LEHIGH UNIVERSITY

Introduction to Linux

File Permission, Process Management & Editors

Alexander B. Pacheco
LTS Research Computing
September 22, 2015

Outline

1. Basic *nix Utilities
2. Redirection
3. File Permissions
4. Process Management
5. Editors

Basic *nix Utilities

Filename Completion

- Filename or Tab completion is a default feature in `bash` and `tcsh`.
- It allows to a user to automatically complete the file, directory or command name you are typing upto the next unique characters using the TAB key.
- Example: Your home directory contains directories `Desktop`, `Documents` and `Downloads`.

  If you enter the command `ls D` ↹ , you will be prompted with above the three directory names.

```
[user@localhost ~]$ ls D ↹
Desktop/ Documents/ Downloads/
[user@localhost ~]$ ls Do ↹
Documents/ Downloads/
[user@localhost ~]$ ls Do
```

Wildcards

- *nix shells have the ability to refer to more than one file by name using special characters called Wildcards.
- Wildcards can be used with *nix utilities such as ls, cp, mv, rm, tar and g(un)zip.

?	match a single character
*	match zero or more characters
[ ]	match list of characters in the list specified
[!]	match characters not in the list specified

- Examples:
  1. `ls */*`
     list contents of all subdirectories
  2. `cp [a-z]* lower/`
     copy all files with names that begin with lowercase letters to a directory called lower
  3. `cp [!a-z]* upper_digit/`
     copy all files with names that do not begin with lowercase letters to a directory called lower

How to Login to Remote Systems?

- Most Linux/UNIX systems allow secure shell connections from other systems.

e.g. You need to login using `ssh` to the LTS HPC clusters.

- Usage: `ssh <username>@<remote host>`
- Example: `ssh alp514@polaris.cc.lehigh.edu`
- If your local machine is a UNIX-like system i.e. Linux, Mac OSX, BSD, AIX, Solaris etc and your username on the local machine is the same as that of the remote machine, then

  you can omit the `<username>@` part of the argument.

  i.e. `ssh <remote host>`
- If the remote machine is listening to ssh connections on a non default port (i.e. different from port 22) add `-p <port number>` option

i.e. `ssh -p <port number> <user>@<remote host>`

- If you need to forward the display of an application from the remote system to your local system, add the `-X` option to `ssh`

  Example: `ssh -X alp514@ssh.cc.lehigh.edu`

File Transfer between two systems I

- `scp` is a command to copy files/directories between two *nix hosts over the SSH protocol.
- Usage:

```
scp <options> <user>@<host>:/path/to/source/file \
    <user>@<host>:/path/to/destination/file/or/directory
```

e.g. You want to copy files between Polaris Cluster and your Linux Desktop/Laptop

```
scp alp514@polaris.cc.lehigh.edu:/home/alp514/octave-tutorial.tar.gz .
scp -r Public apacheco@polaris.cc.lehigh.edu:~/
```

- You can omit the `<user>@` part of the argument if the username is the same on both systems.
- You can omit the `<user>@<host>:` for your local machine.
- Common options are `-r` and `-p`, same meaning as `cp`.
- add `-P <port number>` option for non default ports.

File Transfer between two systems II

- `rsync` is another utility that can be used to copy files locally and remotely.
- Usage: `rsync <option> <source> <destination>`
- It is famous for its delta-transfer algorithm

i.e. sending only the differences between the source files and the existing files in the destination.

- Rsync is widely used for backups and mirroring and as an improved copy command for everyday use.
- Common options:

  `-a`: archive mode

  `-r`: recurse into directories

  `-v`: increase verbosity

  `-z`: compress file data during the transfer

  `-u`: skip files that are newer on the receiver

  `-t`: preserve modification times

  `-n`: dry-run, perform a trial run with no changes made
- Example: `rsync -avtzu corona.cc.lehigh.edu:~/* .`
- If you are a user on National Supercomputing resource such as XSEDE, NERSC, OSG, etc, there are other transfer tools such as globus toolkit (gridftp) and bbcp which provide higher bandwidth and parallel file transfers.

Compressing and Archiving Files I

- Quite often you need to compress and uncompress files to reduce storage usage or bandwidth while transferring files.
- *nix systems have built-in utilities to compress/uncompress files

Compress

```
gzip, zip, bzip2
gzip README Enter
```

Uncompress

```
gunzip, unzip, bunzip2
gunzip README.gz Enter
```

- Gzipped files have an extension `.gz,.z` or `.Z`
- zipped files have an extension `.Zip` or `.zip`
- Bzipped files have an extension `.bz2, .bz`
- To compress/uncompress files recursively, use the `-r` option.
- To overwrite files while compressing/uncompressing, use the `-f` option.

Compressing and Archiving Files II

- *nix provides the `tar` package to create and manipulate streaming archive of files.
- Usage: `tar <options> <file> <patterns>`

  `file` is the name of the tar archive file, usually with extension `.tar`

  `patterns` are pathnames for files/directories being archived
- Common options

  `-c`: create an archive file

  `-x`: extract to disk from archive

  `-z`: filter the archive through gzip (adds/requires extension .gz)

  `-j`: filter the archive through bzip2 (adds/requires extension .bz2)

  `-t`: list contents of archive

  `-v`: verbosely list files processed

e.g. `tar -cvzf myhome.tar.gz ${HOME}/*`

- This becomes useful for creating a backup of your files and directories that you can store at some storage facility e.g. external disk

Redirection

I/O Redirection

- There are three file descriptors for I/O streams

STDIN	: Standard Input
STDOUT	: Standard Output
STDERR	: Standard Error

- 1 represents STDOUT and 2 represents STDERR
- I/O redirection allows users to connect applications

<	: connects a file to STDIN of an application
>	: connects STDOUT of an application to a file
>>	: connects STDOUT of an application by appending to a file
|	: connects the STDOUT of an application to STDIN of another application.

- Examples:
  1. write STDOUT to file: `ls -l > ls-l.out`
  2. write STDERR to file: `ls -l 2> ls-l.err`
  3. write STDOUT to STDERR: `ls -l 1>&2`
  4. write STDERR to STDOUT: `ls -l 2>&1`
  5. send STDOUT as STDIN: `ls -l | wc -l`

File Permissions

File Permissions I

- Since *NIX OS's are designed for multi user environment, it is necessary to restrict access of files to other users on the system.
- In *NIX OS's, you have three types of file permissions
  1. read (r)
  2. write (w)
  3. execute (x)
- for three types of users
  1. user (u)
  2. group (g)
  3. world (o) i.e. everyone else who has access to the system

File Permissions II

```
[user@localhost ~]$ ls -l
total 44
drwxr-xr-x. 2 user user 4096 Jan 28  2013 Desktop
drwxr-xr-x. 2 user user 4096 Jan 28  2013 Documents
drwxr-xr-x. 2 user user 4096 Jan 28  2013 Downloads
-rwxr-xr-x. 1 user user   32 Sep 11 11:57 hello
drwxr-xr-x. 2 user user 4096 Jan 28  2013 Music
drwxr-xr-x. 2 user user 4096 Jan 28  2013 Pictures
drwxr-xr-x. 2 user user 4096 Jan 28  2013 Public
-rw-rw-r--. 1 user user 3047 Sep 11 11:48 README
drwxr-xr-x. 1 root root 4216 Jan 22 16:17 Shared
drwxr-xr-x. 2 user user 4096 Jan 28  2013 Templates
lrwxrwxrwx. 1 user user    5 Jan 23 08:17 test -> hello
drwxr-xr-x. 2 user user 4096 Jan 28  2013 Videos
[user@localhost ~]$
```

- The first character signifies the type of the file

  `d` for directory

  `l` for symbolic link

  `-` for normal file

- The next three characters of first triad signifies what the owner can do
- The second triad signifies what group member can do

File Permissions III

- The third triad signifies what everyone else can do

$$d\,\underbrace{rwx}_{u}\,\overbrace{r-x}^{g}\,\underbrace{r-x}_{o}$$

- Read carries a weight of 4
- Write carries a weight of 2
- Execute carries a weight of 1
- The weights are added to give a value of 7 (rwx), 6(rw), 5(rx) or 3(wx) permissions.
- `chmod` is a *NIX command to change permissions on a file

  Usage: `chmod <option> <permissions> <file or directory name>`
- To give user rwx, group rx and world x permission, the command is

  `chmod 751 filename`

File Permissions IV

- Instead of using numerical permissions you can also use symbolic mode

u/g/o or a	user/group/world or all i.e. ugo
+/-	Add/remove permission
r/w/x	read/write/execute

- Give everyone execute permission:

```
chmod a+x hello.sh
chmod ugo+x hello.sh
```

- Remove group and world read & write permission:

```
chmod go-rw hello.sh
```

- To change permissions recursively in a directory, use the option `-R` (can also be used in the following two commands)

```
chmod -R 755 ${HOME}/*
```

  What is the permission on ${HOME}?

File Permissions V

- The `chgrp` command is used to change the group ownership between two groups that you are a member of.

  Usage: `chgrp <option> <new group> <file or directory name>`

- You can use the `chgrp` command to change the ownership of your files from the `users` group to `abc` group.

  Example: `chgrp -R abc collaborative-work-dir`

- The `chown` command is used to change the owner of a file.
- `chown` can only be executed by the superuser, to prevent users simply changing ownership of files that aren't theirs to access.

  Usage: `chown <new owner>[:<group name>] <file or directory name>`

Process Management

Processes and Jobs I

- A process is an executing program identified by a unique PID
- ★ To see information about your running processes and their PID and status,

  `ps` Enter
- A process may be in foreground, background or be suspended.
- Processes running in foreground, the command prompt is not returned until the current process has finished executing.
- If a job takes a long time to run, put the job in background in order to obtain the command prompt back to do some other useful work
- There are two ways to send a job into the background:
  1. Add an ampersand `&` to the end of your command to send it into background directly.

     `firefox &` Enter
  2. First suspend the job using Ctrl `Z` and then type `bg` at the command prompt.
  3. If you type `fg` then the job will run in foreground and you will lose the command prompt.

Processes and Jobs II

- When a process is running, background or suspended, it will be entered onto a list along with a job number (not PID)

  `jobs` Enter

- To restart a suspended job in foreground or background, type

  `fg %jobnumber` where `jobnumber` is a number greater than 1, or,

  `bg %jobnumber`

- To kill or terminate a process:

  1. Job running in foreground: enter Ctrl `C`
  2. Job whose PID you know
     `kill PID` Enter
  3. Job whose `jobnumber` you know (from `jobs` command)
     `kill %jobnumber` Enter

- The `kill` command can take options specific to UNIX signals
- The most common option is `-9` for the `SIGKILL` signal
- `pstree`: display a tree of processes
- `pkill`: kill process by its name, user name, group name, terminal, UID, EUID, and GID.

Editors

File Editing

- The two most commonly used editors on Linux/Unix systems are:
  1. `vi` or `vim` (vi improved)
  2. `emacs`
- `vi/vim` is installed by default on Linux/Unix systems and has only a command line interface (CLI).
- `emacs` has both a CLI and a graphical user interface (GUI).
- If `emacs` GUI is installed then use `emacs -nw` to open file in console.
- Other editors that you may come across on *nix systems

  `kate`: default editor for KDE.

  `gedit`: default text editor for GNOME desktop environment.

  `gvim`: GUI version of `vim`

  `pico`: console based plain text editor

  `nano`: GNU.org clone of `pico`

  `kwrite`: editor by KDE.

Editor Cheatsheets I

- **vi/vim** and **emacs** are the two most popular *nix file editors.
- Which one to use is up to you.
- **vi/vim** has two modes:
  1. Editing mode
  2. Command mode
- **emacs** has only one mode as in any editor that you use.

Insert/Appending Text	vi
insert at cursor	`i`
insert at beginning of line	`I`
append after cursor	`a`
append at end of line	`A`
newline after cursor in insert mode	`o`
newline before cursor in insert mode	`O`
append at end of line	`ea`
exit insert mode	`ESC`

Editor Cheatsheets II

Cursor Movement	vi	emacs
move left	`h`	`C-b`
move down	`j`	`C-n`
move up	`k`	`C-p`
move right	`l`	`C-f`
jump to beginning of line	`^`	`C-a`
jump to end of line	`$`	`C-e`
goto line **n**	`nG`	`M-x goto-line` ↵ `n`
goto top of file	`1G`	`M-<`
goto end of file	`G`	`M->`
move one page up	`C-u`	`M-v`
move one page down	`C-d`	`C-v`

C : Control Key

M : Meta or ESCAPE (ESC) Key

↵ : Enter Key

Editor Cheatsheets III

File Manipulation	vi	emacs
save file	`:w`	`C-x C-s`
save file and exit	`:wq`, `ZZ`	
quit	`:q`	`C-x C-c`
quit without saving	`:q!`	
delete a line	`dd`	`C-a C-k`
delete *n* lines	*n*`dd`	`C-a M-`*n* `C-k`
paste deleted line after cursor	`p`	`C-y`
paste before cursor	`P`	
undo edit	`u`	`C-_`
delete from cursor to end of line	`D`	`C-k`
search forward for *patt*	`\`*patt*	`C-s` *patt*
search backward for *patt*	`?`*patt*	`C-r` *patt*
search again forward (backward)	`n`	`C-s(r)`

Editor Cheatsheets IV

File Manipulation (contd)	vi	emacs
replace a character	`r`	
join next line to current	`J`	
change a line	`cc`	
change a word	`cw`	
change to end of line	`c$`	
delete a character	`x`	`C-d`
delete a word	`dw`	`M-d`
edit/open file *file*	`:e file`	`C-x C-f` *file*
insert file *file*	`:r file`	`C-x i` *file*
split window horizontally	`:split` or `C-ws`	`C-x 2`
split window vertically	`:vsplit` or `C-wv`	`C-x 3`
switch windows	`C-ww`	`C-x o`

Editor Cheatsheets V

- Do a google search for more detailed cheatsheets

vi	`https://www.google.com/search?q=vi+cheatsheet`
emacs	`https://www.google.com/search?q=emacs+cheatsheet`

More on the **set -o** command

- The **set -o** command can be used to change the command line editor mode among other things (Do **man set** Enter to find out more)
  1. **set -o emacs**: emacs style in-line editor for command entry, this is the default
  2. **set -o vi**: vi style in-line editor for command entry.

Start Up Scripts

- When you login to a *NIX computer, shell scripts are automatically loaded depending on your default **shell**
- **sh,ksh**
  1. `/etc/profile`
  2. `$HOME/.profile`
- **bash**
  1. `/etc/profile`, login terminal only
  2. `/etc/bashrc` or `/etc/bash/bashrc`
  3. `$HOME/.bash_profile`, login terminal only
  4. `$HOME/.bashrc`
- **csh,tcsh**
  1. `/etc/csh.cshrc`
  2. `$HOME/.tcshrc`
  3. `$HOME/.cshrc` if .tcshrc is not present
- The `.bashrc, .tcshrc, .cshrc, .bash_profile` are script files where users can define their own aliases, environment variables, modify paths etc.
- e.g. the **alias** command covered earlier can be put in one of these script files depending on your **shell**

Examples I

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
alias c="clear"
alias rm="/bin/rm -i"
alias psu="ps -u apacheco"
alias em="emacs -nw"
alias ll="ls -lF"
alias la="ls -al"
export PATH=/home/apacheco/bin:${PATH}
export g09root=/home/apacheco/Software/Gaussian09
export GAUSS_SCRDIR=/home/apacheco/Software/scratch
source $g09root/g09/bsd/g09.profile

export TEXINPUTS=.:/usr/share/texmf//:/home/apacheco/LaTeX//:${TEXINPUTS}
export BIBINPUTS=.:/home/apacheco/TeX//:${BIBINPUTS}
```

Examples II

```
# .tcshrc

# User specific aliases and functions
alias c clear
alias rm "/bin/rm -i"
alias psu "ps -u apacheco"
alias em "emacs -nw"
alias ll "ls -lF"
alias la "ls -al"
setenv PATH "/home/apacheco/bin:${PATH}"
setenv g09root "/home/apacheco/Software/Gaussian09"
setenv GAUSS_SCRDIR "/home/apacheco/Software/scratch"
source $g09root/g09/bsd/g09.login

setenv TEXINPUTS ".:/usr/share/texmf//:/home/apacheco/LaTeX//:${TEXINPUTS}"
setenv BIBINPUTS ".:/home/apacheco/TeX//:${BIBINPUTS}"
```

What is a Scripting Language?

- A **scripting language** or **script language** is a *programming language* that supports the writing of **scripts**.
- **Scripting Languages** provide a higher level of abstraction than standard programming languages.
- Compared to programming languages, scripting languages do not distinguish between data types: integers, real values, strings, etc.
- Scripting Languages tend to be good for automating the execution of other programs.
    - analyzing data
    - running daily backups
- They are also good for writing a program that is going to be used only once and then discarded.

What is a script?

- A **script** is a program written for a software environment that automate the execution of tasks which could alternatively be executed one-by-one by a human operator.
- The majority of script programs are “quick and dirty”, where the main goal is to get the program written quickly.

Writing your first script

1. Write a script
   - A shell script is a file that contains ASCII text.
   - Create a file, **hello.sh** with the following lines

```
#!/bin/bash
# My First Script
echo "Hello World!"
```

2. Set permissions

```
apacheco@apacheco:~/Tutorials/BASH/scripts> chmod 755 hello.sh
```

3. Execute the script

```
apacheco@apacheco:~/Tutorials/BASH/scripts> ./hello.sh
Hello World!
```

Description of the script

- My First Script

```
#!/bin/bash
# My First Script
echo "Hello World!"
```

- The first line is called the "SheBang" line. It tells the OS which interpreter to use. In the current example, bash
- Other options are:

sh	: `#!/bin/sh`
ksh	: `#!/bin/ksh`
csh	: `#!/bin/csh`
tcsh	: `#!/bin/tcsh`

- The second line is a comment. All comments begin with "#".
- The third line tells the OS to print "Hello World!" to the screen.

Special Characters

#:	starts a comment.
$:	indicates the name of a variable.
\\:	escape character to display next character literally.
{ }:	used to enclose name of variable.
;	Command separator [semicolon]. Permits putting two or more commands on the same line.
;;	Terminator in a case option [double semicolon].
.	"dot" command [period]. Equivalent to source. This is a bash builtin.
$?	exit status variable.
$$	process ID variable.
[ ]	test expression
[[ ]]	test expression, more flexible than [ ]
$[ ], (( ))	integer expansion.
||, &&, !	Logical OR, AND and NOT

Quotation

- Double Quotation `" "`
  - Enclosed string is expanded ("$", "/" and "‘")
  - Example: `echo "$myvar"` prints the value of `myvar`
- Single Quotation `’ ’`
  - Enclosed string is read literally
  - Example: `echo ’$myvar’` prints `$myvar`
- Back Quotation `‘ ‘`
  - Enclosed string is executed as a command
  - Example: `echo ‘pwd‘` prints the output of the `pwd` command i.e. print the current working directory

Exercises I

- Login to a Linux machine and open a terminal
- Enter the following commands or carry out operations asked for.
- Understand what you are doing and ask for help if unsure. Some commands are incorrect or will fail, enter the correct

1. `echo hello world` Enter
2. `pwd` Enter
3. `whoami` Enter
4. `cd /tmp` Enter
5. `cd -` Enter
6. `mkdir test/testagain` Enter
7. `cd test/testagain` Enter
8. `touch file` Enter
9. Go back to your home directory.
10. Which shell are you using?
11. Review the commands you have just entered.
12. create an alias for removing files which prompt for confirmation and delete the file that you created.
13. From your home directory get a list of files and directory in long format in reverse order with file sizes listed in human readable format.

Exercises II

14. Find out the location of vi, emacs, firefox, google-chrome, thunderbird, latex, pdflatex, gnuplot, python, perl and matlab.
15. Change the permission of the testagain directory to be world writable.
16. open a few applications of choice in foreground one by one and then suspend them,
17. get a list of suspended jobsr,
18. foreground job 1 and close it,
19. background job 2,
20. kill job 3,
21. put job 2 in foreground and close it,
22. check if you still have any jobs running.

Exercises III

1. Exercise courtesy `http://www.doc.ic.ac.uk/~wjk/UnixIntro/Exercise6.html`
2. Copy the file mole.txt
   `wget http://www.doc.ic.ac.uk/~wjk/UnixIntro/mole.txt`
3. Go to the end of the document and type in the following paragraph:
   `Joined the library. Got Care of the Skin, Origin of the Species, and a book by a woman my mother is always going on about. It is called Pride and Prejudice, by a woman called Jane Austen. I could tell the librarian was impressed. Perhaps she is an intellectual like me. She didn't look at my spot, so perhaps it is getting smaller.`
4. Correct the three spelling errors in the first three lines of the first paragraph (one error per line) and remove the extra "Geography" in the 3rd line of the first paragraph.
5. Add the words "About time!" to the end of the second paragraph.
6. Delete the sentence "Time flies like an arrow but fruit flies like a banana" and re-form the paragraph.
7. Replace all occurrences of "is" with "was".
8. Swap the two paragraphs.
9. Save the file and quit.

Exercises IV

Wednesday January 14th
Joined the library. Got Care of the Skin, Origin of the Species, and a book by a woman my mother was always going on about. It was called Pride and Prejudice, by a woman called Jane Austen. I could tell the librarian was impressed. Perhaps she was an intellectual like me. She didn't look at my spot, so perhaps it was getting smaller.
None of the teachers at school have noticed that I am an intellectual. They will be sorry when I am famous. There was a new girl in our class. She sits next to me in Geography. She was all right. Her name was Pandora, but she likes being called "Box". Don't ask me why. I might fall in love with her. It's time I fell in love, after all I am 13 3/4 years old. About time!