Intermediate O2

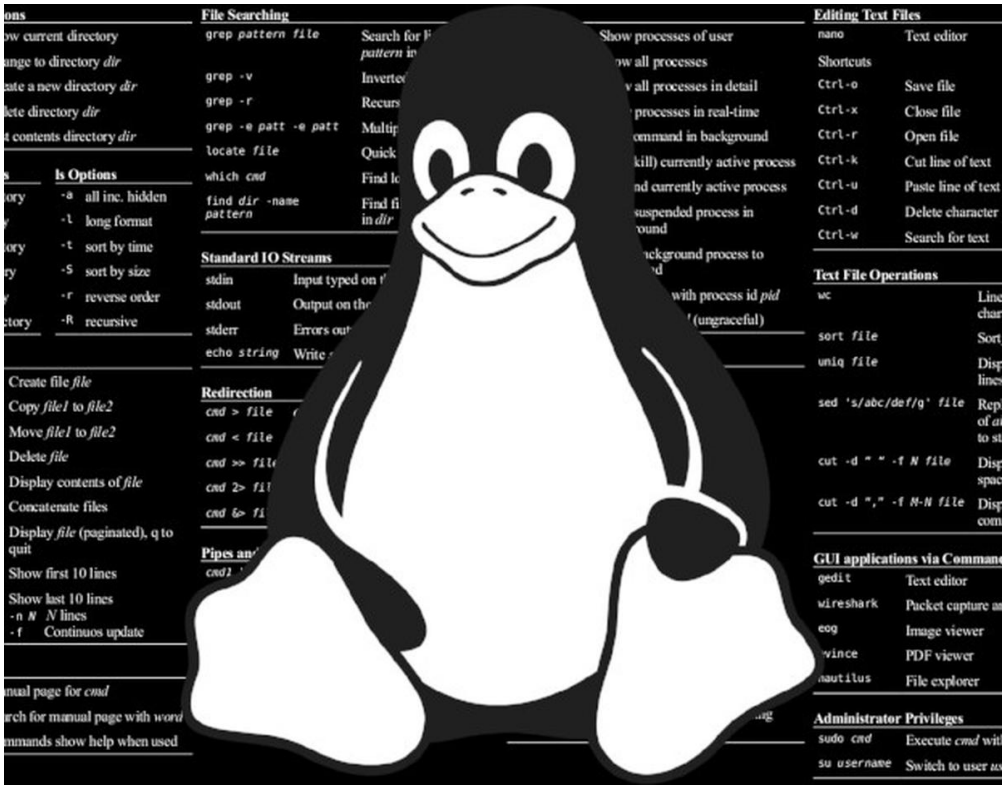
HMS Research Computing

Course Objectives

- Transferring data with rsync
- Linux tools
- Bash “for” loops
- Handling command output
- Customizing your O2 account environment
- SLURM deeper dive
- Cron

Login to O2

Linux tools


File Searching
grep pattern file
grep -v
grep -r
grep -e patt -e patt
locate file
which cmd
find dir -name pattern
Standard IO Streams
stdin
stdout
stderr
echo string
Redirection
cmd > file
cmd < file
cmd >> file
cmd 2> file
cmd &> file
ls Options
-a all inc. hidden
-l long format
-t sort by time
-S sort by size
-r reverse order
-R recursive
Create file file
Copy file1 to file2
Move file1 to file2
Delete file
Display contents of file
Concatenate files
Display file (paginated), q to quit
Show first 10 lines
Show last 10 lines
-n N N lines
-f Continuos update
Show processes of user
Editing Text Files
nano Text editor
Shortcuts
Ctrl-o Save file
Ctrl-x Close file
Ctrl-r Open file
Ctrl-k Cut line of text
Ctrl-u Paste line of text
Ctrl-d Delete character
Ctrl-w Search for text
Text File Operations
wc
sort file
uniq file
sed 's/abc/def/g' file
cut -d " " -f N file
cut -d "," -f N-N file
GUI applications via Comman
gedit Text editor
wireshark Packet capture
eog Image viewer
PDF viewer
nautilus File explorer
Administrator Privileges
sudo cmd
su username

O2 data transfer: which tool to use?

	Local	Remote	Not supported
Tools	cp rsync	sftp scp rsync wget ftp [more]	Inbound FTP and anything else which does not transmit over SSH (port 22).

rsync: most common use

- Local on O2:

  $ rsync -av source/ destination/

  ```
  -a (-rlptgoD , recursive and preserves permissions)
  -v (verbose)
  ```

- Over a network to O2:

  $ rsync -av -e ssh source/ user@transfer.rc.hms.harvard.edu:destination/

  ```
  -z (data compression) option may be useful
  ```

- Dry run: test your command without actually copying

  ```
  $ rsync -n -av source/ destination/
  ```

rsync: more options

- Synchronize directories (be careful !!)

  $ rsync -delete -av source/ destination/

  - this ***overwrites and deletes*** files in the destination which don't match what is in the source.

- Set permissions

  $ rsync -chmod=ug+rw [..]

- Exclude patterns or a list of files from transfer:

  $ rsync -exclude '*.bam' [..]

  $ rsync -exclude-from 'exclude-list.txt' [..]

Exercise: rsync

- Copy the class directory with rsync: *(dry run: -n)*

  $ rsync -n -av /n/groups/rc-training/o2_intermediate ~/

- For real:

  $ rsync -av /n/groups/rc-training/o2_intermediate ~/

head / tail / less / more / cat

- Commands to view text in a file or stream.
- Exercise: examine contents of a data file

```
$ cd ~/o2_intermediate/data
$ cat example.gtf
$ less example.gtf
$ more example.gtf
$ head example.gtf
$ head -20 example.gtf
$ tail example.gtf
$ tail -20 example.gtf
$ tail -f example.gtf                (CTRL-C to quit)
```

ln

- A link is a special file type
  - ln with the -s option is the most common use: “symbolic”
  - Symbolic links work across filesystems

- Example / Exercise:

```
$ mkdir work                (make a directory)
$ ln -s work shortcut       (make a link called “shortcut”)
$ ls -l                     (lower-case “L” file type)
```

find

- find [path to search] [expression] [actions]
  - -name : the filename / pattern
  - -user : user owner
  - -group : group owner
  - -type : type of file (plain file, directory, pipe. etc)
  - -ctime : time of file creation
  - -atime : last access time of a file
  - -mtime : last modification time of a file
  - -exec [command] : runs a command against find's output
  - *(and lots more…)*

find: examples

- List all files matching the name *.bam

```
$ find ./dir -name '*.bam'
```

- Make all files group-writable under a directory:

```
$ find ./dir -type d -exec chmod -v g+rwxs {} \;
$ find ./dir -type f  -exec chmod -v g+rw {} \;
$ find ./dir -exec chgrp -v labgroup {} \;
```

- Remove files not updated in the past 60 days:

```
$ find ./dir -mtime +60d -exec rm -v {} \;
```

find: exercise

- Create symbolic links to all bam files located under a directory tree:

```
$ cd ~/o2_intermediate
$ find . -name '*.bam'
$ find . -name '*.bam' -exec ln -s {} \;
```

wc

- word count
  - -l print number of lines
  - -w print number of words

- Example: (how many lines are in a file)

```
$ cd ~/o2_intermediate/data
$ wc -l example.gtf
```

du

- estimate file space usage
  - -s print size (Kb is default)
  - -h print human readable format (Kb/Mb/Gb/Tb)

- Example: (how many lines are in a file)

```
$ cd ~/o2_intermediate/data
$ du -sh example.gtf
$ du -sh *
```

Commands for Text Processing

sort

- sort lines of text

```
$ sort   file.txt
```

- a few common options:
  - -r (reverse order)
  - -h (human numeric sort: e.g. 2K, 1G, 500M)
  - -u (remove duplicate lines)

Exercise: sort

```
$ cd ~/o2_intermediate
$ sort  sort.txt
$ sort  -r  sort.txt
```

uniq

- report or omit repeated lines

  $ uniq file.txt

  - with no options, uniq prints all lines but removes duplicate entries

- a few common options:

  - -i (ignore case)
  - -c (prefix lines by number of occurrences)
  - -d (print only repeated lines)
  - -u (print only unique lines)

Exercise: uniq

Try these commands:

```
$ cd ~/o2_intermediate
$ cat   uniq.txt
$ uniq   uniq.txt
$ uniq  -d  uniq.txt
$ uniq  -u  uniq.txt
$ uniq  -c  uniq.txt
```

grep (global regular expression print)

- print lines matching a pattern
  - $ grep pattern file.txt
  - $ grep '#pattern 2' file.txt
- a few common options:
  - -i (case-insensitive)
  - -v (does not match the pattern)
  - -n (precede matching line with a line number)

Exercise: grep

```
$ cd ~/o2_intermediate/data
$ grep  stop_codon  example.gtf
$ grep -v stop_codon  example.gtf
$ grep -n stop_codon  example.gtf
$ grep -i cds  example.gtf
```

cut

- remove sections from each line in a file / stream
  - -d defines delimiter (default is a Tab)
  - -s prints only lines containing a delimiter
  - -f prints specified fields

- Examples:

```
$ cut -f 1 file.txt                  (print 1st field only)
$ cut -f 1,3 file.txt                (print 1st & 3rd fields)
$ cut -s -d ":" -f 1 file.txt        (colon space delimiter)
$ O2squeue | cut -s -d " " -f 1      (list of O2 job IDs)
```

Exercise: cut

- remove sections from each line in a file / stream
- default delimiter is a Tab

```
$ cd ~/o2_intermediate/data
$ head example.tab
$ cut -f 1,2 example.tab
$ cut -f 1,3 example.tab
```

awk and sed

- awk
  - a special-purpose programming language for text processing
  - Does similar things as PERL, but sometimes awk gets it done quicker.
  - Example: calculate the average of column 2:

    $ awk ‘{x+=2}END{print x/NR}’ file.txt

- sed
  - a stream editor that works on a per-line basis.
  - Example: global substitution of the string “Harvard” -> “HMS”

    $ sed 's/Harvard/HMS/g' doc.txt > doc_new.txt

Working with Command Output

Command output redirection:

- Redirect: >
  - sends output to a file, overwrites any existing file

    $ grep pattern file.txt > out.txt

- Append: >>
  - sends output to a file, appends to any existing file

    $ grep pattern file.txt >> out.txt

- Pipe: |
  - sends output to be input for another application

    $ cut -1 file.txt | sort | uniq -c

Exercise: handling command output

- Sort field entries from a data file (example.gtf)
- default delimiter is a Tab

```
$ cd ~/o2_intermediate
$ cut -f 4 example.gtf > out.txt
$ grep -i cds  example.gtf >> out.txt
$ cut -f 4 example.gtf > out.txt
$ cut -f 4 example.gtf  |  sort -n | uniq -c
$ grep  stop_codon  example.gtf  |  wc -l
```

Redirecting Standard Error (stderr)

- bash syntax:

```
$ command  2>out.err          (send stderr to a file)
$ command  2>&1               (send stderr to stdout)
$ command > out.txt 2>&1      (send stderr and stdout to a file)
```

- Exercise:

```
$ cd ~/o2_intermediate
$ cat no.txt                  (file does not exist — error)
$ cat no.txt  >out.err        (saves stderr to a new file)
```

Customizing your O2 account

Customizing your O2 account

- Aliases: create your own commands!

  $ alias ll='ls -la'

  $ alias h=history

- Change your default umask
  - Example: create group-writable files by default:

    $ umask 0002

- Set, environment variables like command path:

  $ export PATH=$PATH:/home/user/bin

Adding customizations on login

- ~/.bash_profile
  - executed on login
  - executed once before you get a prompt.

- ~/.bashrc
  - Supplemental config file, executed each time you run “bash”
  - On O2, gets run from ~/.bash_profile
  - Typically, this is where most customizations go:
  - aliases, modules, $PATH, other variables, etc.

Sample ~/.bashrc file

```
$ cat ~/.bashrc
#
alias ll 'ls -la'
alias h history
#
module load gcc/6.2.0
module load R/3.5.1
#
export PATH=$PATH:/home/user/bin
export DUO_PASSCODE=push
```

Exercise: edit your .bashrc file

$ nano ~/.bashrc

(Add some things you would like to set automatically on login)

$ source ~/.bashrc

(to manually run it without having to re-login)

Try it out! (Run an alias command, etc)

bash “for” loops

Automate commands with a “for” loop

- Repeat commands against an designated list
  - this syntax is for bash, but other shells (tcsh) are different

- Examples

  $ for i in 1 2 3 ; do mkdir $x ; done

  $ for i in `cat list` ; do cp $x ~/work ; done
  - more complex loops can be put in bash scripts
  - also useful for submitting batches of jobs to O2!

“for” loop in a shell script

```
#!/bin/bash

list=/home/user/files.txt

for i in `cat $list`
   do
        [command 1]
        [command 2]
   done
```

a few things about Slurm...

Jobs with command line arguments

```
#!/bin/bash
#SBATCH -p short   #partition
#SBATCH -t 0-01:00 #time days-hr:min
#SBATCH -o %j.out  #out file
#SBATCH -e %j.err  #error file

echo $1
```

Exercise: Jobs with arguments

- Run the following:

```
$ cd ~/o2_intermediate
$ sbatch  arguments.sbatch  hello
```

- The output file will contain the argument "hello"

- This technique gets more useful when submitting from a script and the arguments vary over iterations.

A better example (bamsort.sbatch)

```
#!/bin/bash
#SBATCH -p short   #partition
#SBATCH -t 0-01:00 #time days-hr:min
#SBATCH -o %j.out  #out file
#SBATCH -e %j.err  #error file

## Update path for your account:
## dir=/home/rc_training000/o2_intermediate/data

module load gcc/6.2.0
module load samtools/1.9

samtools sort $1 > $dir/"${1%.*}".sorted.bam
#where $1 is a bam file
```

Using sbatch with a bash “for” loop

- To submit a bunch of separate jobs systematically:

```
$ for i in [input] ; do [sbatch command] ; done
```

- Exercise:

```
$ cd ~/o2_intermediate
$ for i in *.bam ; do sbatch bamsort.sbatch $i ; done
```

Canceling one or more job

The [-u] option is always required.

```
$ scancel -u your_user
$ scancel -u your_user -v[vv]
$ scancel -u your_user -p short
$ scancel -u your_user -t PENDING
$ scancel -u your_user -t RUNNING
$ scancel -u your_user -t SUSPENDED
$ scancel -u your_user JOBID1 JOBID2 [..]
```

Canceling jobs: exercise

```
$ cd ~/o2_intermediate

#submit some jobs and kill them:
$ for i in *.bam ; do sbatch bamsort.sbatch $i ; done
$ scancel -u your_user    #kill all jobs

#repeat:
$ for i in *.bam ; do sbatch bamsort.sbatch $i ; done
$ scancel -u your_user JOBID1 JOBID2
```

Job Monitoring

```
$ O2squeue
$ squeue -u your_user
$ squeue -u your_user -t PENDING
$ squeue -u your_user -t RUNNING
$ squeue -u your_user -p short

$ O2sacct
$ sacct -j JOBID
```

Cron

Process automation: cron

- Task Scheduler for Linux
- O2 has a centralized cron server where jobs get executed.
- Examples:
  - Automate a nightly rsync process
  - Run a weekly analysis report
  - Purge old files on a schedule

Cron: Editing a Crontab

- Create/Edit a crontab from a login server using: **crontab -e**
- Format of a cron job process:

  [Minute] [Hour] [Date] [Month] [Day of the Week] Command

  Asterisk (*) = “every”

- Example: have a job run at 2:00am every Monday:

```
0 2 * * 1 sbatch /home/user/rsync.sbatch
```

For more direction

- http://hmsrc.me/O2docs
- http://rc.hms.harvard.edu
- RC Office Hours: Wed 1-3p Gordon Hall 500
- rchelp@hms.harvard.edu