Distros:

Starters:

1. **Ubuntu** *(And all it's derivatives)*
2. **Mint** *(And all it's derivatives)*

Others:

1. **CentOS** *(Great for servers apparently*)
2. **Debian**
3. **OpenSUSE**
4. **Manjaro** (Great for gamers)
5. **Elementary OS**
6. **Fedora**
7. **ZorinOS**
8. **Gentoo**
9. **Ubermix** *(Great for kids)*
10. **Tails** *(That one that Snowden used and praised)*

There are many more distros and probably more being developed, but this is a pretty good starting list, if I say so myself.

Common Directories:

/ **:** "Root" top of the system hierarchy.

/bin : Binaries and other executable programs.

/etc : System configuration files.

/home : Home directories, for each user account.

/opt : Optional or third party software.

/tmp : Temporary space, typically cleared on reboot.

/usr : User related programs.

/var : Variable data, most notably log files.

Comprehensive Directory Listing:

/boot : Files needed to boot the operating system.

/cdrom : Mount point for CD-ROMs.

/cgroup : Controls Groups hierarchy.

/dev : Device files, typically controlled by the operating system and the system administrators.

Basic Linux Commands:

ls - List directory contents.

cd - Changes the current directory.

pwd - Displays the present working directory.

cat - Concatenates and displays files.

echo - Display arguments to the screen.

man - Displays the online manual.

exit - Exits the shell or your current session.

clear - Clears the screen.

Navigating Man Pages:

Enter - Move down one line.

Space - Move down one page.

g - Move to the top of the page.

G - Move to the bottom of the page.

q - Quit.

man -k search_term - Allows you to search the man pages.

Directory Shortcuts:

. : This directory.

.. : The parent directory.

cd : Change to the previous directory.

To execute a command/file or script in your current directory use “ **cd ./name**"

Creating and Removing Directories:

mkdir [-p] directory - Create a directory

rmdir [-p] directory - Remove a directory.

rm -rf directory - Recursively removes directory.

• [**-p]** is not needed, but it's the parent directory command.

rmdir only removes empty directories, to remove a full directory and it's contents use "**rm -rf**".

!!! *Note: After removing something using the above commands, you'll no longer be able to retrieve them, use this at your own risk.* **!!!**

Decoding ls -l Output:

$ ls -l

Example: **-rw-rw-r-- 1 rick users 10400 Sep 27 09:30 funnydata.data**

Permission: **-rw-rw-r--**

Number of links: **1**

Owner name: **rick**

Group name: **users**

Number of bytes in the file: **10400**

Last modification time: **Sep 27 09:30**

File name: **funnydata.data**

Permissions:

$ ls -l

Example: **-rw-rw-r-- 1 rick users 10400 Sep 27 09:30 funnydata.data**

SYMBOL	TYPE
-	Regular file
d	Directory
l	Symbolic link

SYMBOL	PERMISSION

SYMBOL	PERMISSION
r	Read
w	Write
x	Execute

PERMISSION	FILE	DIRECTORY
Read (r)	Allows a file to be read.	Allows file names in the directory to be read.
Write (w)	Allows a file to be modified.	Allows entries to be modified within the directory.
Execute (x)	Allows the execution of a file.	Allows access to contents and metadata for entries.

PERMISSION SYMBOL:	PERMISSION CATEGORIES:
Symbol	**Category**
u	User
g	Group
o	Other
a	All

Groups:

- Every user is in at least one group.
- Users can belong to many groups.
- Groups are used to organize users.
- The "***groups***" command displays a user's groups.
- You can also used "***id -Gn***".

Secret Decoder Ring:

```
 __Type
|
▼   ▼--Group
-rw-rw-r-- 1 rick users 10400 Sep 27 09:30 funnydata.data
 ▲     ▲--Other
 |__User
```

Changing Permissions:

ITEM	MEANING
chmod	Change mode command
ugoa	User category user, group, other, all
+ - =	Add,subtract or set permissions
rwx	Read, Write, Execute

```
chmod g+w funnydata.data #Adds a permission

chmod g-w funnydata.data #Removes a permission

chmod g+wx funnydata.data #Adds the write and execute
permissions

chmod u+rwx,g-x funnydata.data #Adds the read, write,
execute permissions to "user" but removes the execute
permission from "group"

chmod a=r funnydata.data #Sets the permission read to
"all"

chmod o= #Leaving the "=" assignment operator empty
removes all the permissions, in this case from "others"
```

Numeric Based Permissions:

R	W	X	
0	0	0	Value off
1	1	1	Binary value for on
4	2	1	Base 10 value for on

It's always Read then Write then Execute, in that specific order.

OCAT	BINARY	STRING	DESCRIPTION
0	0	---	No permissions
1	1	--x	Execute only
2	10	-w-	Write only
3	11	-wb	Write and execute (2+1)
4	100	r--	Read only
5	101	r-x	Read and execute (4+1)
6	110	rw-	Read and write (4+2)
7	111	rwx	Read, write, and execute (4+2+1)

Order Has Meaning:

	U	G	O
Symbolic	rwx	r-x	r--
Binary	111	101	100
Decimal	7	5	4

Commonly Used Permissions:

SYMBOLIC	OCTAL

SYMBOLIC	OCTAL
-rwx------	700
-rwxr-xr-x	755
-rw-rw-r--	664
-rw-rw----	660
-rw-r--r--	644

When asked or from documentation you get asked to use "**777**" permission code ask yourself if there isn't a better way, since this gives everybody on the system access. It can cause trouble, as an example the injection of malicious code, the same goes for "**666**" permission code.

Working With Groups:

- New files belong to your primary group.
- The "**chgrp**" command changes the group.

```
groups #Will display the groups that you're in/assigned to
ricky sales #As an example

chgrp sales sales.data #Would add the sales group to the
"sales.data" file
```

Directory Permission Revisited:

- Permissions on a directory can affect the files in the directory.
- If the file permission look correct, start checking directory permissions.
- Work your way up to the root.

File Creation Mask:

- File creation mask determines default permissions.
- If no mask were used, permissions would be:
  - **777** for directories
  - **666** for files

The umask Command:

```
umask [-S] [mode]
```

- Sets the file creation mask to mode, if given.
- Use **-S** for symbolic notation.

How does umask work:

	DIRECTORY	FILE
Base Permission	**777**	**666**
Subtract Umask	**-022**	**-022**
Creations Permission	**755**	**644**

	DIRECTORY	FILE
Base Permission	**777**	**666**
Subtract Umask	**-002**	**-002**
Creations Permissions	**775**	**664**

Octal Substraction Is an Estimation:

	DIRECTORY	FILE
Base Permission	**777**	**666**
Subtract Umask	**-007**	**-007**
Creations Permission	**770**	**660***

Common umask modes:

- **022**
- **002**
- **077**
- **007**

Table of all the resulting permutations of umask:

OCTAL	BINARY	DIR PERMS	FILE PERMS
0	**0**	**rwx**	**rw-**
1	**1**	**rw-**	**rw-**
2	**10**	**r-x**	**r--**
3	**11**	**r--**	**r--**
4	**100**	**-wx**	**-w-**
5	**101**	**-w-**	**-w-**
6	**110**	**--x**	**---**
7	**111**	**---**	**---**

Special Modes:

- **umask 0022** is the same as **umask 022**.

- **chmod 0644** is the same as **chmod 644**.
- The special modes are:
    - **setuid**
    - **setgid**
    - **sticky**

Take note that the special modes are declared by prepending a character to the octal mode that normally use with **"umask"** or **"chmod"**.

Displaying the contents of files:

- **cat file**: Displays the contents of a file.
- **more file**: Browse through a text file.
- **less file**: More features than more.
- **head file**: Output the beginning or top portion of a file.
- **tail file**: Output the ending or bottom portion of a file.

Head and Tail:

- Displays only 10 lines by default.
- Change this behaviour with **(-n)**.
- **n** = number of lines.
    - **tail -15 file.txt** (would display 15 lines of either the beginning/bottom and/or top/bottom of a file)

View files in real time:

- **tail -f file** : Follow the file.
    - Displays data as its being written to the file.

  *(Using **tail** is the best option as it displays what is being written to the file as last-line.)*

The find command:

```
find [path] [expression]
```

- Recursively finds files in path that match **expression**. If no arguments are supplied, it finds all files in the current directory.

Find options (Expressions):

- **-name pattern**: Find files and directories that match pattern.

- **-iname pattern**: Like (-name), but ignores case, like upper and lower case characters.
- **-ls**: Performs an (ls) on each of the items found.
- **-mtime days**: Finds files that are **days** old.
- **-size num**: Finds files that are a certain **size**, using the **num** argument.
- **-newer file**: Finds files that are newer than the supplied **file** argument.
- **-exec command**: **{} , \ , ;** (Run command against all the files that are found)

Delete, copying, moving and renaming files:

Removing files:

- **rm file**: Remove file.
- **rm -r dir**: Remove the directory and it's contents.
- **rm -f file**: Force remove and never prompt for confirmation.

Copying files:

```
cp [source_file] [destination file]
#Copy source_file to destination_file

cp [src_file1] [dest_dir]
#Copy source_file to destination_directory
```

Cp options:

- **cp -i**: Run in interactive mode.
- **cp -r**: [source_directory] [destination]
    - Copy [source directory] recursively to destination.

Moving and renaming files:

- **mv**: Move or rename files and directories.
    - **mv** [source] [destination]
    - **mv -i** [source] [destination]
        - **-i** enables interactive mode.

Creating a collection of files:

```
tar - c|x|t f [tarfile] [pattern]
#Create, extract or list contents of a tar achive using
pattern if supplied
```

Tar options:

- **c** : Create a tar archive.
- **x** : Extract files from the archive.
- **t** : Display the table of contents.
- **v** : Be verbose.
- **z** : Use compression.
- **f** : Use this file.

Wilcards:

The two main wildcards:

- ***** - matches zero or more characters.
    - ***.txt** (Would find all files ending in the file extension of '.txt')
    - **a*** (Would find all files starting with the letter 'a')
    - **a*.txt** (Would find all the files that start with 'a' and end with '.txt')
- **?** - matches exactly one character. It also represents any character.
    - **?.txt** (Would find all files that only have one character before ending with '.txt')
    - **??.txt** (Would find all files that have two characters before ending with '.txt')
    - **a?** (Would find all the two character files that start with 'a')
    - **a?.txt** (Would find all the files that start with 'a' and then have one more character before ending with '.txt')

More wildcards - character classes:

- **[ ]** - A character class.
    - Matches any of the characters included between the brackets. Matches exactly one character.
    - **[aeiou]** (Would be used to find all the files starting with a vowel)
    - **ca[nt]*** (Would find all the files starting with 'ca' and then follow with either an 'n' or 't' or a key like '0')
    - Which would find files like:
        - can
        - cat
        - candy
        - catch

- **[!]** - Matches any of the characters NOT included between the brackets. Matches exactly one character.
    - **[!aeiou]*** (Would find al lthe files that do not start with a vowel)
        - Which would find files like:
            - baseball
            - cricket

More wildcards - ranges:

- Use two characters seperated by a hyphen to create a range in a characters class.
- **[a-g]***
    - Matches all files that start with a,b,c,d,e,f or g. So from 'a' through 'g'
- **[3-6]***
    - Matches all files that start with 3,4,5 or 6. So from '3' through '6'

Named character classes:

- **[[:alpha::]]** (Matches alphabetic letters both lower and uppercase letters)
- **[[:alnum:]]** (Matches alphanumeric characters and digits any upper or lowercase or decimal digits)
- **[[:digit:]]** (Matches the numbers and decimal from 0 to 9)
- **[[:lower:]]** (Matches any lowercase letters)
- **[[:space:]]** (Matches wide space, this means character such as spaces, tabs and newline characters)
- **[[:upper:]]** (Only matches uppercase letters)

Matching wildcard patterns:

- \ - Escape character. Use if you want to match a wildcard character.
- Match all files that end with a question mark:
    - *\?
    - Which would find a file like:
        - done?

I/O (Input & Output:

Input/Output types:

I/O NAME	ABBREVIATION	FILE DESCRIPTOR
Standard Input	**stdin**	**0**

I/O NAME	ABBREVIATION	FILE DESCRIPTOR
Standard Output	**stdout**	**1**
Standard Error	**stderr**	**2**

Redirection:

> : Redirects standard output to a file. (Overwrites *(truncating)* existing contents.)

>> : Redirects standard output to a file. (Appends to any existing contents.)

< : Redirects input from a file to a command.

& : Used with redirection to signal that a file descriptor is being used.

2>&1 : Combine stderr and standard output.

2>file : Redirect standard error to a file.

```
$ ls -l
#Output here

$ ls -l > files.txt
#All of the ouput of "ls -l" will be transcribed to the
file "files.txt"

$ ls >> files.txt
#Will append the regular "ls" output tp the file
"files.txt"
```

The null device:

The null device is a special file that will throw away anything that is fed to it.

>/dev/null : Redirect output to nowhere.

```
$ ls here not-here 2> /dev/null
here

$ ls here not-here > /dev/null 2>&1
```

Comparing Files:

Comparing the contents of files:

```
diff file1 file2 #Compare two files

sdiff file1 file2 #Side-by-side comparison

vimdiff file1 file2 #Highlight differences in vim
```

diff Output:

```
$ diff file1 file2
3c3
...

# LineNumFile1-Action-LineNumFile2
# Action = (A)dd (C)hange (D)elete
```

Searching in files using pipes:

The grep command:

grep - Display lines matching a pattern.

```
$ grep pattern file
```

grep Options:

-i : Perform a search, ignore case.

-c : Count the number of occurrences in a file.

-n : Precede the output with line numbers.

-v : Invert Match. Print lines that don't match.

The file command:

file file_name: Display the file type.

```
$ file sales.data
# sales.data: ASCII text

$ file *
# bin: directory

random.tar:
#POSIX tar archive
```

Pipes:

> *WIP*

| : Pipe symbol

command-output | command-input

The cut command:

cut [file] : Cut out selected portions of file. If file is omitted, use standard input.

cut Options:

-d delimiter : Use delimiter as the field separator.

-f N : Display the Nth field.

Transfering and copying files:

> *WIP*

Commandline:

- **SCP** : Secure copy. *(secure)*
- **SFTP** : SSH file transfer protocol. *(secure prefered)*
- **FTP** : Regular file transfer protocol. *(insecure)*

Graphical clients:

- **Cyberduck** (Mac)
- **FileZilla** (Mac, Windows, Linux)
- **Winscp** (Windows)

```
# Example commandline:

$ sftp linuxsvr
# You'll be asked for a password if there is one
$ quit # To disconnect from the server
```

Customizing the shell prompt:

- **Bash**, **ksh** and **sh** use **$PS1**
- **Csh**, **tcsh** and **zsh** use **$prompt**

Customizing the prompt with PS1:

- **\d** : Date in "Weekday Month Date" format "Tue May 15"
- **\h** : Hostname up to the first period.
- **\H** : Hostname.
- **\n** : Newline.
- **\t** : Current time in 24-hour HH:MM:SS format.
- **\T** : Current time in 12 hour HH:MM:SS format.
- **\@** Current time in 12-hour am/pm format.
- **\A** : Current time in 24-hour HH:MM format.
- **\u** : Username of the current user.
- **\w** : Current work directory.
- **\W** : Basename of the current working directory.
- **$** : If the effective UID is 0, a $, otherwise a $.

For the complete listing see the 'Bash man-page'

```
# Persist PS1 changes:
$ echo 'export PS1="[\u@\h \w]$"' >> ~/.bash_profile
```

Shell aliases:

Aliases:

- Shortcuts
- Use for long commands
- Use for commands you type often

Creating aliases:

alias [name [=value]]

Type "alias" to show all the created and in use aliases.

Use name=value to create a new alias.

```
# Example:
$ alias clr='clear'
```

Removing aliases:

- **unalias name** : Remove the "name" alias.

- **unalias -a** : Remove all aliases.

Persisting aliases:

- Add aliases to your personal initialization files.
    - .bash_profile

      ```
      # Quick way to add them to your
      'bash_profile'
      $ vi ~/.bash_profile
      ```

      Note that you will need to close and re-open the terminal(s) for your new alias to take, otherwise it will still not work.

Environment variables:

- Name/Value pairs.
- Can change how an application behaves.
- An example environment variable:
    - EDITOR=nano

Viewing environment variables:

```
$ printenv
# Results:
HOSTNAME=web01
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
USER=casper
MAIL=/var/spool/mail/vagrant
PATH=/usr/bin:/usr/sbin:/sbin:/usr/sbin
PWD=/home/casper
HOME=/home/casper

# You can check each individually:
$ printenv HOME
# Result:
/home/casper

# The following won't work, because of case-sensitivity:
$ printenv home

# Environment variables are uppercase by default. Although
there are some lowercase ones, but you'll rarely come
across those.

$ echo $HOME
```

```
# Result:
/home/casper

# If ther are too many environment variables and they
scroll off your terminal screen, you can use the follwing
piping command to open it in "less", which is a pager
utility.

$ printenv | less
```

Creating environment variables:

Syntax:

```
$ export VAR="value"

# Do not use a space between the equal sign and the value,
otherwise you will encounter an error.
```

Examples:

```
$ export EDITOR="vi"

# This will set the 'EDITOR' variable and setting it so
use vim as the default editor. If there already was a
'value' attached to the "EDITOR variable, then it will be
overwritten.

$ export TZ="US/Pacific"

# This will set the timezone to "US Pacific" time.
```

Removing environment variables:

Syntax:

```
$ unset VAR
```

Example:

```
$ unset TZ
```

Persisting environment variables:

```
$ cat~/.bash_profile export TZ="US/Central"
```

For extra information never forget that you can check the man pages.

Processes and job control:

Listing processes and information:

- **ps** : Display process status.
- **ps options:**
    - **-e** : Everything, all processes.
    - **-f** : Full format listing.
    - **-u username** : Display username's processes.
    - **-p pid** : Display information for PID

Common ps commands:

- **ps -e** : Display all processes.
- **ps -ef** : Display all processes, full.
- **ps -eH** : Display a process tree.
- **ps -e --forest** : Display a process tree.
- **ps -u username** : Display a user's processes.

Other ways to view processes:

- **pstree** : Display processes in a tree format.
- **top** : Interactive process viewer.
- **htop** : Interactive process viewer. *(I personally recommend this one.)*

Background and foreground processes:

- **command &** : Start command in background.
- **Ctrl -c** : Kill the foreground process.
- **Ctrl -z** : Suspend the foreground process.
- **bg [%num]** : Background a suspended process.
- **fg [%num]** : Foreground a background process.
- **kill** : Kill a process by job number or PID.
- **jobs [%num]** : List jobs. *(The "+" sign signifies the most recent job, and the "-" sign signifies the last job.)*

Killing processes:

- **Ctrl -c** : Kills the foreground processes.
- **kill [-sig] pid** : Send a signal to a process.
- **kill -l** : Display a list of signals.

Scheduling repeated jobs with Cron:

Cron:

- **cron** : A time based job scheduling service.
- **crontab** : A program to create,read,update, and delete your job schedules.
- Use **cron** to schedule and automate tasks.

Crontab format:

```
* * * * * command
| | | | |
| | | | +-- Day of the week (0-6)
| | | +---- Month of the year (1-12)
| | +------ Day of the month (1-31)
| +-------- Hour (0-23)
+---------- Minute (0-59)
```

Example crontab entries:

```
# Run every monday at 07:00
$ 0 7 * * 1 /opt/sales/bin/weekly-report

# Run every 30 minutes:
$ 0,30 * * * * /opt/acme/bin/half-hour-check

# Another way to do the same thing.
$ */2 * * * * /opt/acme/bin/half-hour-check

# Run for the first 5 minutes of the hour.
$ 0-4 * * * * /opt/acme/bin/first-five-minutes
```

Redirecting output example:

```
# Run at 02:00 every day and
# send output to a log file.
$ 0 2 * * * /root/backupdb > /tmp/db.log 2>&1
```

Crontab shortcuts:

SHORTCUT	NUMERICAL
@yearly	0 0 1 1 *
@annually	0 0 1 1 *
@monthly	0 0 1 * *
@weekly	0 0 * * 0
@daily	0 0 * * *
@midnight	0 0 * * *
@hourly	0 * * * *

Not all of theses will work, depending on your distribution or tools, to check you should have a look at "cron man"

Using the crontab command:

- **crontab file** : Install a new crontab from file.
- **crontab -l** : List your cron jobs.
- **crontab -e** : Edit your cron jobs.
- **crontab -r** : Remove all of your cron jobs.

Switching users and running commands as others:

The su command:

- **su [username]** : Change user ID or become superuser.
- **su options:**
  - A hyphen [-] is used to provide an environment similar to what the user would expect had the user logged in directly.
  - **-c command** : Specify a command to be executed.

whoami example:

```
# The whoami command displays yor current account
name.

$ whoami
casper

$ su oracle
Password:

$ whoami
oracle
```

Sudo - super user do:

- **sudo** : Execute a command as another user, typically the superuser.

Using sudo:

- **sudo -l** : List available commands.
- **sudo command** : Run command as root.
- **sudo -u root command** : Same as above.
- **sudo -u user command** : Run as user.
- **sudo su** : Switch to the superuser account.
- **sudo su -** : Switch to the superuser account with root's environment.
- **sudo su - username** : Switch to the username account.
- **sudo -s** : Start a shell.
- **sudo -u root -s** : Same as sudo-s.
- **sudo -u user -s** : Start a shell as user.

Changing the sudo configuration:

- **visudo** : Edit the /etc/sudoers file. *(This command has to be started as root and it will using the vim editor to edit.)*

Suodoers format:

```
user host=(users)[NOPASSWD:]commands

adminuser ALL=(ALL)NOPASSWD:ALL
casper linuxsvr=(root) /etc/init.d/oracle
```

Shell history and tab completion:

Shell history:

- Executed commands are added to the history.
- Shell history can be displayed and recalled.
- Shell history is stored in memory and on disk
    - **~/.bash_history**
    - **~/.history**
    - **~/.histfile**

History command:

- **history** : Displays the shell history.
- **HISTSIZE** : Controls the number of commands to retain in history. *(export HISTSIZE=1000)*

! Syntax:

- **!N** : Repeat command line number N.
- **!!** : Repeat the previous command line.
- **!string** : Repeat the most recent command starting with "string".
- **! : N** : [Event] [Separator] [Word].
- **!** : Represent a command line (or event).
    - **!** = The most recent command line.
    - **!** = !! .
- **:N** : Represents a word on the command line.
    - **0** = command, 1 = first argument, etc.

(The exclamation mark is often referenced to as "bang")

! Syntax examples:

```
$ head files.txt sorted_files.txt notes.txt
<Output from head command here>
$ !!
$ head files.txt sorted_files.txt notes.txt
<Output from head command here>
$ vi !:2
vi sorted.files.txt
<vi editor starts>
```

Some more ! syntax:

- **!^** : Represents the first argument.
  - !^ = !:1
- **!$** : Represents the last argument.

  ```
  $ head files.txt sorted_files.txt notes.txt
  # !^ = files.txt
  # !$ = notes.txt
  ```

Searching shell history:

- **Ctrl-r** : Reverse shell history search.
- **Enter** : Execute the command.
- **Arrows** : Change the command.
- **Ctrl-g** : Cancel the search.

Tab completion:

- **Tab** autocompletion
  - Commands
  - Files, directories, paths
  - Environment variables
  - Usernames (~)

Installing and managing software:

Package:

- A collection of files.
- Data / Metadata
  - Package description
  - Version
  - Dependencies

Package Manager:

- Install, upgrades, and removes packages.

- Manages dependencies.
- Keeps track of what is installed.

Installing software on RPM distros:

(RPM stands for "RedHat Package Manager")

A brief list of distros that use RPM:

- RedHat
- CentOS
- Fedora
- Oracle Linux
- Scientific Linux

yum:

- **yum search string** : Search for string.
- **yum info [package** : Display info.
- **yum install [-y] package** : Install package. *(**-y** will automatically answer "yes" to all prompts from '**yum**' therefore only do this, if you're certain about the package.)*
- **yum remove package** : Remove package.

(To install or remove a package/software you will need 'root' access)

rpm:

- **rpm -qa** : List all installed packages.
- **rpm -qf /path/to/file** : List the file's packages.
- **rpm -ql package** : List package's files.
- **rpm -ivh package.rpm** : Install package.
- **rpm -e package** : Erase (uninstall) package

Installing software on DEB distros:

- Debian
- Linux Mint
- Ubuntu

APT - Advanced Packaging Tool:

- **apt-cache search string** : Search for string.
- **apt-get install [-y] package** : Install package. *(**-y** will automatically answer "yes" to all prompts from '**APT**' therefore only do this, if you're certain about the package.)*
- **apt-get remove package** : Remove package, leaving configuration.
- **apt-get purge package** : Remove package, deleting configuration.
- **apt-cache show package** : Display information about the package.

(To install or remove a package/software you will need 'root' access)

dpkg:

- **dpkg -l** : List installed packages.
- **dpkg -S /path/to/file** : List file's packages.
- **dpkg -L package** : List all files in package.
- **dpkg -i package.deb** : Install package.

> *More to come?*