- **man <command>** = insert any command you want to know more about in the command space of the example
- **echo** = Command used to send all the input to the "standard output"(screen). Format it such as echo "text here" or without the quotation marks.Can also be used to create a blank file
    - -e is the special character modifier. It makes echo read special characters within commands. Such as \t, this is tab. It can look like this: echo -e "\t *"
    - echo "\n this is a new line print"
    - -n = makes it so echo stays on the same line at the end of its output "no new line"
    - \n = new line, goes inside quotes within an echo command.
    - echo $SHELL = shows your current shell
- **history** = command used to show all previous commands. Persists through multiple iterations of the terminal
    - If you want to delete a particular command, enter history -d <line number> . To clear the entire contents of the history file, execute history -c
    - !# replace the number symbol by an actual number so you can execute a specific command line
    - history | more -8 = shows the history 8 lines at a time. "|" is a pipe, found above the Enter key.
- **cd** = change directory
    - cd - = change between your pwd and the previous pwd
- **rm** = command to remove files and directories
    - -r = add this to remove directories
    - {} = insert all the files/directories to be removed in here
    - rm.bash_history is used to remove the history, may need a couple of uses
- **ls** is used to list the available directories
    - Add a A- to this command in order to show hidden files (files that start with a dot)
    - -l = can show even the size of the file in a given directory, long format of this
        - This long format shows the permissions at the beginning of the line. Those permissions are rwx(read, write, execute), they are represented by the dashes are divided on 3. The first 3 dashes are for the user, the following 3

are for the group, and the last 3 are for others. These permissions are set via binary, so the first dash has a value of 4, the second dash has a value of 2, and the third one has a value of 1.

- Read = 4 Write=2 execute=1

- -i = shows the Inode of a file

- **pwd** stands for present working directory
- **cat** = command used to show what a text file holds. Can show multiple files at the same time. It prints the contents from the top to the bottom of the lines
- **tac** = prints the contents from a file from bottom to the top
- >> = command used to append (add to the end of)
- > = is used to send data to a file, it will overwrite any information in there
- **;** = allows for usage of several commands in a single line, such as cd ..;pwd
- White name is a text file, green file is an executable, blue file is a directory, light blue is a link
- **chmod 755** = command used to change a file to an executable file. Follow the command with the name of the file to alter.
    - chmod can then be used to add permissions to the user, group, or others, via
        - Chmod u+x (this is a sample of adding the execute command to the user)
            - u = user
            - g = group
            - o = other
            - a = all
- **./** = command used to execute a file. Place this command right in front of the file to be executed.
- **cp** = is the command to copy files. If formatted like this cp (original file name)(new file name) a new file is then created. You can add new endings to files here.
    - cp ../sept12.txt . this command is copying sept12.txt from a parent directory to the pwd(present working directory), which is represented by the lonely dot at the end
    - cp -r a/b a/c = this is a sample command of how to copy DIRECTORIES
- **exit** leaves the terminal

- **alias** allows to create a shortcut of sorts to a given command. Ex: alias cls='clear'
    - Aliases only last for as long as the current terminal is open.
    - cls used to clear a screen in windows
- **$USER** is accessed by the command whoami, it shows who the user is.
- **more -#** = this will display a specified number of lines at a time
- **^command to be replaced^newcommand** = will replace a previous command with the new command, or it will replace a character for another
- **file** = a command that will tell you the type of a file
- **wc** = word count, it shows (in this order), number of lines, number of words, and number of characters. This works well with the ^ ^ command replacement
    - -w as a modifier will return ONLY the words
    - -l as a modifier will return ONLY the lines
    - -c as a modifier will return ONLY the characters
- **date +'%F'** = short formatted date, it is year - month - day
    - r = 12 hour time
    - R = 24 hour time
    - D = shorter date
- **tee** = think of it as a T intersection, it will send whatever precedes it into a file as well as the screen
    - -a = this command >>
- | = Pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command and so on. It can also be visualized as a temporary connection between two or more commands/ programs/ processes
- **read** = prints a string
    - -p = tells the script to wait
- **uname -s** = shows the name of the shell (maybe)
    - -r = shows the version of the kernel
    - -a = shows you all about the system you are using
    - -i = shows the version of the OS's kernel

- **mkdir** -p a/{b/{d,e},c} = this is a sample command of how to create multiple directories at once
    -
- **tree <directory>** = command to visualize a tree of directories
    - tree -L 1 <directory> = shows one level of a tree
- **touch** a/b/d/fyl{1..4}.txt = this is a sample of a command to create several empty text files into a directory.
    - touch {uniqueName, uniqueNAME}.txt
    - When applied to an already existing file, it will change the timestamp of the file
- **mv** = move command, works kind of like copy, but think of it like RENAME
- **rsync** -avh <directories go here> –delete = copies whatever is in the source directories to the target directory, and then deletes whatever doesn't match the original directory
- **/etc/passwd** = information of all the users available. Login, home, and shell, has 7 fields, read it with cat
    - cut -d: -f1 /etc/passwd
    - 1 = login id
    - 2 = password
    - 3 = user id
    - 4 = group id
    - 6 = home
    - 5 = user id info
    - 7 = default shell
    - **cut -d: -f1,6,7 /etc/passwd** = only the fields specified will be the ones printed, and it will print in the given order by the file being used. In other words, cannot rearrange fields
    - -d = used to designate the character that is a field marker
- **cat /etc/shells** = all available shells in the system
- id <user> = identifies the user by id number, group number, wheel(aka sudoers), etc
- /boot
- **ln** = link command, creates a hard link, which is akin to giving a file an alias, or a link to it

    - -s = soft link creation, creates a pointer link, think of it as a shortcut used in windows. It has its own unique ID, or Inode.
- **stat** = gives you system information about the specified file
- **unset** = will clear an user variable such as mymail=somethiing@gmail.com
- **shopt -s nocaseglob** = shell option command, in this particular case, we are telling the shell to ignore lower or upper case
- **rename** -v 's/.htm/.html/' *.htm = rename command in Ubuntu.
    - -v = will tell you what files were renamed to what
    - ‘ ‘ = encases what do you want to change
    - *.htm = tells you what to look for and change.
- **grep <word> <file>** = finds the specified word in the specified file, can be used as a search function.
    - -c = will make it so it counts how many it found
    - -i = will ignore capitalization
    - -n = will show the line number from the txt file
    - -v = will remove your word from the total
- **Sed** = line oriented, instead of going for words
    - Sed ‘p’ = will make it so each line is printed twice
    - Sed -n ‘p’ = prints twice, but no new lines
- Sudo adduser <user> = creates an user
- sudo userdel -r <user> = deletes an user
    - Confirm deletion by checking passwd, and the home directory
- runlevel = runlevel 0 shuts the pc, 6 will restart it, runlevel 5 is the GUI gnome
    - init executes specific runlevels
- who -r = shows runlevel

- \ + enter = this will move your cursor to a new line to continue typing a command
- Esc + . = brings the previous argument to the new command, it will also cycle to older arguments
- Ctrl + h = find and replace, super useful for scripts

- **expr** = command used to execute mathematical operations. Consists of operators and operands, operates on **integers only**
    - expr 4 + 3 //example of how to get this to work
    - expr 4+3 //this will print a string
    - expr 4 * 3 // this is for a multiplication
- bc = basic calculator
    - bc <<< "scale=2;5 / 4" //scale is for how many decimals you want, do note that this thing is executing a string. Or in other words this determines the **<u>precision</u>**
    - obase = output base = bc <<< 'ibase = 16;obase = 2; FFFF' = you can run conversion between bases
- awk "BEGIN {printf \"%.2f\n\", 100/30}"
    - Awk '{print}' filename = will print the contents of a file
    - awk -F',' '{print NR " - "$1 " "$7 " "$4 " "$9}' Agencies.csv | more -10
- Tr is a translator capable of manipulating files
-
    -

You can create mathematical variables with a simple equals sign, example: x=5.

Note, leave NO space between the equals sign and everything else

Postfix vs Prefix = a value will be updated after printing, and a value will be updated before printing.(a++ vs ++a), to implement this, use double parentheses

$((a++))

Gnome is a GUI

case begins a case check, esac marks the end of a case. Example

case $counter /*this is is a variable*/ in

1) conditions go here;;
2) Conditions go here;;

esac

Hidden files start with a dot. Such as .bash_history. Adding a -A to ls will make it so you can see hidden files with ls
ls -a shows hidden files AND implied directories

pwd stands for Present Working Directory. Note that the terminal starts at the Root directory.

Any name that starts with a $ is a variable. But names in capital, such as $PWD is a system variable so echo can be used to show what a variable has, such as echo $PWD this will show you the present working directory

You can use these system variables within other commands in order to complete paths or statements. Think of these variables as $PWD $HOME, etc…

Note that commands and variables are distinct. Commands access the system variables

Under a **ls** command:
Blue names are directories
White names are texts
Green names are executables

. is for implied directories

These are some shortcuts for names and commands
. pwd
..parent directory
~ home directory (its named tilde)

Deep blue files are Directories
Cyans are pointers. They hold the path of other locations
Whites are text files
Greens are executable'

Adding -l to ls will make it so the terminal brings out the long listing of the files in the directory specified.

cat /etc/os-release = shows you all the information about your OS including the “Pretty Name”

echo $SHELL = will print the current shell
cat /etc/shells = shows all the shells you can possibly run

Control + d = used to close the terminal
Control + shift + t = opens a new terminal

sudo = superuser do. This allows you to run commands as an “administrator”
sudo su = allows you to run commands as the root

When using cd, paths started with / will form an absolute path, they work from anywhere. And they are called ABSOLUTE paths Otherwise you will be using a relative path that depends on where you are located.

```
counter=1
while [ $counter -le 5 ]
do
        case $counter in
                1) t='\t';;
                2) t='\t\t';;
                3) t='\t\t\t';;
                4) t='\t\t';;
                5) t='\t';;
                esac
        echo -e $t$'\40'$'\40'$'\100'
```

```
        echo -e $t$'\40'$'\100'$'\100'$'\100'
        echo -e $t$'\100'$'\100'$'\100'$'\100'$'\100'
        echo -e $t$'\40'$'\40'$'\100'
        echo " "
        ((counter++))
done

</html>
```

Agencies CSV fields! Columns are fields, rows are records

1 Cust_id,

2 PGMS,

3 PGM_TYP,

4 AGENCY_NAME,

5 Religious,

6 ADDRESS,

7 CONTACT_NAME,

8 Gender,

9 PHONE,

10 BORO,

11 STATE,

12 ZIP,

13 Date_Of_Membership,

14 Credit_Limit.