

Introduction To Linux



Patrick Fitzhenry



High Performance Computing

- Linux is an operating system, somewhat like MS Windows or MacOSX.
- Linux is a particular variant of Unix. It is freely available as many distributions, that are *essentially* the same.
- I will assume that you have at least had some contact with computers, probably through the MS Windows environment.
- Linux has different commands from MS Windows and different ways of doing things.

Why Have This Class?

- Linux is the operating system used on the majority of eRSA's facilities.
- You will almost exclusively be presented with a text-based command line prompt at which you are expected to type commands ...**no point and click**.
- This means that you will need to know what commands to type at the prompt ...**help** is not one of them.
- Today, we will go through a basic set of commands that will allow you to start using the Linux operating system.

Courseware And Documentation

- There is a guided tutorial available on the internet at <http://www.ee.surrey.ac.uk/Teaching/Unix/>.
- What we do today comes directly from this course. I strongly recommend that you do the rest of the course and refer back to it when needed.
- All the Linux commands are fully documented in an online manual, often referred to as the “man pages”.
- The manual can be invoked from the command line by typing **man cmd**, where cmd is the command you wish to see the documentation for.
- You can even look up the manual’s documentation about itself by typing **man man**.

Looking Around

- The first thing you will want to do after login is to see what is around you.
- The **ls** command will give you a listing of the files in a directory.
- The **ls** command takes many options. What they are and what they do is described in detail in the manual (**man ls**).
- **ls -a** -lists 'all' files, including hidden files.
- **ls -l** -a 'long' listing of files, including permissions.

Looking Around...

- **Example: ls-al**

```
hydra.sapac.edu.au - PuTTY
[pfitzhen@hydra ~/training]$ ls -al
total 10080
drwxrwxr-x   3 pfitzhen pfitzhen   129 Jul 30 18:44 .
drwx----- 78 pfitzhen pfitzhen  16384 Jul 30 18:42 ..
-rw-rw-r--   1 pfitzhen pfitzhen   3379 Jul 30 18:42 dalton-corvus.txt
-rw-rw-r--   1 pfitzhen pfitzhen 10282907 Jul 30 18:42 mkldocumentation.tgz
-rwxr-xr-x   1 pfitzhen pfitzhen   557 Jul 30 18:44 mpitest.sh
-rw-----   1 pfitzhen pfitzhen   865 Jul 23 20:51 myjob.o337699
drwxrwxr-x   2 pfitzhen pfitzhen    6 Jul 30 18:44 sandbox
-rw-rw-r--   1 pfitzhen pfitzhen  1262 Jul 30 18:43 tryptophan.mol
[pfitzhen@hydra ~/training]$ █
```

- **Notes:**

- Command names are short and not always obvious.
- Options can be combined, eg. **ls-alrt**.
- Options often have a short (**-a**) and a long (**--all**) form.
- This is quite standard in the Linux world

File Permissions

- Files (and directories) have a set of permissions that dictate who can access them and what level of access they have
- There are permissions for 3 sets of users:
 - **o**- the owner of the file
 - **g**- users belonging to the same group as the owner of the file
 - **a**- all other users
- There are 3 types of access for each of these:
 - **r** read-can look at file contents
 - **w** write-can alter file contents
 - **x** execute-can run the file as a command

Pathnames

- Often, you will want to know where you are in the file system hierarchy, the file tree, or the path.
- **pwd**, print the working directory:
 - /home/admin/pfitzhen/training
- Pathnames are quoted as either the;
 - *absolute pathname*, the full path to a file from the root of the file system
 - *relative pathname*, the path to a file from here
- It is a common source of errors to mix these concepts up.
- `~` is special. It refers to the current user's home directory



Navigation

- **cd dir**, change directory to dir (go there).
 - dir must be a valid path, either absolute or relative, to a directory.
- We can change the existing file system structure by adding new directories and/or removing old directories.
 - **mkdir dir**, make a new directory called dir
 - **rmdir dir**, remove a directory called dir *
- Looking for files can be accomplished using the **find** command. Look it up in the man pages;
 - e.g. **find . -name "myfile*" -print**

Copying & Moving Files

- Locally –source and destination on the same machine:
 - **cp fname1 fname2**, copies fname1 to fname2
 - There will then be 2 identical files with different names.
 - **mv fname1 fname2**, renames fname1 to fname2
 - There will then be only 1 file, named fname2.
- Remotely –source and destination on different machines:
 - **scp**, secure copy
 - e.g. **scp myfile.txt pfitzhen@hydra:~/**
 - Can also use a program called **sftp**, secure file transfer protocol. Find out about it in the man pages.

Removing Files

- **rm fname** accomplishes this.
- This is a very powerful command. It is easy to accidentally delete whole sections of the file system and lose your data.
- **rm -r** recursively removes files **and directories**.
 - It can be used instead of **rmdir** as the directories are recursively emptied before removal.

File Editors - vi

- **vi** – a text editor supplied with all Linux distributions. It has cryptic commands and no prompts. Can be confusing at first.
- It has two *modes*, Command and Input. You need to switch between them to input and save data.
- A good quick tutorial can be found at http://math.la.asu.edu/vi_tutorial/vi3.html

File Editors - emacs

- **emacs** – an XWindows based text editor with a more WYSIWYG feel, drop-down menus ...and finally ...**point and click**.
- If you want to run emacs from a remote session on a MS Windows based computer, you will need an XServer running locally.
|
- A good tutorial guide for emacs can be found at <http://www.linuxhelp.net/guides/emacs>.

Displaying File Contents

- **cat** - scrolls file contents to the screen. For long files, you only see the end of the file.
- **more** - scrolls file contents to the screen 1 page at a time
 - Spacebar scrolls a page at a time.
 - Type **q** to quit and return to command line.
 - Older distributions can only scroll forward.
- **less** - scrolls file contents to the screen 1 page at a time
 - Like more but with extra functionality.
 - Type **b** to scroll backwards 1 page at a time.
- **head** - displays beginning of a file on the screen
 - **-n** option allows control of number of lines to show.
- **tail** - displays end of a file on the screen
 - **-f** option allows interactive updating.

Searching File Contents

- **grep pattern fname** – searches the file fname for pattern
- and displays each line where the pattern occurs.
- This is a very useful and adaptable command. Read the
- man page.
- You will come across the concept of regular expressions
- when investigating this command. These are very
- powerful but may be confusing at first.
- They are worth persevering with if you are going to do a
- lot of computation in your professional life.

Redirecting Input & Output

- You can redirect *standard input* (usually the keyboard) and *standard output* (usually the screen).
- The `>` and `<` symbols accomplish this.
- This is especially useful if your program requires input and is running on a machine with no keyboard or screen in the middle of the night somewhere –as is common for jobs running on eRSA's facilities.
- Redirecting standard output (stdout) and standard error (stderr) to a file can allow you to monitor your job's progress.
 - **cmd < infile.dat** - redirect stdin to cmd to be from file infile.dat
 - **cmd >& outfile.log** - redirect stdout and stderr to file outfile.log

Pipes

- Sometimes, you may wish to make the output of one command become the input to another command, i.e. pipe one command through another.
- The | symbol accomplishes this.
 - e.g. **cat /etc/passwd | grep pfitzhen**
 - pipes the output of cat to grep
 - This is a pretty useless example. Why ?
- A more useful example may be something like:
 - **tail -f joboutput.log | grep -i "energy = "**
 - to monitor an output file to check on energy calculations after each iteration.

Wildcards

- We have been using wildcards throughout this presentation and I have been assuming that you are already familiar with them.
- * - as a wildcard, the asterisk stands for "zero or more characters".
- ? - as a wildcard, the question mark stands for "exactly one character".
 - eg `ls *.log`
`ls test39?.log`
`grep -rni debug src/*`

Changing File Permissions

- We noted earlier that all files and directories have a set of permissions attached to them.
- If you are the owner of a file you can change the permissions of a file to allow or prevent access by other users.
- The **chmod** command accomplishes this. Look up its details in the man pages.
 - e.g. **chmod 755 myprog**
 - allows all access to the owner and read and execute right to everyone else.
- An alternate notation is **chmod a+rx myprog**.



Archiving And Compressing Data

- After you have generated data you will probably wish to archive it.
- **tar** - an archiving program that will pack up a portion of the file system into a single file.
- **gzip/gunzip** – a file compression program.
- These two programs can be combined either with pipes or through the use of options to the tar program.
 - e.g. **tar czvf data.tgz mydatadir/**
 - creates a compressed archive file data.tgz containing all files and directories under mydatadir, preserving the file hierarchy.

Processes

- Sometimes you may wish to see what the machine is currently up to.
- **top** - list the top 20 or so runnable processes interactively.
 - Use <Ctrl>-C to exit
- **ps** - list the status of processes on the machine.
 - e.g. **ps - aux**

Shells

- The interface you type commands into on a Linux system is known as the *shell*. It is a program that continuously takes Linux commands and executes them as you type them at the command prompt.
- There are a variety of different shells available. **bash** and **csh** are the 2 most common ones and they take commands slightly differently. By default, at eRSA you are presented with csh.

Shellscripts

- A *shellscript* is a text file, made executable, that just contains a series of Linux commands.
- When a shellscript is run it executes each of the commands in order until it reaches the end of the file.
- As with all programming languages there are selection constructs (if-then-else) and looping constructs (foreach-end) available to control the flow of execution within the shellscript.

Shellscripts...

- Example
- Create the following in a file called bigfiles

```
#!/bin/tcsh  
# Usage : bigfiles NNN where NNN is size in 512-byte blocks  
@ argc= $#  
if ($argc!= 1) then  
echo "$0 must have a filesize in 512-byte blocks as its argument"  
exit(1)  
endif  
@ fsize= $1  
set today = `date "+%d-%m-%Y"`  
foreach f ( `find ~ -size +$fsize-print` )  
echo "$today -$f has filesize greater than $fsizekbytes"  
end
```

Shellscripts...

- Save the file bigfiles and make it universally executable.
- What will bigfiles do ??
- Run it and see ...

Summary

- This course covered only the bare essentials to get you working on a Linux platform.
- There are many useful and cheap references that will give you more help if needed.
- Google is a good place to start searching for solutions to computer problems.
- If you are stuck and “computer says no”, contact someone on the eRSA helpdesk at helpdesk@eresearchsa.edu.au or on by phone at 8303 8263