

# **Linux Basics**

**Dr. Jung Soh**

**657.019 Transcriptomics (WS 2018/19)**

# Introduction

- ▶ Linux approach to doing things
  - Sets of simple building blocks (commands) exist
  - Users put them together to do what they want
  - Be lazy! Get computers do repetitive, tedious things
- ▶ Why command line?
  - Graphical User Interfaces are easier for some things
    - Word processing, video editing, graphic design
  - Command lines are far better for other things
    - Data exploration/reporting, file manipulation

# Outline

- ▶ Terminal/command line
- ▶ Moving around
- ▶ About files
- ▶ Manual pages
- ▶ File manipulation
- ▶ Wild cards
- ▶ Permissions
- ▶ File filters
- ▶ Piping/redirection
- ▶ Job control

# Convention: < > and [ ]

## ▶ command <something>

- Replace <something> with an actual thing
- <something> is required
- Examples:
  - `kill <pid>` -> `kill 78392`
  - `ls <file>` -> `ls myfile.txt`

## ▶ command [something]

- Supply [something] or omit [something]
- [something] is optional
- Examples:
  - `wc [-c] <file>` -> `wc myfile.txt`
  - `head [-number] <file>` -> `head -47 myfile.txt`

# Terminal

- ▶ Opening terminal
  - Applications -> Terminal Emulator, or
  - Right-click on desktop, then “Open Terminal Here”
- ▶ A terminal runs a shell
  - `echo $SHELL` to see current shell
  - The process:
    - At prompt, type command, finish it with Enter
    - Shell runs the command, prints the output
    - Shell prints prompt again
- ▶ Be lazy! Use command history
  - Up/down arrow keys bring previous/next command
  - Left/right arrow keys let you edit

# Moving around: key commands

- ▶ **pwd**
  - Print working directory (=folder): Where am I?
- ▶ **ls [options] [locations/files]**
  - List: What do I have in current or given location?
  - Most useful option is **-l** (long list), to show more information
- ▶ **cd [location]**
  - Change directory: Move to a specific location

# Moving around: 2 kinds of paths

- ▶ Path = location = directory or file

	Absolute Paths	Relative Paths
Begin with	/ or ~	Any other letter than / or ~
Relative to	Root location (/)	Current location
Where I am matters?	No	Yes
Examples	<code>/export/home/jsoh</code> <code>/tmp</code> <code>/bin/pwd</code> <code>/etc/init.d</code> <code>~jsoh</code>	<code>Documents/maps</code> <code>../usr</code> <code>./runme.sh</code>

# Moving around: special paths

- ▶ `~` (tilde): your home directory
  - `~` = `/export/home/json`
  - `~/Desktop` = `/export/home/json/Desktop`
  - Begins an absolute path
- ▶ `.` (dot): current directory
  - Begins a relative path
- ▶ `..` (dotdot): parent directory (one level up)
  - Use repeatedly to go up hierarchy (`..../..` goes two levels up)
  - Begins a relative path

# Moving around: practice

- ▶ **cd** (without arguments) takes you to your home directory
- ▶ Try these commands:
  - **pwd**
  - **ls**
  - **cd Documents**
  - **ls**
  - **cd /var**
  - **ls**
  - **cd ~/Documents**
  - **pwd**
  - **cd ../../..**
  - **pwd**
  - **cd**
  - **pwd**

# Moving around: Tab

- ▶ Be lazy! Use TAB completion
  - Typing a long path is too much work and is error-prone
  - Hit Tab after typing one or two initial letters
  - Path completed automatically if unique
  - If multiple paths match, hit Tab again to see possibilities and choose
  - Safe (can't change anything), faster, avoid errors
- ▶ Try typing this command with tab completion

```
ls /export/projects/eremophila/interproscan_out/all_orfs
```

# About files

- ▶ Everything in Linux is a file
  - Text, data, directory, keyboard, monitor, ...
  - **file <path>** shows the type of **<path>**
- ▶ Names are cAsE seNsitive
- ▶ Spaces in a file name allowed, but highly discouraged
  - Use quotes or escape character (\) to treat multiple words as one
    - **cd 'My documents'** or **cd My\ documents**
- ▶ Hidden files/directories
  - Begin with a **.** (dot)
    - configuration files etc
  - **ls -a** (all) shows hidden items

# Manual pages

- ▶ **man <command>**
  - Prints manual pages of command
  - Try `man ls` (type `q` to quit)
- ▶ **man -k <term>**
  - Lists command related to `<term>` (keyword)
  - Try `man -k PDF`
- ▶ **Searching while viewing manual pages**
  - `/<term>` searches for `<term>`
  - `n` shows the next match for `<term>`
  - Try `man ls` then search for `subdir` by typing `/subdir`

# File manipulation: directories

## ▶ **mkdir [options] <directory>**

- Make directory: Start organizing your files
- Try on your home directory:
  - `mkdir linux_basics`
  - `ls`

## ▶ **rmdir [options] <directory>**

- Remove directory (directory must be empty)
- Try on your home directory:
  - `rmdir linux_basics`
  - `ls`

# File manipulation: copy

- ▶ **cp [options] <source> <destination>**
  - Copy source (file without **-r** option) to destination
  - **cp <file1> <file2>**
    - Copy **<file1>** to **<file2>**
  - **cp <file> <directory>**
    - Copy **<file>** to **<directory>/<file>**
  - **cp -r <directory1> <directory2>**
    - If **<directory2>** exists
      - Copy **<directory1>** to **<directory2>/<directory1>**
    - If **<directory2>** does not exist
      - Copy **<directory1>** to **<directory2>**

# File manipulation: move/rename

▶ **mv [options] <source> <destination>**

- Move <source> to <destination>
- **mv <file1> <file2>**
  - Move/rename <file1> to/as <file2>
- **mv <file> <directory>**
  - Move <file> to <directory>
- **mv <directory1> <directory2>**
  - If <directory2> exists
    - Move <directory1> to <directory2>/<directory1>
  - If <directory2> does not exist
    - Move/rename <directory1> to <directory2>

# File manipulation: remove

- ▶ **rm [options] <file>**
  - Remove **<file>**: cannot undo!
- ▶ **rm -r <directory>**
  - Remove **<directory>** (directory can be non-empty)
  - Recursive: everything under **<directory>** will be gone!
- ▶ **cp** and **mv** can be silently destructive
  - Overwrites destination file (if it already exists)
  - Use “interactive” option to have a chance to confirm:
    - **cp -i**
    - **mv -i**

# Text file viewing/editing

- ▶ **cat <file>**
  - View <file>
- ▶ **less <file>**
  - View <file> one screen/page at a time
  - Space: next page, **b**: previous page, **q**: quit
  - **more <file>** works similarly
- ▶ Text file editing
  - Gedit is a simple editor
    - **gedit <file>**
  - Vi is a traditional, more powerful editor
    - **vi <file>**

# Wildcards

- ▶ Refer to multiple files or directories
  - **\*** : zero or more characters (anything)
  - **?** : any single character
  - **[ ]** : possible single characters
- ▶ Can be used anywhere in a path
- ▶ Try on your home directory:
  - `ls -d D*`
  - `ls -d Do*`
  - `ls -d /export/data/????`
  - `ls /usr/bin/[yz]*`

# Permissions: abilities and people

- ▶ 3 things you can do with a file
  - **r**ead: can read the contents
  - **w**rite: can change the contents
  - **e**xecute: can execute (run) if it is a program or script
- ▶ 3 sets of people
  - **u**ser: owner of the file (always one person)
  - **g**roup: group the file belongs to (always one group)
  - **o**thers: all other people (not the owner, not in the group)
- ▶ A file has  $3 \times 3 = 9$  permission characters

# Permissions: viewing

- ▶ Order: user **rwx**, group **rwx**, others **rwx**

```
[jsoh@cbt01 ~]$ ls -l duout  
-rw-r--r--. 1 jsoh ucalgary 562 May 31 17:46 duout
```

- ▶ Character 1: file type (**d**: directory, **-**: file)
  - **-**: file
- ▶ Next 3: user (owner) **rwx**
  - **rw-**: **jsoh** can read, write, but not execute
- ▶ Next 3: group **rwx**
  - **r--**: People in group **ucalgary** (except **jsoh**) can only read
- ▶ Next 3: others **rwx**
  - **r--**: Others can only read

# Permissions: changing

- ▶ **chmod <permissions> <path>**
- ▶ Permissions is a 3-part string
  - Who this change is for: a combination of **u** (owner), **g** (group), and **o** (others), or **a** (all)
  - Grant or deny: either **+** (grant) or **-** (deny)
  - Which permissions: a combination of **r** (read), **w** (write), and **x** (execute)

```
[jsoh@cbt01 ~]$ ls -l duout
-rw-r--r--. 1 jsoh ucalgary 562 May 31 17:46 duout
[jsoh@cbt01 ~]$ chmod go+w duout
[jsoh@cbt01 ~]$ ls -l duout
-rw-rw-rw-. 1 jsoh ucalgary 562 May 31 17:46 duout
[jsoh@cbt01 ~]$ chmod o-w duout
[jsoh@cbt01 ~]$ ls -l duout
-rw-rw-r--. 1 jsoh ucalgary 562 May 31 17:46 duout
```

# Permissions: directories

- ▶ Directory permissions are interpreted differently than file permissions
  - **r**: Can read the contents of the directory (e.g. can do `ls`)
  - **w**: Can create/remove files and directories in the directory
  - **x**: Can enter the directory (i.e. can do `cd`)

# Simple line-based commands

- ▶ Some useful Linux commands for text processing
  - **head**: Print first (default 10) lines of input
  - **tail**: Print last (default 10) lines of input
  - **sort**: Sort (default alphabetical order) lines of input
  - **uniq**: Print/count unique lines of sorted input
  - **wc**: Count lines, words, and characters of input
  - **cut**: Cut out (vertically) certain fields of input
  - **paste**: Paste together (sideways) input fields
  - **nl**: Number lines of input
  - **tac**: Print last line first, ..., first line last
  - **rev**: Print last character first, ..., first character last, per line
- ▶ Check out the man pages for these!

# More powerful text commands

- ▶ **grep**
  - Print lines of input containing some text
- ▶ **diff**
  - Show difference between two files
- ▶ **awk**
  - Field-based line processing
  - Actually a scripting language

# Redirection and piping

- ▶ > saves (redirects) command output into a file

```
[jsoh@cbt01 ~]$ ls -d b*  
b2gFiles  b2gWorkspace  bin  blast-db-FASTA  
[jsoh@cbt01 ~]$ ls -d b* > start_with_b_files  
[jsoh@cbt01 ~]$ cat start_with_b_files  
b2gFiles  
b2gWorkspace  
bin  
blast-db-FASTA
```

- ▶ >> appends to an existing file
  - > overwrites/deletes the existing file
- ▶ | pipes to (sends output as input of) next command

```
[jsoh@cbt01 ~]$ ls -d b*  
b2gFiles  b2gWorkspace  bin  blast-db-FASTA  
[jsoh@cbt01 ~]$ ls -d b* | grep b2 | wc -l
```

# Job control

- ▶ **Ctrl-C** kills most running (foreground) programs
  - To kill hanging programs or wrong commands
- ▶ **Ctrl-Z** stops (suspends) most running programs
  - **bg** right after runs the program in background
- ▶ **jobs** shows the (terminal-specific) job numbers of stopped or background jobs
  - **fg %<job number>** runs the job in foreground
  - **bg %<job number>** runs the stopped job in background
- ▶ **ps** and **kill** commands allow full control of processes
- ▶ **top** shows top (in terms of CPU usage) processes

# Summary

- ▶ There are only a handful of key commands to learn, in order to start using Linux
- ▶ When in doubt, look up man pages
- ▶ Use Tab completion for typing paths (and commands)
- ▶ Be aware of potentially destructive commands
- ▶ Learn by doing