Linux Fundamentals

For the Discovery cluster

Boris Yazlovitsky
Research Computing
Northeastern University
Logging in

1. Need terminal window

Windows:

- Putty: needs putty, winscp or filezilla, xming
- mobaXterm
- Oracle Virtual Box: needs a lot of memory, 50GB+ disk to be useful
- Cygwin: needs cygwin, xming

MacOS/Linux: comes with the OS
Navigation: It’s a tree!

Everything in Linux is a tree

- Filesystem
- Process table
Discovery filesystem

```
/scratch
  /
  /usr
    man  bin  local
    /
    /bin
    stata  gaussian  gcc
    /
    bin
    14  15
    6.4.0  7.2.0
    8.1.0

/home
  you
  /
  /bin
  stata
  gaussian
  gcc
  sas
  cuda
  berkleygw
  amber
  XXX

shared
  centos7
  v1
  v2
  vn
```
Filesystem navigation commands

**cd** (change directory)

**pwd** (print working directory)

**pushd** put current directory in dir stack

**popd** go to the last directory pushd’ed

**mkdir newdir** (make directory “newdir”)

**mkdir -p newdir/subdir1/subdir2/subdir3** (make path newdir/subdir1/subdir2/subdir3)
Filesystem navigation: viewing

ls - list files

- Options: `ls -[lart1d]` most common options
  - `-l`: long listing
  - `-a`: all files (including hidden files, (ones that begin with a .))
  - `-r`: show in reverse order (default sorting order is alphanumeric)
  - `-t`: sort by time modified
  - `-l`: show all in 1 column
  - `-d`: show directory names but not their content
Viewing file content

cat: print content of file to the stdout (more on streams later)
more: print content of file to the screen but pause between pages
less: like more but with more options
vim: open file in an editor
File and directory manipulation

copy: `cp [-rpiv]

- files: `cp file1 file2
- Directories: `cp -r dir1 dir2

move/rename: `mv oldpath/oldname newpath/newname

Remove: `rm -[irf] / `rmdir -[pmv]

- Files: `rm file1 file2 file3 .... filek filename
- Directories: `rmdir dir1 dir2 dir3 ... dirk dirn
Other file and directory utilities

To find a pattern in a file:

To search in a single file: `grep pattern file`

To search in entire directory: `grep pattern *`

To search in entire directory in files matching a pattern:

`grep search_pattern *filename_pattern` *

To find the lines containing the word dog in all files which end with .bar:

`grep dog *.bar`
More file utilities: heads & tails

Show the first N lines of a file:

```
head -N filename
```

Show the last N lines of a file:

```
tail -N filename
```

Follow a file as it is being written to:

```
tail -f filename
```
Find a file in your directory tree

I want to find all the files which end with .err:

`find . -type f -name '*.err'`

I want to find all the files which end with .err and have the word 'h2o' in them:

`find . -type f -name '*.err' -exec grep -H h2o {} \;`
grep is a very powerful utility and has many options

grep --help will show all options

Most common are:

- `l` just show list of files that match a pattern
- `i` ignore case
- `n` show line numbers where the pattern is found
- `h` do not show filename in front of the line where the pattern was found
- `v pattern` : show all lines that do not have the specified pattern
Other file and string utilities

Sed: The following will replace every instance of the word sun with moon and print to stdout

cat file | sed -e 's/sun/moon/g'

Cut: Show the 1st, 3rd, and 4th fields of every line of a file with data delimited by a ‘|’ s.a. a .csv

cat file.csv | cut -d'|' -f1,3,4

Awk: easy example: show the 3rd word of every line:

cat file | awk '{print $3}'}
Permissions and ownership

Long listing (ls -l) shows ownership and permissions on a file or a directory

```
drwxr-xr-x 2 bojin9 users 4096 Oct 16 18:59 d
```

Type, user perms, group perms, others’ perms, owner, group, size, date modified

```
-rw-r--r-- 1 bojin9 users   0 Oct 16 19:02 f
```
Permissions

permissions/owners

Change permissions: chmod [-R] mode file

- Mode: permissions: r-read, w-write, x-execute for owner (user) group, others
- Chmod ugo+rwx file: makes file readable, writable and executable by owner, their, group and everyone else
- Chmod u=rwx,g=rx,o=x makes the file or directory readable, writable, executable by user, readable & executable by group and executable only by others.
Permissions continued

Octal method: x := 1, w:=2, r:=4

777 => rwxrwxrwx

755 => rwxr-xr-x

644 => rw-r--r--

750 => rwxr-x--
ownership

chown user file

Usually system will not allow for regular users
Statistics about a file or directory

`ls -l` shows easy to read but limited information

`stat` shows not-so-easy to read but more complete information, taken from the inode itself

`stat f`

File: 'f'

Size: 22          Blocks: 0          IO Block: 16777216 regular file

Device: 32h/50d  Inode: 306520281   Links: 1

Access: (0755/-rwxr-xr-x)  Uid: (50505/ bojin9)  Gid: (  100/   users)


Modify: 2018-10-16 19:37:06.546496000 -0400
archives

Sometimes it may be necessary to pack up an entire directory into a single file, for archiving, reducing the number of files in the filesystem, or easy transfer.

Create an archive from directory foo:

```
tar cf foo.tar ./foo
```

c: create, f: name of file to crate

Unpack an archive:

```
tar xvf foo.tar
```

x: xtract, v: verbose, f: filename of the archive
archives

Often tar archives are compressed to form what is known in the unix/linux world as a “tarball”

To create or unpack a tarball, use the z option

```
tar czf foo.tgz ./foo

tar zxf foo.tgz
```

`zip` creates archives compatible with windows archives
Compressing is not limited to tar files. Any file can be compressed using several compression utilities. Difference is in the compression algorithm:

- gzip
- bzip2
- zip
processes

Linux processes also follow a tree-like structure

```bash
pstree -aup
```

```bash
ps -eaf
```

top
Process control

To stop a process: `kill pid` to kill it completely: `kill -9 pid`

To see which files a process has open: `lsotf -p pid`

To watch what system calls a process calls while running: `strace -f -p pid`

`-f`: to follow all the child processes
Pipes: Combining processes

Sometimes it is necessary to take the output of one process and run another process on it:

Task: I want to see the hostnames, and jobids of all my SLURM jobs that failed after Oct. 10:

```
sacct -p -u bojin9 --starttime=2018-10-10 --format=jobid,nodelist,state | grep -i failed | cut -d'|' -f1,2
```

Note 3 processes are involved: `sacct`, `grep`, and `cut`. Each sends its stdout to the next processes’ stdin.
I want to follow the output of my running batch job but print only lines which match a pattern

Output file job.out

tail -f jobid.out  |  grep pattern
streams

A process or a command that produces output will print it to either a log file, stderr (standard error) or stdout (standard out) to catch the output into files file descriptors have to be used.

Stdout is file descriptor 1

Stderr is file descriptor 2

`command 2>out.err 1>out` will put the errors in file ‘out.err’ and regular output in ‘out’

`command >out-all 2>&1` combines stderr and stdout and writes to out-all
Documentation

Almost all utilities have the --help option which shows what the options are.

`man utility` shows the manual page for the utility, or a system call or any concept.

A fun game is to `strace` a running process and google or read the man page on the system calls.

REMEMBER TO EXPERIMENT