ echo "image1,.069,str,1.78,k" | awk -F"," '{print $1 " " $4}'
image1 1.78
```

Lab 8 - Text Manipulation Exercise

- Change directory to linux (cd)
- In the file: grepfile, replace the word “big”, with “gigantic” (sed)
- In the file: grepfile, replace all instances of “world”, with “space” (sed)
- The commafile file has data that is comma delimited. Extract the data and produce a text file named results that has the data separated by space, or tabs. (awk, >)
- EXTRA: Use keywords “Buchnera, China” and search this file: /common/contrib/workshops/linux/assembly_summary.txt, extracting the 1st, 2nd, and 3rd fields and redirecting the output to buchnera_china. In that file, Change SAMN to be “NASM” instead. (grep, awk, >, sed)

Lab 8 Solutions

- sed -i ‘s/big/gigantic’ grepfile
- sed -i ‘s/world/space’ grepfile
- awk -F, '{print $1 " " $2 " " $3}' commafile > results
- grep Buchnera assembly_summary.txt|grep China |awk '{print $1 " " $2 " " $3}' > ~/linux/buchnera_china
- sed -i 's/SAMN/NASM/' ~/linux/buchnera_china

NORTHERN ARIZONA UNIVERSITY

Questions?

- Lots more to Linux
- Try this book out for more:
- http://linuxcommand.org/tlcl.php