Linux Command Line

By: Megan Avery
Updated July 2018

Agenda

Basic command line usage and the most common terminal commands

Notes about the Mac command line

- All the commands in this slidedeck will work on mac as well as Linux with some exceptions. Those exceptions will be noted.

- Mac has a built in terminal but I like to use iTerm because it looks a little nicer

Before We Begin

- Make sure that you have the “Linux_Examples” folder in your “Documents” folder. This will allow you to follow along with the command line walk through.
- Open a new terminal, you should be defaulted to inside your ~ directory. This is where you want to be when the walkthrough begins.

Basic Terminal

#

Any command with a pound sign at the beginning is a no op

[up arrow]

- this will bring up the last command you used in the your terminal
    - if used multiple times will keep going up in your command history
- can be useful when using long/ repeated commands
- the down arrow will go back through commands in the other direction

history

- this will print out a list of your previous terminal commands
- can be useful if you are trying to remember a complex command you did previously but can’t quite remember

clear

- this will clear your terminal screen
- can be useful when have just run something complex and would like a clean slate

man

- short for **man**ual
- **usage:** `man [command]`
  - will give you helpful usage information about certain commands/ system calls
  - is especially useful to look at the flags for commands

echo

- this will basically just print something to your terminal window
- usage: `echo [words]`
- is really useful in bash scripts
    - **NOTE:** bash scripts are a way to run a collection of terminal commands as a single command

[ctrl] + c

- this will stop whatever is currently running in your terminal
- can be especially useful when you accidentally run a program with an infinite loop
    - Or any other long running program/ script that you want to stop

[ctrl] + [shift] + c

- copy something from you terminal
- On mac: [command] + c
- useful if trying to Google what an error means

[ctrl] + [shift] + v

- paste into the terminal
- On mac: [command] + v
- useful if just Googled a way to do something cool

[ctrl] + r

- This will let you search previously used commands
- This is useful if you used a long command in the past and remember a word from it
- Pressing [ctrl] + r again will traverse previous occurrences of the search query

sudo

- short for **s**uper **u**ser **do**
- will allow you to run commands you normally aren't allowed to
- **usage:** `sudo [command]`
  - **NOTE:** requires you to have password permission

ssh

- short for **s**ecure **sh**ell
- **usage:** `ssh [host]@[IP]`
- works natively for Macs and Linux machines
- for Windows machines will need to use Putty
  - WinSCP is also a great program for Windows
- there is also a secure shell extension for Chrome

exit

Will close out of the terminal window without having the hit the little x in the corner

Directories and Files

Vocab Alert!

Directory: folder

Path: information that goes before a file or directory name that means it lives somewhere besides the current directory

- **Note:** . is the current directory and .. is the directory above the current one

Relative Path: a path that is calculated from the current directory

- ex: Documents/Linux_Examples

Absolute Path: a path that is calculated from the base directory, always begins with a /

- ex: /Users/mavery/Documents/Linux_Examples

To Note

- Any [source] and [destination] in subsequent command slides can be file/ directory names with/without paths
- Any [file] in subsequent command slides can be files with/without paths

ls

- short for **li**s**t**
- lists all the files/ directories in the current directory
- you might also want to try the sl command on the lab machines
- Can also call like **`ls [directory]`** to get list for a different directory

ls -l

- will list files along with their permissions
- permissions:
  - read - can view the stuff
  - write - can edit the stuff
  - execute - can run (for scripts and such)
- 3 sets
  - (owner) (group) (anyone)

pwd

- short for **p**rint **w**orking **d**irectory
- any easy way to know where you are in the file hierarchy if you forget

[tab]

This will autocomplete whatever you are currently doing in the terminal

cd

- short for **c**hange **d**irectory
- used to navigate between directories in your file structure
- **usage:** `cd [directory]`

mkdir

- short for **ma**ke **dir**ectory
- will make a new directory for you
- **usage:** `mkdir [directory name]`
  - **NOTE:** can also use relative vs. absolute paths instead of just a directory name

cp

- short for **co**p**y**
- a way to make a copy of something in a different directory
- **usage:** `cp [source] [destination]`
    - again can use relative or absolute paths for the source and destination
    - **NOTE:** Copies to destinate and keeps the original in source as well
    - **NOTE:** When copying directories you will need to use the -r flag

scp

- short for **s**ecure **cop**y
- a way to copy files between computers
- **usage:** `scp [source] [destination]`
    - from other computer: `scp [host]@[ip]:[source/file name] [destination on your computer]`
    - to other computer: `scp [source/file name] [host]@[ip]:[destination on other computer]`

touch

- will either create a new file or update the last modified date on a file to the current date
- **usage:** `touch [file]`

mv

- short for **mo**v**e**
- a way to actually move files/directories around on your computer
  - also an easy way to rename directories
- **usage:** **`mv [source] [destination]`**
  - as usual you can use either a relative or absolute path for the source and destination

cat

- short for **cat**enate
- will print a file's contents to the terminal
- **usage:** `cat [file]`

rm

- short for **rem**ove
- deletes a file
- **usage:** `rm [file name]`
- helpful things:
  - `rm -rf [directory name]`
    - will delete a directory and everything inside it
    - use with caution, if you don’t give a destination for this it will delete EVERYTHING from your current directory down

chmod

- used to change permissions
- usage: `chmod [new settings] [destination]`
- new setting options

Reference	Operator	Mode
u - user g - group o - others a - all (everybody)	+ add - remove = set exactly	r - read w - write x - execute

Vocab Alert! - WIP

Regex: regular expression, a way of doing custom word matching

Simple usage:

- * - wildcard, will matching anything
- ? - zero or more occurrences
- + - one or more occurences
- | - or
- ( ) - group a part of a query

https://en.wikipedia.org/wiki/Regular_expression

grep

- a way to search through file(s)
- usage: **`grep [search query] [file]`**
    - can search for things using regex
- helpful flags:
    - -n lists the line number next to matches
    - -r search recursively
    - * instead of a file name will search the whole directory

find

- used to find out where a file lives in your file hierarchy
- usage: `find [path] -name [file]`
    - if path is not given then will search the current directory and every directory it contains

diff

- short for **diff**erence
- shows the difference between 2 files
- **usage:** `diff [file 1] [file 2]`
- helpful flags:
  - -b ignore whitespace diffs
  - -i ignore case
  - --side-by-side - see differences next to each other

head

- Shows only the the beginning of a file, the number of lines based on input
- **usage:** `head -[number] [file]`

tail

- Shows only the end of a file, the number of lines based on input
- **usage:** `tail -[number] [file]`

Java

java -version

- will tell you what version of Java is currently installed on your machine
- will also tell you if java is not installed on your machine at all

javac

- used to compile a java program
- **usage:** `javac [file]`
  - file must have the .java extension
- if successful will create a .class file with the same name as the original Java file

java

- used to run a compiled Java file
- usage: **`java [name of .class file without extension]`**
    - ex: java Test
        - don’t put .class at the end of the file name
        - this would have come from compiling a file called Test.java

Redirection Input/ Output

|

- **usage:** `[command 1] | [command 2]`
- will make the output from command on the left the input for the command on the right, can chain together as many commands as you need

> and >>

- **usage:** `[command] > [file]`
- will redirect output on left into the file on the right
- single > will will replace the contents of the file with the given output and double >> will append to the file

<

- usage: `[command] < [file]`
- will redirect thing on the right to be the input for the thing on the left
- is really good for testing projects that take in user input

Fun Stuff

cal

will give you a little ASCII calendar of the current month with the current day highlighted

date

will give you the current date and time as a string

yes

- will print the same phrase repeatedly in your terminal until you hit `[ctrl] + c`
- **usage:** `yes [some words]`

cowsay

- will take a phrase and print a little ASCII art cow saying that phrase
- usage: **`cowsay [some words]`**
- can also pipe things into cowsay
    - you could have a cow tell you your grep output
- there are also many other animals you could do
    - for a list do cowsay -l
    - **usage for different animal:**
        - **`cowsay -f [animal file] [some words]`**
        - Get a list of possible animal files with **`cowsay -l`**

THE END!