**command line commands**

**$** - shell prompt

**ls** - list files/directories in working directory

**pwd** - print working directory

**cd** - change directory

**mkdir** - make directory

**touch** - create file

options modify the behaviours of commands

**ls -a** - list all files (hidden files too)

**ls -l** - lists all files in a long format

**ls -t** - order files and folders by the time they were last modified

**cp** - copy (one or more files into another folder)

**„\*”** - select group of files

**mv** - file, directory moving, renaming

**rm** - remove

**echo** - echos stdin to terminal as stdout

**cat** - outputs the content of a file to the terminal

**„>”** - directs content somewhere else overriding the original

„**>>”** - directs content somwhere else (adding) keeping the original

**|** - is a "pipe". The | takes the standard output of the command on the left, and pipes it as standard input to the command on the right. You can think of this as "command to command" redirection.

**sort** - takes the standard input and orders it alphabetically for the standard output.

**uniq** - stands for "unique" and filters out adjacent, duplicate lines in a file.

**grep** - stands for "global regular expression print". It searches files for lines that match a pattern and returns the results.

**grep -R** - searches all files in a directory and outputs filenames and lines containing matched results. -R stands for "recursive".

**grep -Rl** - searches all files in a directory and outputs only filenames with matched results. -R stands for "recursive" and l stands for "files with matches".

**sed** - stands for "stream editor". It accepts standard input and modifies it based on an expression, before displaying it as output data. It is similar to "find and replace".

Let's look at the expression 's/snow/rain/':

* s: stands for "substitution". it is *always* used when using sed for substitution.
* snow: the search string, the text to find.
* rain: the replacement string, the text to add in place.
* g expression, meaning "global". Here sed searches **forests.txt** for the word "snow" and replaces it with "rain", *globally*. All instances of "snow" on a line will be turned to "rain".

**nano** - is a command line text editor. It works just like a desktop text editor like TextEdit or Notepad, except that it is accessible from the command line and only accepts keyboard input.

1. Ctrl + O saves a file. 'O' stands for output.
2. Ctrl + X exits the nano program. 'X' stands for exit.
3. Ctrl + G opens a help menu.
4. clear clears the terminal window, moving the command prompt to the top of the screen.

**~/.bash\_profile** is the name of file used to store environment settings. It is commonly called the "bash profile". When a session starts, it will load the contents of the bash profile before executing commands.

* The ~ represents the user's home directory.
* The . indicates a hidden file.
* The name **~/.bash\_profile** is important, since this is how the command line recognizes the bash profile.

**alias** - command allows you to create keyboard shortcuts, or aliases, for commonly used commands.

The alias is then made available in the current session through source

**export** - makes the variable to be available to all child sessions initiated from the session you are in. This is a way to make the variable persist across programs.

**PS1** - is a variable that defines the makeup and style of the command prompt.

1. export PS1=">> " sets the command prompt variable and exports the variable. Here we change the default command prompt from $ to >>.
2. After using the source command, the command line displays the new command prompt.

**HOME** - variable is an environment variable that displays the path of the home directory.

**PATH** - is an environment variable that stores a list of directories separated by a colon.

**env** - command stands for "environment", and returns a list of the environment variables for the current user. Here, the env command returns a number of variables, including PATH, PWD, PS1, and HOME.

**Git**

**Git** - is a software that allows you to keep track of changes made to a project over time. Git works by recording the changes you make to a project, storing those changes, then allowing you to reference them as needed.

**init** - means initialize. The command sets up all the tools Git needs to begin tracking changes made to the project.

 A *Working Directory*: where you'll be doing all the work: creating, editing, deleting and organizing files

 A *Staging Area*: where you'll list changes you make to the working directory

 A *Repository*: where Git permanently stores those changes as different *versions* of the project

**git** **status** - As you write the screenplay, you will be changing the contents of the working directory. You can check the status of those changes

**git** **add** **filename** - In order for Git to start tracking scene-1.txt, the file needs to be added to the staging area.

**git** **diff** **filename** - we can check the differences between the working directory and the staging area

**git commit** - permanently stores changes from the staging area inside the repository.