**Command Line**

**Commands**

* tab
  + start typing a filename, then press tab for autocompletion
* ctrl + c
  + interrupt the current command
* ‘’
  + use single or double quotes around any file names that have spaces in them
* |
  + way to chain commands
  + the output from the statement on the left is used as the input in the statement on the right
  + can chain multiple commands together this way
* up arrow
  + cycles through previous commands
* history
  + prints out a list of previously entered commands
* !*serial*
  + will rerun the command from your history with that particular *serial* number
  + i.e. !7
  + !head will rerun that last used version of the head command you used
* wildcard chars
  + \*
    - use to specify 0 or more chars (when used after a dir in a path, it selects everything)
    - this is a wildcard
      * s\* starts with s (may or may not contain additional chars)
      * path/\* everything in the path
  + ?
    - matches a single character (put it wherever, including in the middle of a name)
  + []
    - matches any single char inside the brackets
  + {}
    - matches any of the comma separated patterns in the brackets
    - these patterns can include other wildcards
* pwd
  + print working directory
* ls [*path*]
  + lists the contents of the working dir or the specified *path*
  + ls ~ (shows home dir)
  + ls -R (shows everything below the dir)
  + ls -F (prints a / after every dir and a \* after every runnable program) (use in conjunction with -R)
* cd *path*
  + change directory to the specified *path*
  + cd .. goes up one dir (don’t forget space between dots and cd)
  + cd ~ for home dir
* cp *original copy*
  + copies an *original* file with new name *copy* into the same dir
  + if you add a directory at the end of the statement, it will attempt to copy all listed files to the new dir
  + you can create a copy in a new dir by specifying the path when you name the *copy*
  + can **overwrite files** if the *copy* already exists!
* mv *file(s)* *directory*
  + moves the *file(s)* to the specified *directory*
  + can rename files and directories using this
    - mv *old-name new-name*
      * will **overwrite files** if *new-name* already exists
* rm *file(s)*
  + deletes all of the supplied *file(s)*
  + no “trash can”, files are gone for good!
* rmdir *dir*
  + deletes an entire directory (must be empty for this to work)
* mkdir *dir*
  + creates a new directory within the current directory
* cat *file(s)*
  + will display the contents of any and all *file(s)* passed
* less *files*
  + will display the contents of the *files* one page at a time
  + :n move to the next page
  + :p move to the previous page
  + :q quit
* head *file*
  + displays the first 10 lines of a file
  + good for viewing column names in csv files
  + head -n *num* will print the specified *num* of lines
* tail
  + displays the last lines of a file
* man *command*
  + opens the manual for a command using the “less” mode
* cut *options* *file*
  + displays columns rather than rows like head does
  + *options*
    - -f *columns* -d *delimiter*
    - *columns* can be a range 1-5 or comma separated 1, 5
    - *delimiter* can be a , (and maybe a tab somehow?)
* paste *options* *files*
  + pastes the files together in a display
  + options
    - -d delimiter
* grep *text file*
  + displays any lines from *file* that contain the supplied *text*
  + common flags
    - -c print count of matching lines rather than the lines themselves
    - -h do not print the names of files when searching multiple files
    - -i case insensitive
    - -l print the names of files that contain matches, not the lines
    - -n print the line numbers for the matches
    - -v show only the lines that don’t match
* > *filename*
  + redirects the output of a command to a new file
* wc
  + “word count”
  + counts the total number of chars, words, and lines in a file
  + options
    - -c, -w, -l flags will count just one of the above
* sort
  + puts data in alphabetical order
  + -n numerical order
  + -r reverse order
* uniq
  + removes **adjacent** duplicate lines
  + can add -c to count

**Flags**

* good practice to use - and put flags before filenames or directory paths
* -n *num*
  + number of lines
* -R
  + recursive
  + use with ls
  + shows everything underneath (no matter how deep) the current directory
* -F
  + prints a / after every dir and a \* after every runnable program (use in conjunction with -R)
  + use with ls

**Paths**

* start with / to specify an absolute path
* do not start with / to specify a path relative to the current dir
* Special Chars
  + ~ home dir (/home/repl)
  + .. up a dir
  + . here

**Environment Variables**

* full list available by entering “set”
* use a variable with “echo” and a $ to print its value
  + echo $USER displays the current USER value
* HOME
  + user’s home directory
  + /home/repl
* PWD
  + same as pwd
* SHELL
  + which shell program is being used
  + /bin/bash
* USER
  + user’s ID
  + repl

**Shell Variables**

* remember to use $ to access the value of a variable
* *var=value*
  + *value* could be a filename or a path, etc.

**Loops**

* for *variable* in *list*; do *command*; done
  + can have multiple do commands, but only the first one gets “do” and separate each with a ;
* *list* items should not have a comma between if enumerating directly in the loop
* Example
  + for filename in /path/\*; do echo $filename; done
    - displays every file in the /path/
* Example 2 using variables
  + files=path/\*.csv
  + for f in $files; do echo $f; done
* Writing in a shell script (.sh file) to use with bash
  + for *variable* in *list*

do

statement 1

statement 2

etc.

done

**Editing a File**

* nano *filename*
  + opens the file for editing
* ctrl k delete a line
* ctrl u un delete a line
  + if you ctrl k, then ctrl u twice, it’s like copying/pasting
* ctrl o save the file (output)
  + prompted to confirm filename (or change) and press enter
* ctrl x exit the editor

**Creating/Executing Files that Contain Commands**

* Creating
  + nano *file.*sh to open/create a file
  + type in commands
  + save
* Executing commands in the file
  + bash *file*.sh *filename*
    - executes code in *file.*sh on *filename*