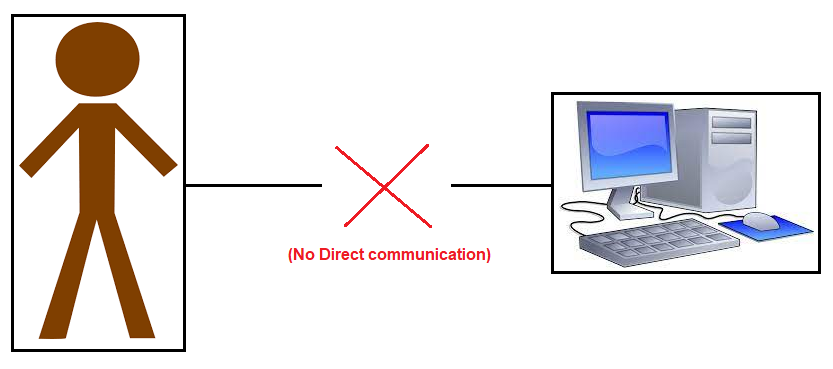
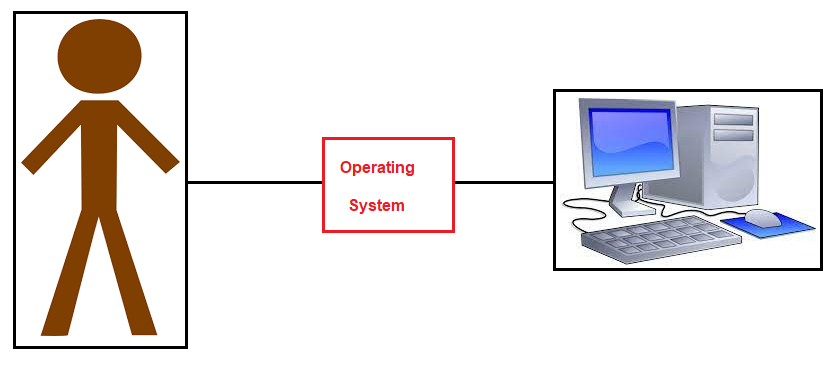
What is computer? A computer is a machine that can be programmed to carry out sequences of arithmetic or logical operations automatically.

What is Human? It’s a user who would be using computer

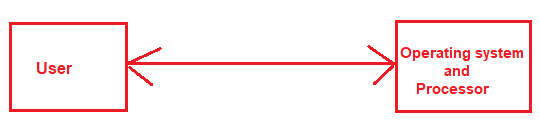
What’s the problem? Computer understand “0” and “1”, and human understand high level language. So, there is no possibility of direct communication.



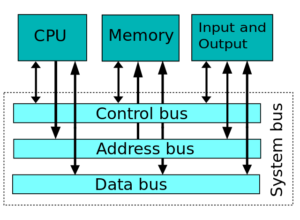
To solve above problem, we have introduced vice-versa translator which convert human language to machine language, and now communication is possible.



Operating system is a software component, so let me place it correctly.



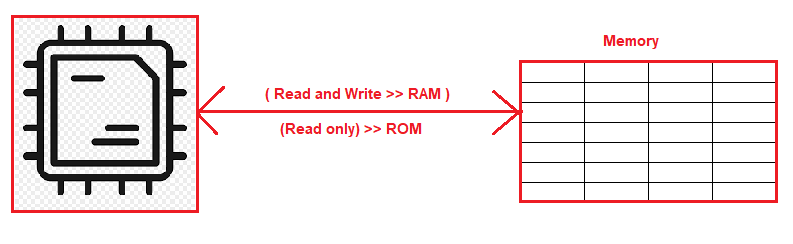
SO, now let me start operating system, but before that let me understand few architectural concept(s).



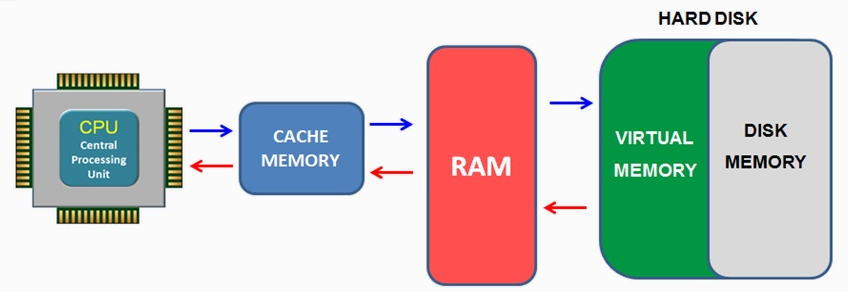
What is Processor? Information processor, a system which takes information in one form and transforms it into another form by an algorithmic process

What is memory? **Memory** refers to the processes that are used to acquire, store, retain, and later retrieve information. There are three major processes involved in **memory**: encoding, storage, and retrieval.

RAM vs ROM?



Relationship between RAM and ROM>> SWAP memory

Note: Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU

Now, let me start OS.

|  |  |
| --- | --- |
| BIOS | Here connectivity test is performed between input and output devices. |
| MBR | it loads boot program from ROM to RAM |
| GRUB | It displays menu interface that list various available operating system |
| Kernel | it is responsible for namespace and c-groups |
| INIT | it's master process, it's parent for all process with pid 1 |
| Run level | Run level is a state of init where a group of process are defined to start at OS start |

**1. BIOS**

**In this process, processor send test signal to input and output devices.**

BIOS stands for Basic Input/Output System. In simple terms, the BIOS loads and executes the Master Boot Record (MBR) boot loader.

When you first turn on your computer, the BIOS first performs some integrity checks of the HDD or SSD.

Then, the BIOS searches for, loads, and executes the boot loader program, which can be found in the Master Boot Record (MBR). The MBR is sometimes on a USB stick or CD-ROM such as with a live installation of Linux.

Once the boot loader program is detected, it's then loaded into memory and the BIOS gives control of the system to it.

**2. MBR**

MBR stands for Master Boot Record and is responsible for loading and executing the GRUB boot loader.

The MBR is located in the 1st sector of the bootable disk, which is typically /dev/hda, or /dev/sda, depending on your hardware. The MBR also contains information about GRUB, or LILO in very old systems.

**3. GRUB**

Sometimes called GNU GRUB, which is short for GNU GRand Unified Bootloader, is the typical boot loader for most modern Linux systems.

The GRUB splash screen is often the first thing you see when you boot your computer. It has a simple menu where you can select some options. If you have multiple kernel images installed, you can use your keyboard to select the one you want your system to boot with. By default, the latest kernel image is selected.

The splash screen will wait a few seconds for you to select and option. If you don't, it will load the default kernel image.

In many systems you can find the GRUB configuration file at /boot/grub/grub.conf or /etc/grub.conf. Here's an example of a simple grub.conf file:

#boot=/dev/sda

default=0

timeout=5

splashimage=(hd0,0)/boot/grub/splash.xpm.gz

hiddenmenu

title CentOS (2.6.18-194.el5PAE)

root (hd0,0)

kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/

initrd /boot/initrd-2.6.18-194.el5PAE.img

**4. Kernel**

The kernel is often referred to as the core of any operating system, Linux included. It has complete control over everything in your system.

In this stage of the boot process, the kernel that was selected by GRUB first mounts the root file system that's specified in the grub.conf file. Then it executes the /sbin/init program, which is always the first program to be executed. You can confirm this with its process id (PID), which should always be 1.

The kernel then establishes a temporary root file system using Initial RAM Disk (initrd) until the real file system is mounted.

**5. Init**

At this point, your system executes runlevel programs. At one point it would look for an init file, usually found at /etc/inittab to decide the Linux run level.

The available run levels:

Run level 0 is matched by poweroff.target (and runlevel0.target is a symbolic link to poweroff.target).  
  
Run level 1 is matched by rescue.target (and runlevel1.target is a symbolic link to rescue.target).  
  
Run level 3 is emulated by multi-user.target (and runlevel3.target is a symbolic link to multi-user.target).  
  
Run level 5 is emulated by graphical.target (and runlevel5.target is a symbolic link to graphical.target).  
  
Run level 6 is emulated by reboot.target (and runlevel6.target is a symbolic link to reboot.target).  
  
Emergency is matched by emergency.target.

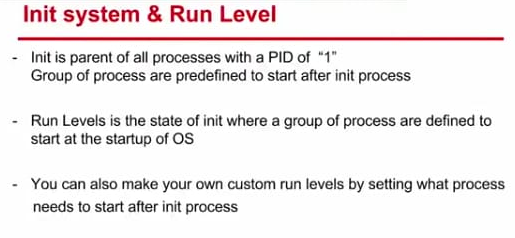
systemmd will then begin executing runlevel programs.

**6. Runlevel programs**

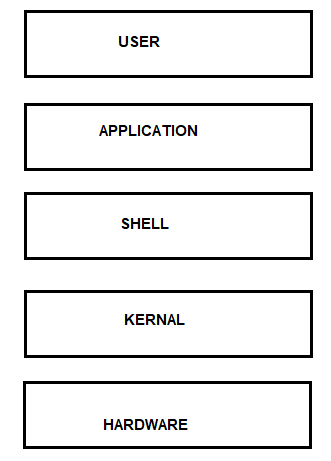
Depending on which Linux distribution you have installed, you may be able to see different services getting started. For example, you might catch starting sendmail …. OK.

These are known as runlevel programs, and are executed from different directories depending on your run level. Each of the 6 runlevels described above has its own directory:

* Run level 0 – /etc/rc0.d/
* Run level 1 – /etc/rc1.d/
* Run level 2  – /etc/rc2.d/
* Run level 3  – /etc/rc3.d/
* Run level 4 – /etc/rc4.d/
* Run level 5 – /etc/rc5.d/
* Run level 6 – /etc/rc6.d/



Now, my system if fully up, and now user is able to communicate to computer, and even I can login to OS.

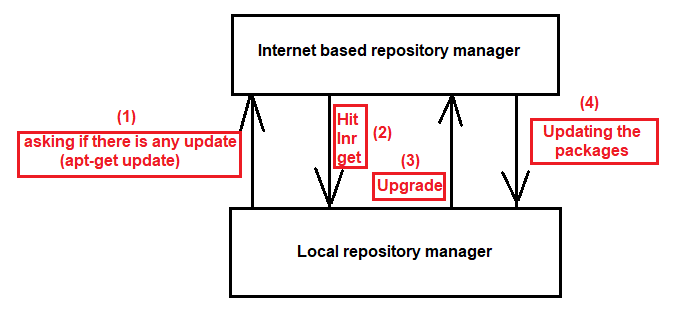


Apt-get update: Apt command work on the database of available package. Apt-get update command will refresh the local database of available software. Once you run this command you can see information is getting fetched from various server.

root@ip-172-31-1-140:~# apt-get update Igr:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB] Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal InRelease Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB] Reading package lists... Done

Hit: There is no change is package version. Igr: Package is being ignored Get: New version of package is available.

Apt-get upgrade - Perform an upgrade



root@ip-172-31-1-140:~# apt-get upgrade Preparing to unpack .../23-open-vm-tools\_2%3a11.1.5-1~ubuntu20.04.2\_amd64.deb ... Unpacking open-vm-tools (2:11.1.5-1~ubuntu20.04.2) over (2:11.0.5-4) ... Setting up libglib2.0-0:amd64 (2.64.6-1~ubuntu20.04.3) ...

There is a very famous saying: **Necessity is the mother of invention**.

Generally, there is a problem with IT industry “How we can communicate efficiently with hardware.”

Can we directly talk to processor? Yes, an embedded programme running directly on a microcontroller, or one with a minimal RTOS (real-time operating system) may be able to talk to hardware directly. As for how - you would have to know the precise details of how your hardware is interfaced to the computer.

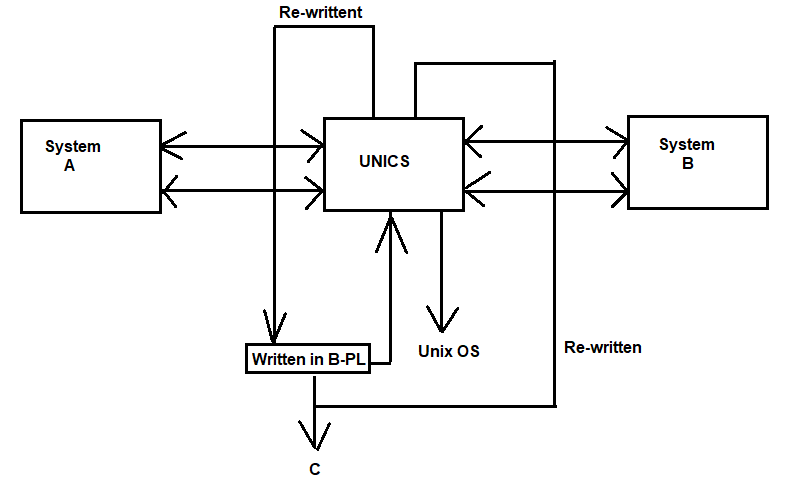
Note: Computer understand only 0/1, and we understand high level language, to overcome this communication gap operating system introduce.

You can talk to computer directly if you understand 0/1; but generally, user is not aware of it.

Thought? Can I design a system with can convert user input to 0/1? Link: <https://www.rapidtables.com/convert/number/binary-to-ascii.html>

01000101 01111000 01100001 01101101 01110000 01101100 01100101 > Example

**History of Unix OS**



### Linux – The Operating System

* RedHat Linux, Ubuntu, Centos, SUSE etc are called as distributions.
* Distribution is Linux + software suite of applications,developer tools
* In this essence Linux is core of the operating system: kernel

#### Layers of Abstraction in Linux



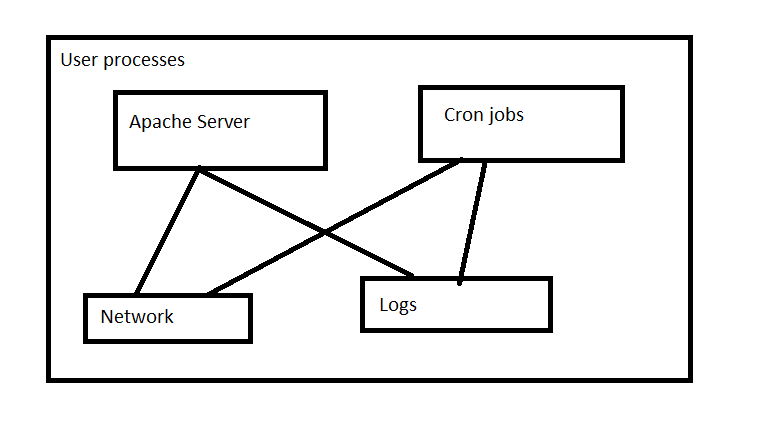
Layers are

1. Hardware, 2) Kernel and 3) User Process

Kernel is in charge for managing

* + Processes
  + Memory
  + Device Drivers
  + System Calls
    - fork
    - exec

User Space: Kernel allocates memory for user processes and this is called as userspace.

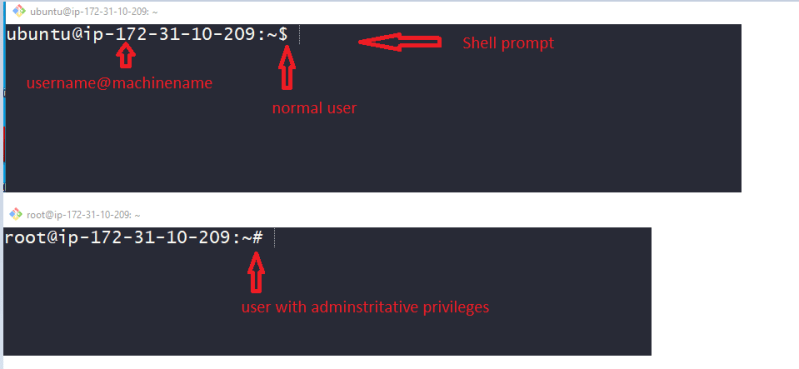


**User**: A user is an entity that can run processes and own files.

### Shell and Terminal

* when we speak of the command line we are referring to shell. Shell is a program takes commands and passes them to OS to carry out.
* Almost all the distributions a shell program called as bash is supplied.
* To interact with shell, we need a terminal

Let’s create a linux instance and login into that



### Standard Input and Standard Output

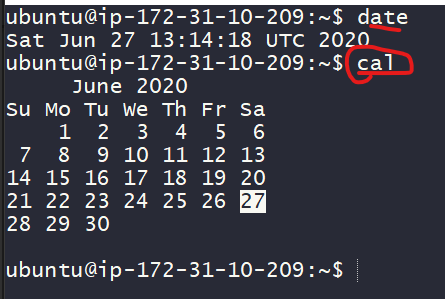
* Linux Processes use I/O streams to read and write data.
* Streams are very flexible, the source of input stream can be a file, device or it can be even the output stream of other process

### Linux Commands

In the shell prompt we generally execute commands. Let’s execute some simple commands

date

cal



Basic command syntax

<command> <args>

echo hello

Arguments of two types

1) Positional arguments:

* <command> <arg1> <arg2> ...
* cp 1.txt 2.txt

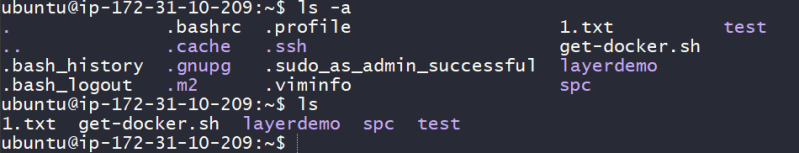
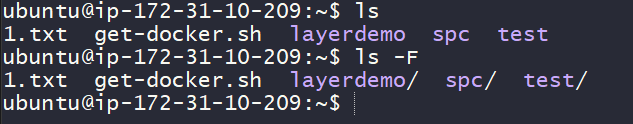
2) Named arguments:

* <command> --<argname> <argumentvalue>
* **File –name filename.txt**

**ls**: this command is used to list the contents of the directory

ls

ls -a



**touch**: this command creates an empty file

touch 2.txt

**cp**: this command copies files

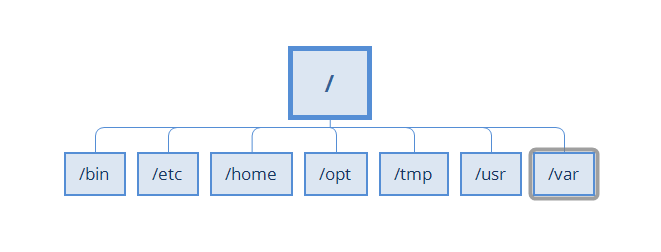
cp file1 file2

**mv**: this command moves (rename) the files

**rm**

**echo**

#### Linux directory hierarchy



* / => root directory
* /bin => Binaries and other executables
* /etc => system configuration files
* /home => home directories
* /opt => optional or third party softwares
* /tmp => Temporary space
* /usr => User related programs
* /var => variable data, log
* experiment with mkdir, rmdir, less, file, head, tail, exit

### Environment and Shell variables

Shell variables: Shell can temporarily store variables called as shell variables

<VAR-NAME>=<value>

TOPIC=linux

To access variable use $

echo $TOPIC

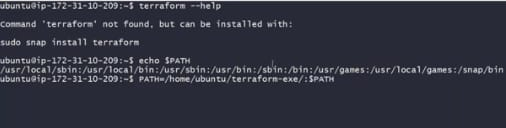
Environmental variable: This is also like shell variable, but its not specific to the shell.

export <variable>

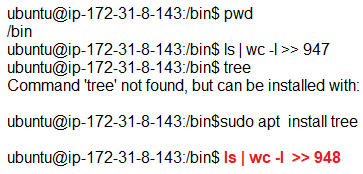
There is one important environmental variable which is PATH. PATH variable will inform linux to find the commands

echo $PATH





What is command? Command is an application that is designed to perform specific task.



Why we need to do ./ or sh before running any command? Because we are not installing them in their bin directory.

Demo 1:

1.) Installing terraform as a command.

2) Describing linux distribution.

Linux = Unix (+) Core utilities.



**Linux Useful utils**

Which Shell are you running?Execute the command echo $0

Aborting and suspending the execution of linux programs Control-Z suspends the program Control-C aborts the program

Redirecting output of a command to a file instead of terminal (stdout) command > file Eg: ping -c 4 google.com > pingresults.txt If you want to append the text to existing file use >> Eg: ping -c 10 google.com >> pingresults.txt

Redirecting input can be done using < eg: head < /proc/cpuinfo

Common Errors

* NO Such file or directory
* file exists
* Not a Directory, Is a directory
* NO space left of device
* Permission Denied

What is softlink (symbolic link) in linux and what is hardlink?

* Softlink (symbolic) is an actual link to original file whereas hard link is a mirror copy of original file.
* If you delete the original file, soft link has no value (since t whatever file it points no longer exists)
* If you delete the original file, hard link will still has the data of the original file

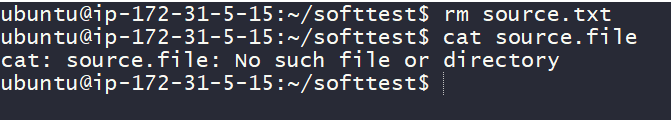
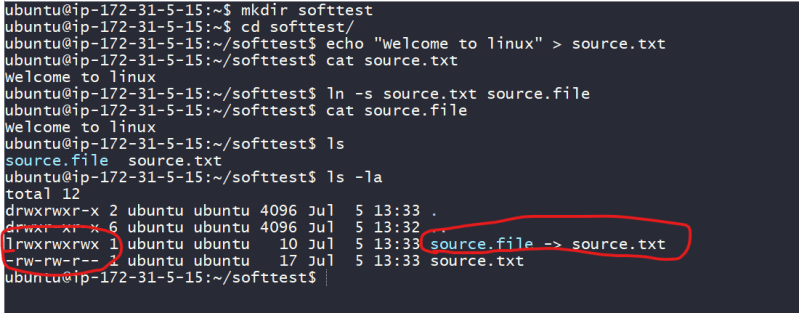
Softlink

* Can cross the file systems
* Allows you to link between directories
* permissions will not be updated
* has only the path to original file not contents

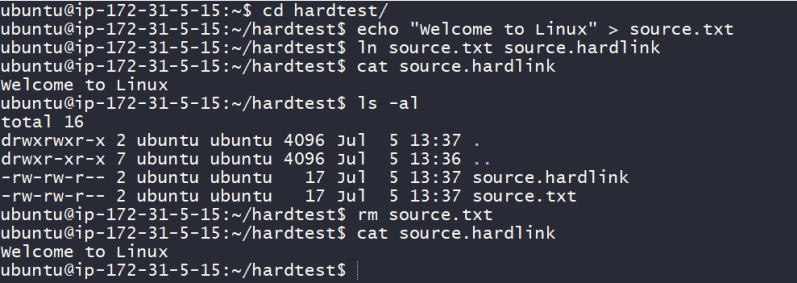
Hard link

* Can’t cross file system boundaries (it exists on same filesystem)
* can’t link directories
* permissions will be updated if we change the source
* It has contents of original file

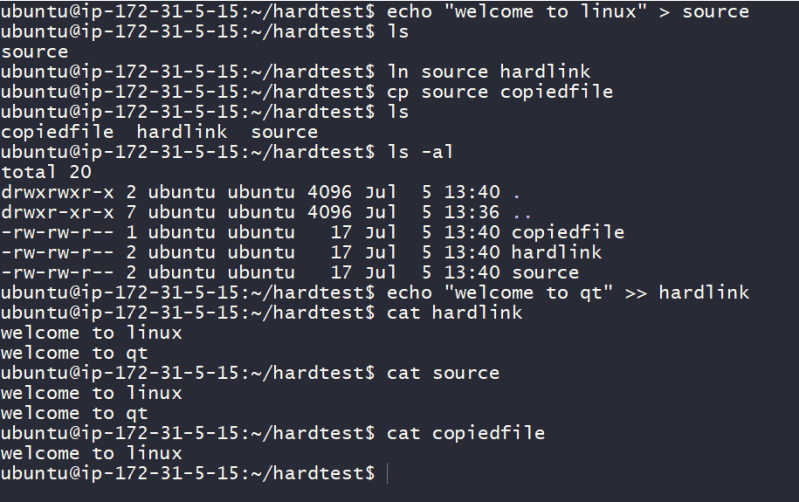
Create a Symlink or softlink



Create a hardlink



What will be difference between hard link and normally copied file?



Identifying Commands

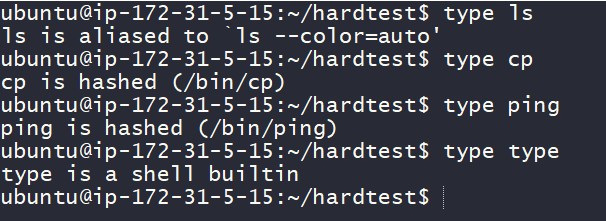
type: Type is a shell built-in command which displays the kind of command once you give a particular command

type ls

type cp

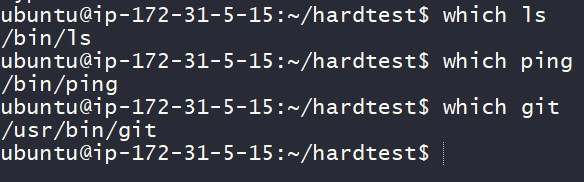
type ping

type type



which-Display an Executable location

which ls



Getting a Commands Documentation

help: bash has a built-in help facility for each of the shell builtins.

help cd

–help: Many executable programs support a –help option displays the description of commands

mkdir --help

man: Most executable programs intended for command line usage will provide manual

man <command>

man ls

apropos-Display appropriate commands

apropos partition

info command: This command will show mauals with hyperlinked much like web docs

**Pipelines**

Is a capability of commands to read output of command as input to other command

Generally, operator | is referred as pipe

command1 | command2

Pipelines are often used to perform complex operations on data. It is possible for you to put several commands together into a pipeline

ls -al /bin /usr/bin | sort | less

ls -al /bin /usr/bin/ | sort | uniq | wc -l

**tee**

tee command will read from stdin and send the output to stdout and files

ls -al /bin /usr/bin | tee ls.txt | sort | uniq | wc -l

### SEEING THE WORLD AS SHELL SEES

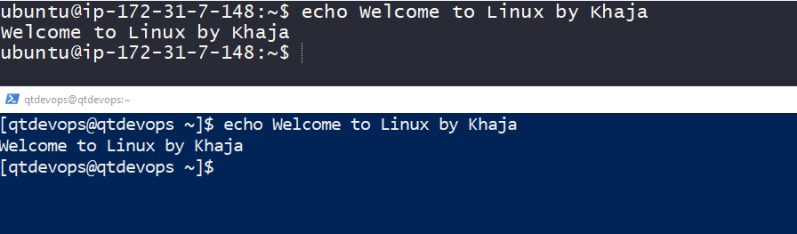
Let’s look at some of the magic that occurs in the command line when we press ENTER key.

#### Expansion

Each time we type a command & press Enter key, bash performs several substitutions upon text before command is executed. The process that makes it happen is called expansion

To demonstrate this let’s try

echo Welcome to Linux by Khaja



Note: Images with terminal in black (git bash) is connected to ubuntu 18 and Images with terminal in blue (PowerShell) is connected to centos 8.1

Now let’s execute the following



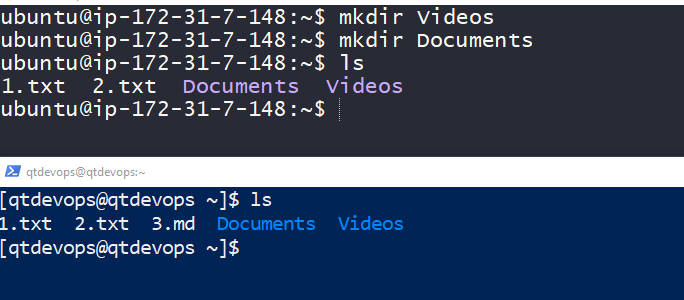
What happened? \* is a wild card character, so your bash will try to find/match any characters in the files present in current working directory (cwd).

So initially when we executed there were no files in cwd, so it printed \*. Then we created two files and execute echo \* again, now \* will match any file name in cwd so it printed the file names

Pathname Expansion: The mechanism by which wild cards work is called as pathname expansion

Let’s experiment with the following

create two directories Videos and Documents in the cwd

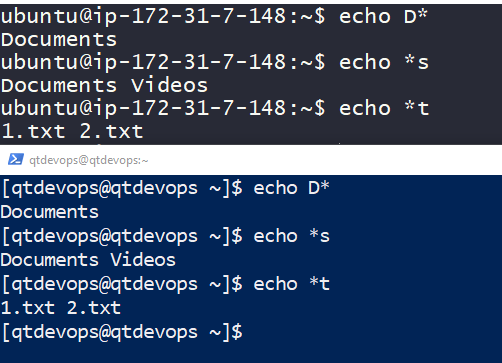


Let’s use pathname expansions

echo D\*

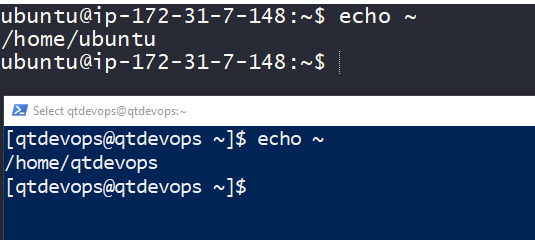
echo \*s

echo \*t





Tilde Expansion: Tilde ~ expans into the name of home directory of the current user



Arithmetic Expansion: Shell allows you to expand arithmetic’s and the syntax would be $((expression))

Operators are

**+** : Addition

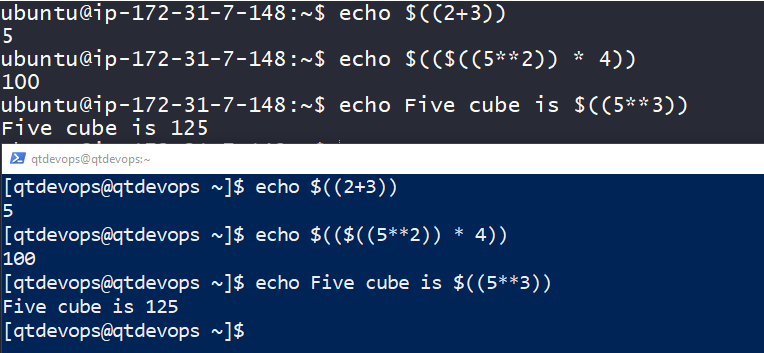
**–** : Subtraction

**\*** : Multiplication

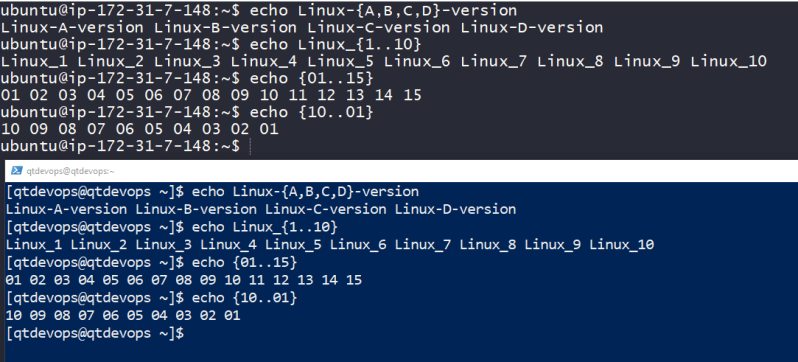
**/** : Division

**%** : Modulo (remainder)

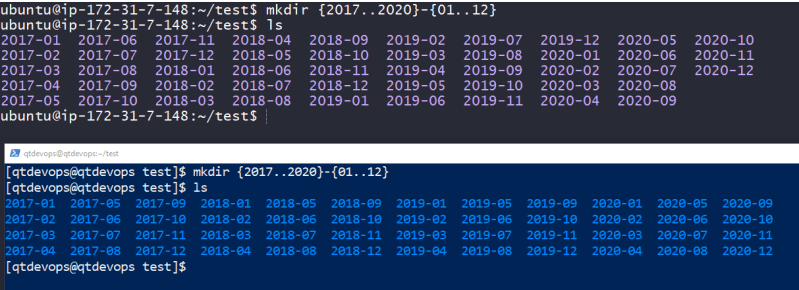
Exponentation



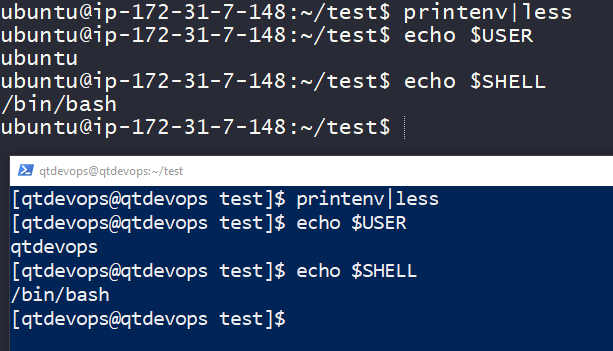
Brace Expansion: Brace expansion can create multiple text strings



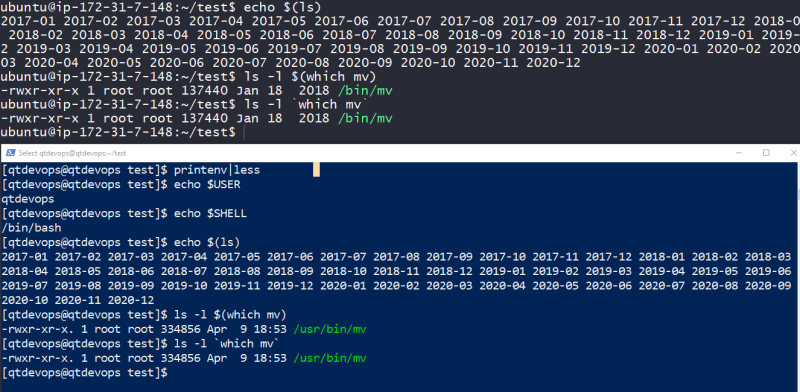
Using brace expression let’s try to create folder for every month from year 2017 to 2020 in the format of yyyy-mm



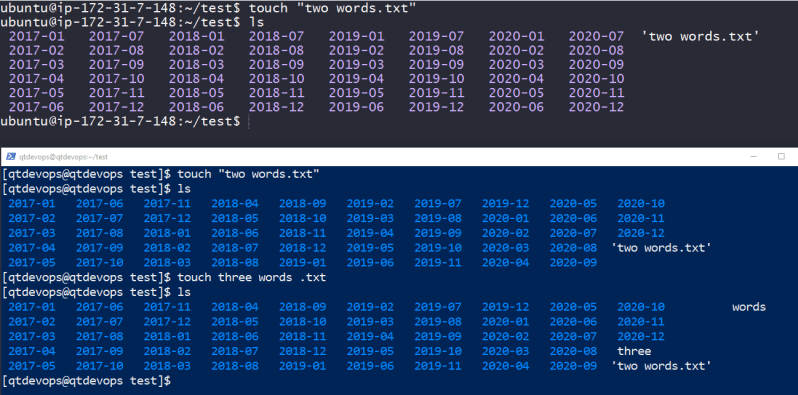
Parameter Expansion:



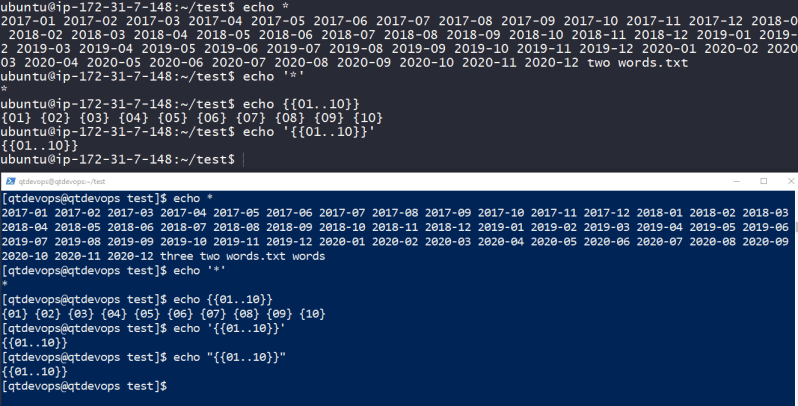
Command Substitution: this allows us to use output of a command as expansion



Quotes



Supress expansion:



**Escaping Characters**

Using Backslash escape sequences 

Backslash escape sequences

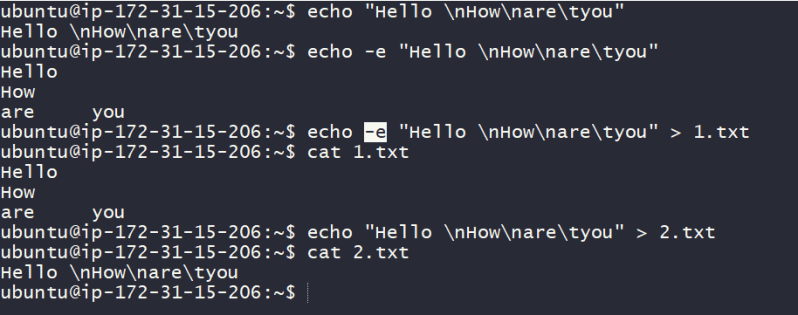
\a => Bell

\b => Backspace

\n => new line

\r => Carriage return

\t => Tab return



**Keyboard tricks**

bash uses a library called *ReadiLine* to implement command line editing.

Cursor Movement Commands

CTRL-A : Move the cursor to beginning of the line

CTRL-E : Move the cursor to the end of line

CTRL-F: same as right arrow key

CTRL-B: same as left arrow key

ALT-F: Move cursor forward by one word

ALT-B: Move cursor backward by one word

CTRL-L: same as using clear command

Modifying Text:

CTRL-D: Delete the character at cursor location

CTRL-T: exchanges the character at the cursor location with one preceding it

ALT-T: exchanges the word at the cursor location with one preceding it

ALT-U: converts the characters from cursor location to the end of the line to uppercase

ALT-L: converts the characters from cursor location to the end of the line to lowercase

Killing and yanking (Cutting and Pasting) Text

CTRL-K: Kill text from the cursor location to the end of the line

CTRL-U: Kill text from the cursor location to the begining of the line

CTRL-Y: Yank the text from kill-ring (clipboard) and insert at current cursor location

ALT-D: kill text from cursor location to the end of current word

ALT-backspace: kill text from cursor location to the begining of current word

COMPLETION: shell can help you through a mechanism called as completion. Completion occurs when your press **TAB** key while typing command

In Recent versions of bash has a facility called as programmable completion.

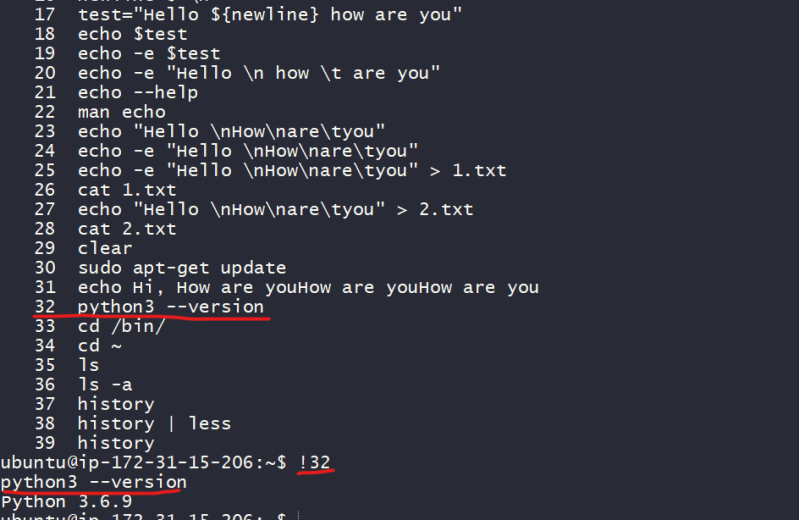
Using History

Bash maintains the history of commands that have been entered. This list of commands are kept in your home directory in a file called as **bash\_history**

commands to try

history

history|less



Commands

CTRL-P or up-arrow: Move up to the previous history entry

CTRL-N or down-arrow: Move to the next history entry

Sequence Action

!!: Repeat the last command

!number: Repeat history list item *number*

!string: Repeat last history item starting with string

!?string: Repeat last history item containing string

### Permissions

Traditionally linux is a multiuser system which differs from MS-DOS which is a multi-tasking system.

Since more than one person can be using the computer at the same time, we need to some sort of permissions in linux.

So, let’s look at these commands which are essential part of linux system security

id: Display User identity

chmod: Changes file’s mode

umask: Set the default file permissions

su: Run a shell as another user

sudo: Execute a command as another user

chown: Change file’s owner

chgrp: Change a file’s group ownership

passwd: Change a user’s password

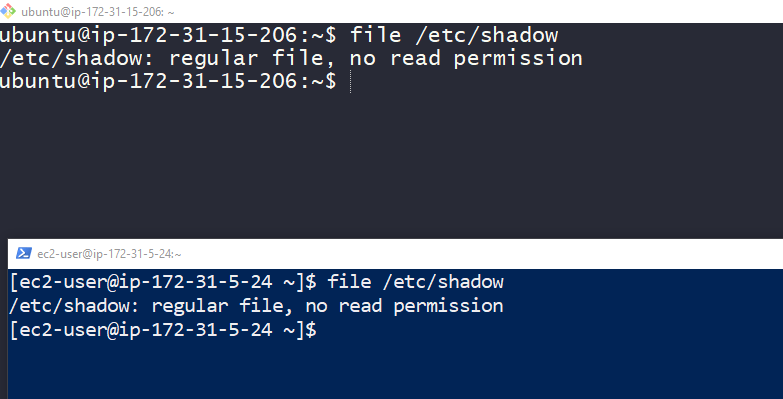
useradd: add the user using a built-in linux

adduser: adds the user using a perl script that internally useradd

groupadd: creates a new group

### OWNERS, Group members and Everybody else

note: Throughout this series , terminal with blue background is connected to Redhat and terminal with black ground is connected to ubuntu 18 

Let’s execute a simple check on /etc/shadow 

The reason for this error message is , we don’t have permission to read this file

In this it is very critical to file, to know how to gain control access over file, give permissions to special groups and all the others (everybody)

Let’s find the information about our identity using id command 

Id’s commands output. When a user is created in linux, users are assigned a number called as uid (user id), The user is assigned a group ID (gid) and may belong to additional groups

All of this information in linux comes from couple of files

/etc/passwd: User accounts are defined

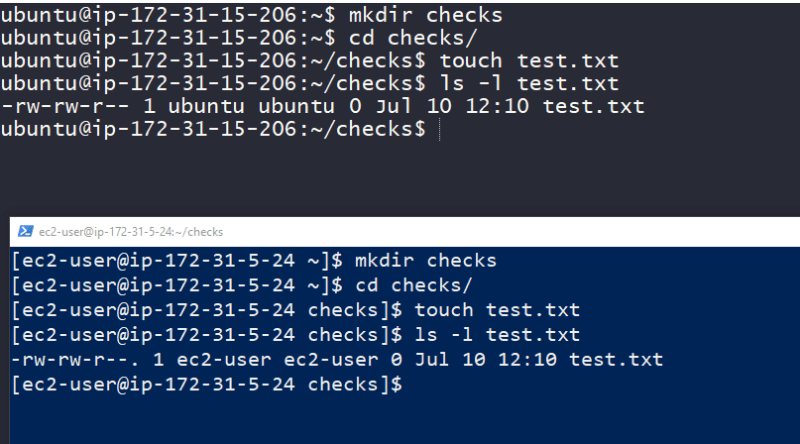
/etc/group: groups are defined

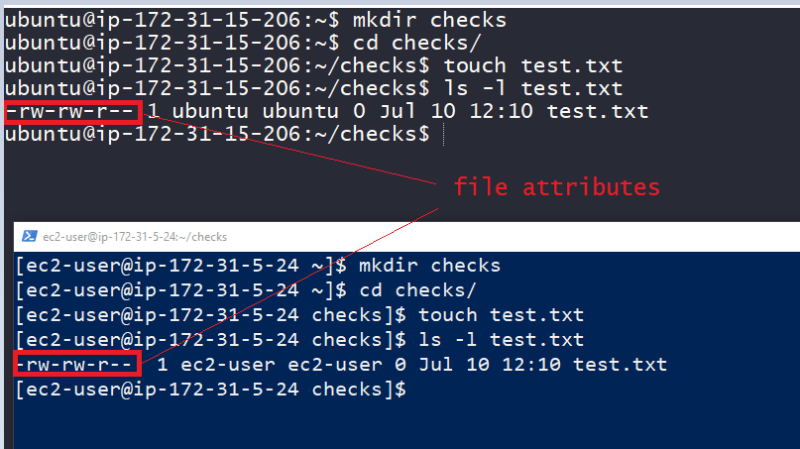
/etc/shadow is the file which also gets modified when users and groups are created as it holds information about user’s password.

In linux systems, there are superuser accounts (uid 0)

#### Reading, Writing and Executing

Access rights to files and directories are defined in terms of read access, write access and execution access

Let’s create a simple test.txt as shown below and execute the commands 

The first character highlighted in the below image are file attributes 

The first of these characters will help us in describing file type

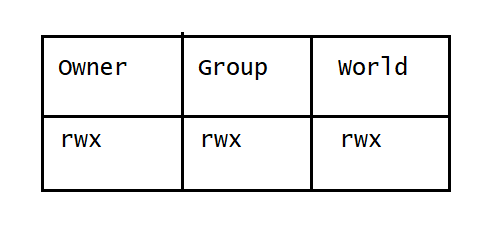
**–** => A regular file

**d** => directory

**l** => A symbolic link

**c** => A character special file eg terminal

**b** => A block special file eg: hard drive or DVD drive

The remaining nine characters of the file attributes are called as file mode, They represent read, write and execute permissions for the file’s owner, the file’s group owner and everybody else 

Permission Attributes

r => Allows file to be opened and read, Allows a directory contents to be listed

w => Allows file to be written to or truncated. The ability to delete or rename file is determined by directory attributes. Allows files in the directory to be created deleted and renamed.

x => Allows a file to be treated as a program and executed. Program files written in scripting must also be set as readable to executed. Allows a directory to be entered

File attributes:

-rwx—— => A regular file that is readable, writable and executable only by the files owner, no one else has access

-rw——- => A regular file that is readable and writable by file’s owner, no one else has access

-rw-r–r– => A regular file that is readable and writable by file’s owner, Members of file owners group may read the file. The file is world-readable

-rwxr-xr-x => A regular file that is readable, writable and executable by files owner. The file may be read or executed file’s group member and world

drwxrwx— => A directory. The owner and members of owners group may enter the directory and create, rename and remove files from the directory

chmod: Change file mode

numeric notation

read ‘r’ = 4

write ‘w’ = 2

execute ‘x’ = 1

examples:

666 => rw-rw-rw-

755 => rwxr-xr-x

400 => r--------

To change file mode we use chmod

Chmod Symbolic notation

u => user

g => group

o => others

a => all

Symbolic notation examples

u+x => Add execute permission for the owner

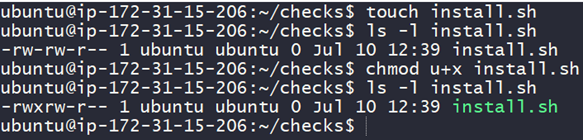
u-x => Remove execute permission for the owner

+x => Add execute permission for owner,group and world => a+x

o-rw => Remove read and write permissions for others (world)

go=rw ? (Exercise)

u+x,go=rx ? (exercise)



### Changing identities

Often, we want to gain superuser privileges to carry out some administrative tasks. what are possible ways

Logout and log back in as alternative user

Use the **su** command

Use the **sudo** command

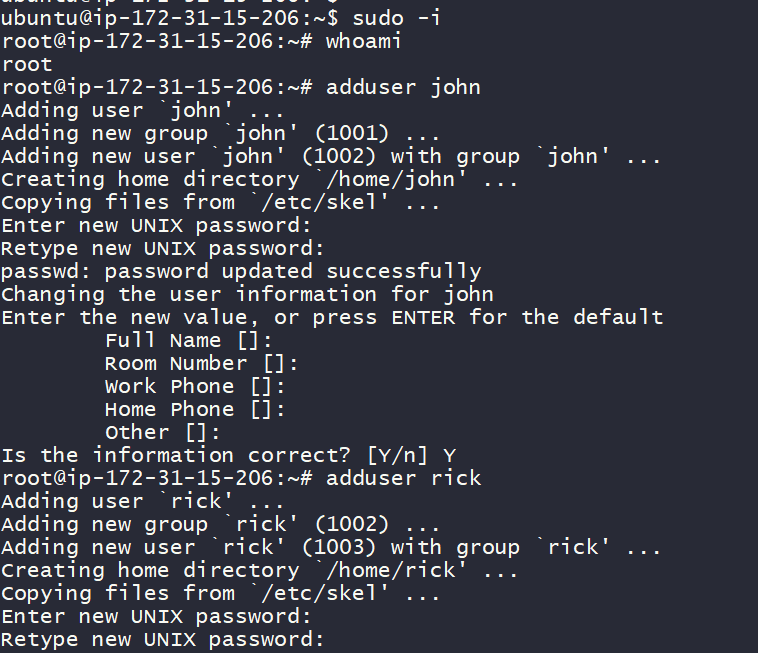
For experimentation lets create 3 user john, rick, siva

sudo -i # become root user

adduser john

adduser rick

adduser siva



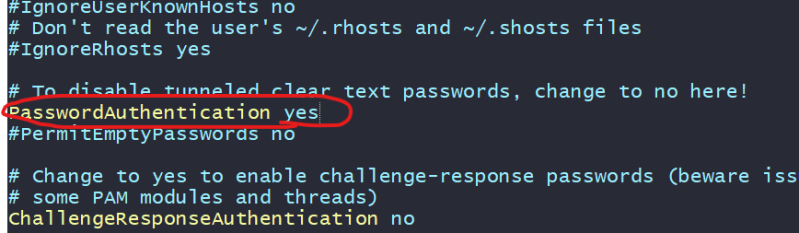
Since in AWS password authentications are disabled we will be enabling

vi /etc/ssh/sshd\_config

change PasswordAuthentication to yes

service sshd restart

exit



### su: Run Shell with Substitute User and Group Id

**su** command will look like

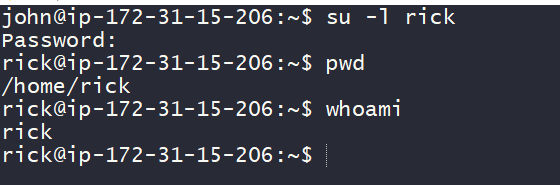
su [-[l]] [user]

Execute help for **su**

man su

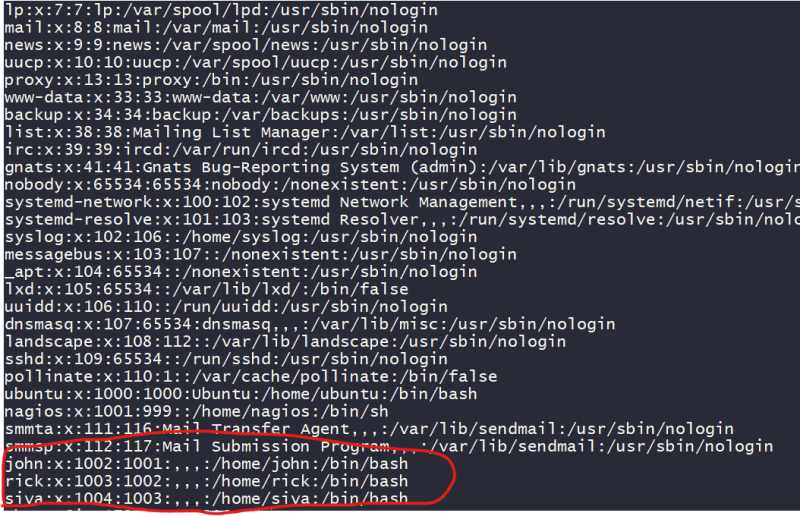
su --help

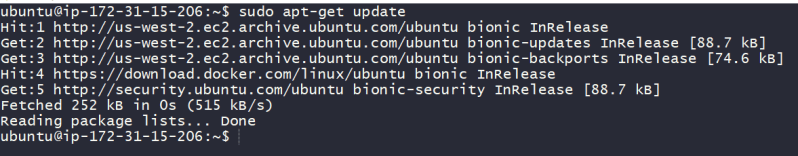
Let’s try to become john 

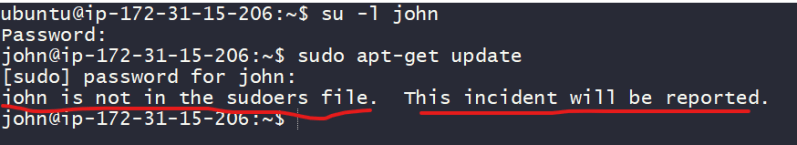
Let’s try to become rick 

If you want to become root user and if root user has a password

su -

Lets look at file ‘/etc/passwd’ 

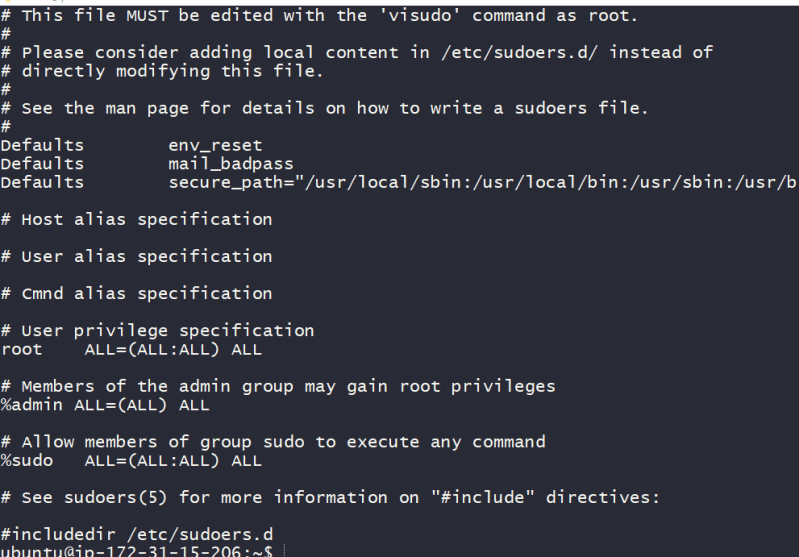
With default user (ubuntu) lets execute sudo apt-get update and it works 

Lets try to execute same command by switching user to john 

How to give john the permission to execute sudo commands?

become ubuntu (default) user

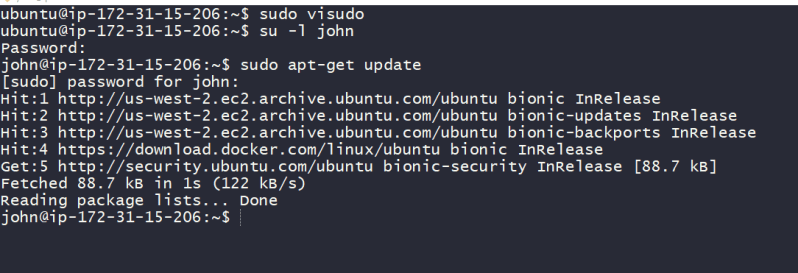
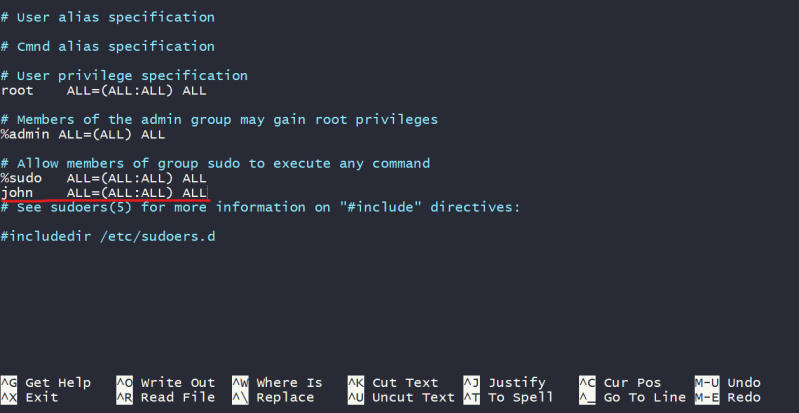
sudo cat /etc/sudoers



John is not part of sudoers file so lets add john to sudoers

# login as default user (ubuntu)

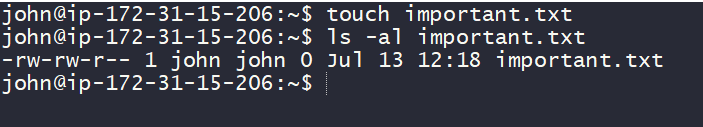
sudo visudo



If we want to execute any command with sudo on the ubuntu machine user should be part of sudoers file, then user can execute commands with sudo.

Sudo is run a command as superuser

### chown: Change File Owner and group

Lets create a simple file as john 

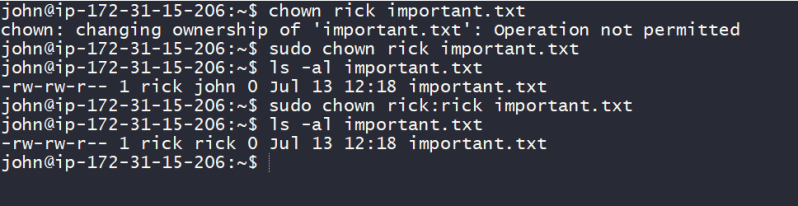
I want to change the owner of this file to rick. This is where a linux command **chown** comes into play

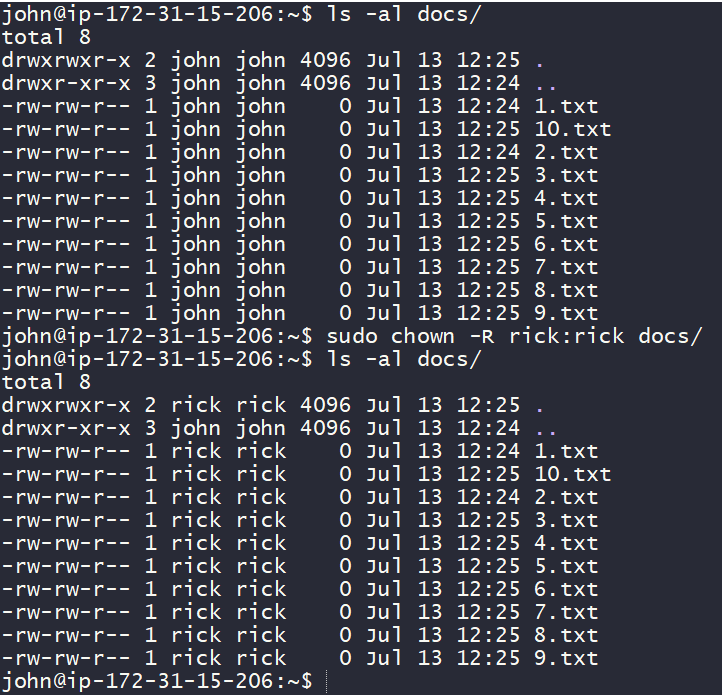
man chown

chown --help

Syntax is

chown [owner][:[group]] <file>

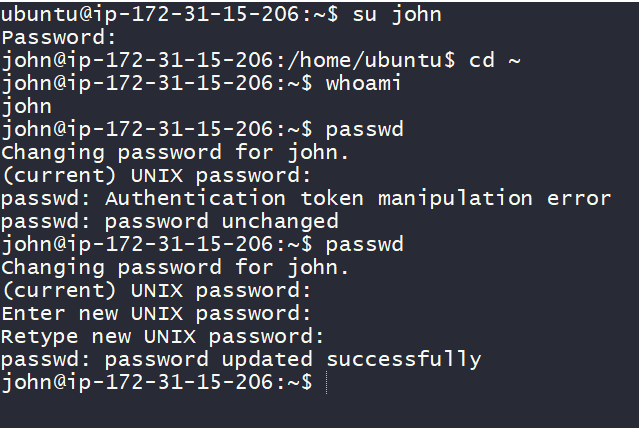
lets change ownership of a file 

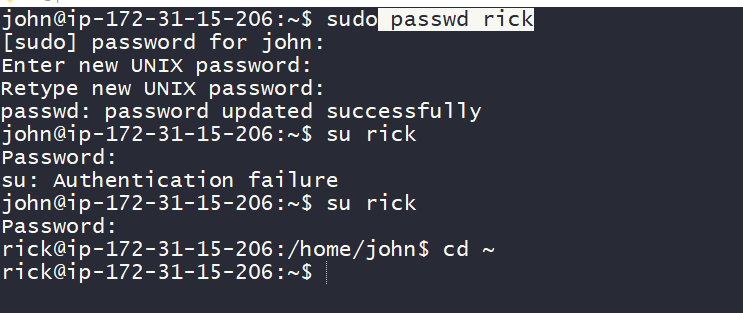
Now lets see how to change ownership of folder 

### Change your password

Command for changing the password

passwd [user]

Lets login as user john and change his passwd 

Login as sudo user and change password of other user 

root user can change password of any user

### Change mode

Syntax

chmod [reference][operator][mode] file ..

Reference

u => owner

g => group

o => others

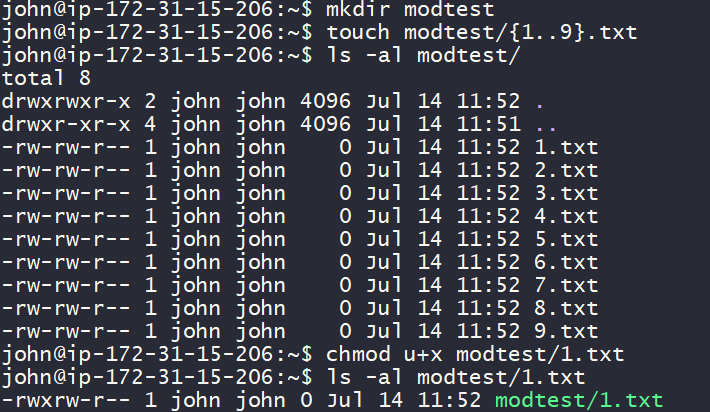
a => all

Operators

=> Add the modes

=> Remove the specific modes

=> The modes specified are made exact

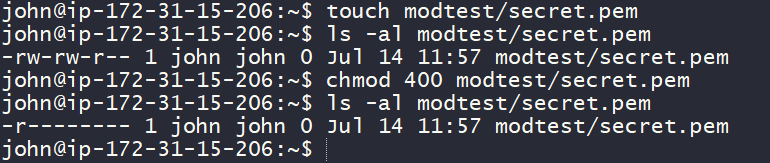
Examples 

mkdir -p modtest/{a,b,c}

ls -al modtest/

chmod g=rw modtest/a

ls -al modtest/



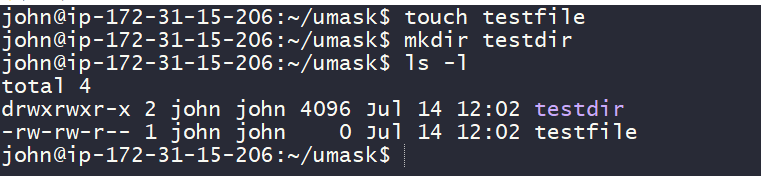
### umask

UMask is known as User Mask or it is also called as User file creation Mask.

This is a base permission or a default permission when a new file or folder is created in linux

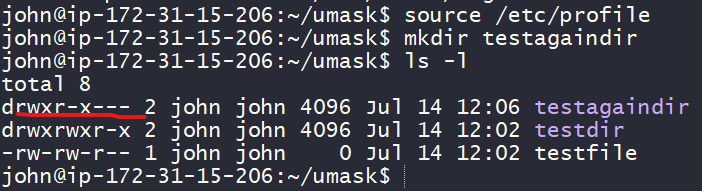
touch testfile

mkdir testdir



The default permissions for file are 644 and for directory are 775

Let’s add following entry into a file /etc/profile and add the entry umask 0027



777

027

750

rwxr-x---

Now observe directory permissions for the newly created directory

### Processes

A process is basically a program that is executing.

Executing the program happens in a sequential

To run anything on OS, a process has to be created

For any process to run

A program is loaded into memory and it becomes process. Process has 4 sections

Stack:

Stack contains temporary data such as what method/function paramters, return addresses & local

Heap: This is dynamically allocated memory to process

Data: Global & static variables

Text: Program counter, Process registers

Process LifeCycle

Start

Ready

Running

Waiting

Terminated or Exited

Let’s understand commands which we would examine w.r.t process

ps: Report the snapshot of current process

top: Display task

jobs: List active jobs

bg: Place the job in the background

fg: Place the job in the foreground

kill: Send a signal to a process

Killall: Kill a process by name

shutdown: Shutdown or reboot a system

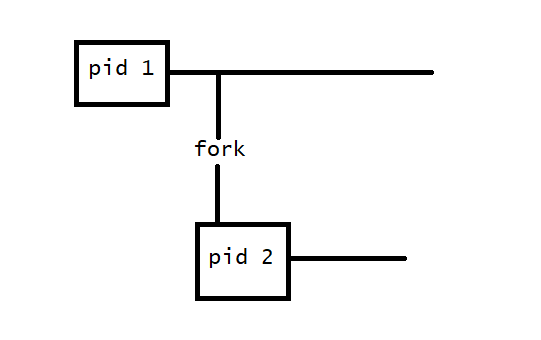
Daemon Programs:

When a linux starts, kernel has to run some activites, so it initiates those activities as process and launches a program called as **init**

**init** runs a series of shell scripts called as init scripts, which start all the system services.

Many of these services are implemented as daemon programs, Programs that jus run in the background and generally they don’t have UI.

### fork() and exec()

fork() creates a new process by duplicating the current process. The new process is referred as child and existing process from which duplicate is created is referred as parent 

Child will its own unique PID (Process ID),

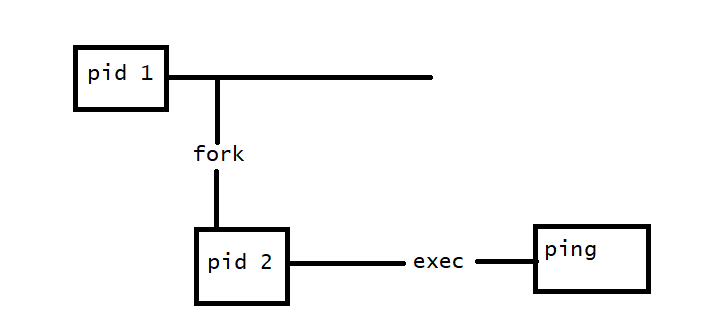
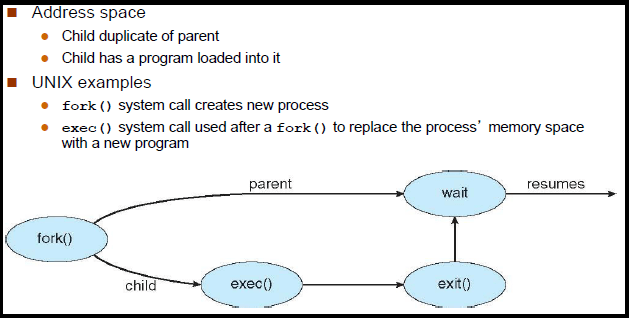
The childs parent process ID will be same as parents process ID

Child doesn’t inherit parents memory locks and semaphore adjusments

Child doesn’t intherit outstanding I/O operations from parent.

On success, the PID of the child process is returned to parent and 0 is returned to child. On failure, -1 is returned to parent

exec() replaces the current process with new one.



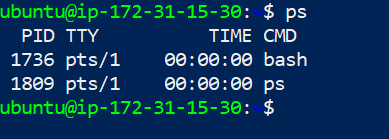
### Viewing Processes

The most commonly used command to view processes is ps

ps --help

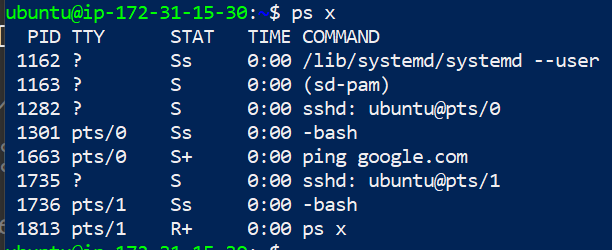
man ps

Let’s execute ps



The result in this example lists two process process 1736 and 1809 which are bash and ps.

To get bigger picture let’s try to execute ps x



Adding the x opts tells ps to show all of the our processes regardless of what terminal they are controlled by. The presence of ? in the TTY column indicates no controlling terminal

State Meaning

R: Running

S: sleeping

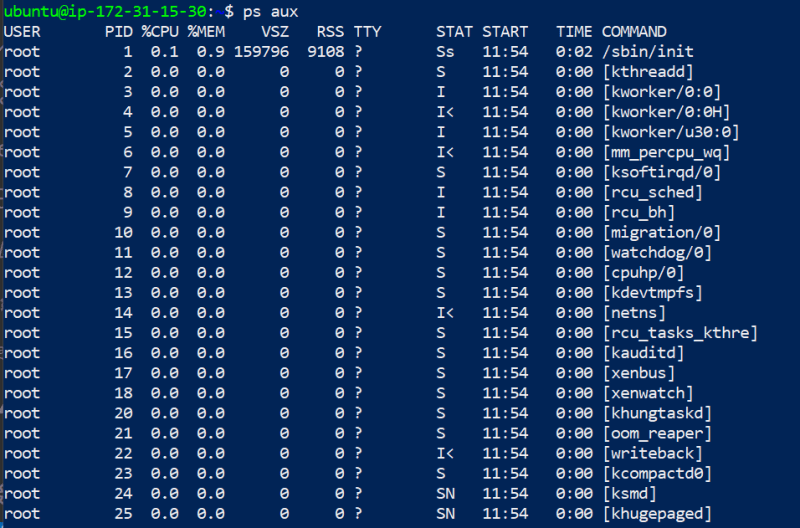
D: Uninterruptible sleep

T: stopped

<: A high priority process

N: A low priority process

Let’s get into more details of the process by executing ps aux



Header Meaning

USER: USER ID. This is owner of process

%CPU: Cpu usage in percent

%MEM: Memory usage in percent

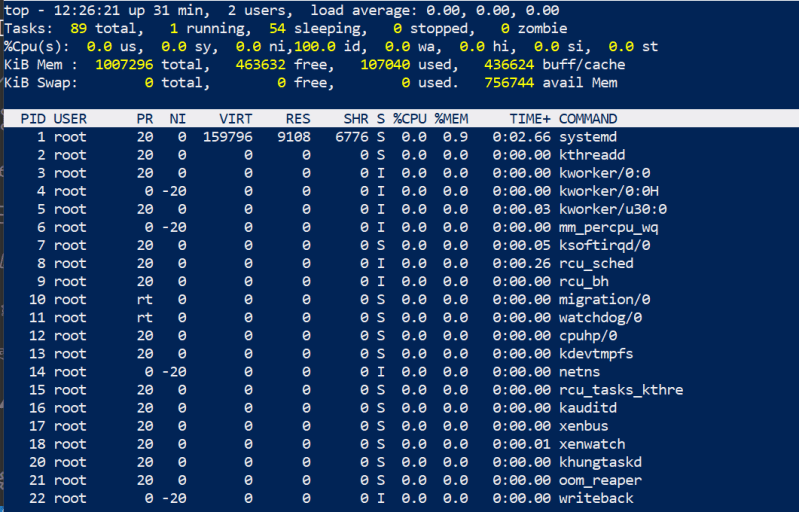
VSZ: Virtual Memory Size

RSS: Resident Set size. This is amount physical memory(RAM) the process is using in kilobytes

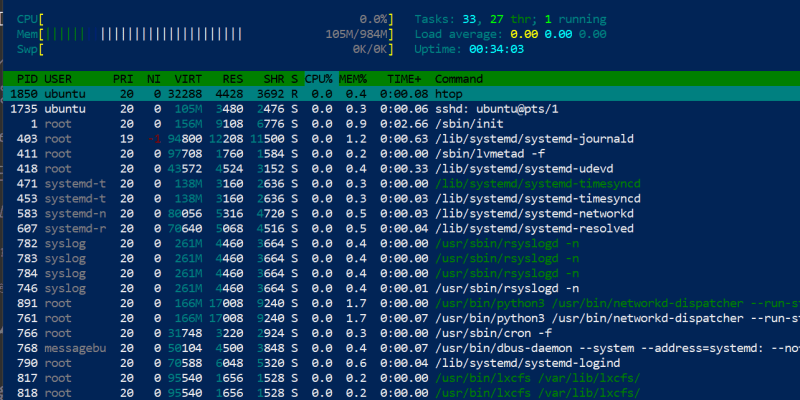
START: Time when the process started.

Ps command can tell you lot about what machine is doing at the moment when you typed the ps command

To get dynamic view of machines activity use top clear



Also try htop



PS command shows static info, whereas top and htop commands show us dynamic info

We will be using a ubuntu vm and redhat vm. Ubuntu is connected for git bash (terminal in black) and redhat from powershell



### Linux Job

A linux job is a process which shell manages. Each job is assigned with a sequential job id. But each job is process & it will also have PID

Jobs will have following statuses

Foreground:

Background:

Stopped

Commands:

jobs: List all the jobs

bg %n: Places the current or specified job in the background

fg %n: Brings the current or specified job into the foreground

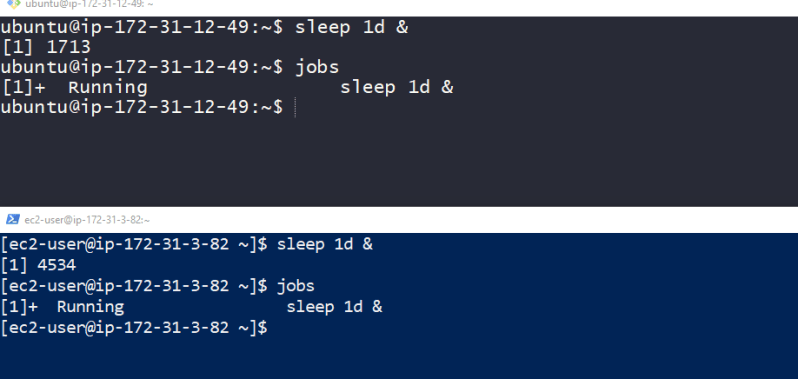
Control-z: Stops the foreground job & places it in the background as stopped job

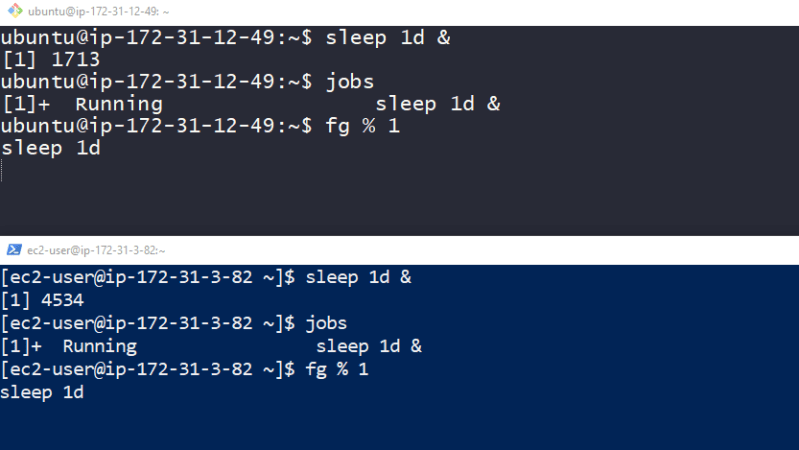
Run a Job in the Background

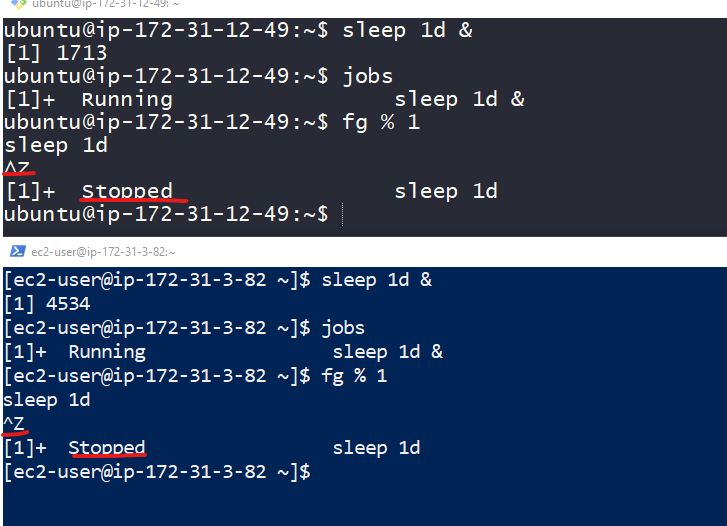
You need to enter the command followed by an ampersand &

sleep 1d &

jobs



To bring jobs to foreground 

To stop the jobs 

### Signals

The kill command is used to kill process.

The kill command doesn’t kill process rather it sends signals. Signals are one of several ways that os communicates to program

Ctrl-C: Send a signal called as INT(interrupt)

Ctrl-Z: Sends a signal called as TSTP(Terminal Stop)

Kill command syntax

kill -signal PID ...

IF no signal is specified then the TERM (terminate) signal is sent by default

### Let’s look at Common Signals

HUP: Hang up. The signal number is 1

INT: Interrupt. This performs same functionas CTRL-C sent from terminal. It usally terminates the program . The signal number is 2

KILL: Kill. This signal is special because this signal will not reach process, kernel will terminate program immedietly, so no clean up oppurtunity given to your program. Signal number is 9

TERM: Terminate. This is default signal sent by kill. Signal number is 15

CONT: Continue. This will restore the process after STOP or TSTP signal. This signal is sent by bg and fg commands. Signal number is 18

STOP: Stop. This signal causes a process to pause without terminating. Signal number is 19

TSTP: Terminal Stop. Signal number is 20

### Kill with Signals

### PreviewPreview

### Bash nohup

The meaning of nohup is ‘no hangup’.

Normally when we logout from system then all the running programs or process are hangup or terminated. If you want to run any program after logut or exit from Linux OS then you have use nohup command.

### Shutting down the system

Four commands

halt

poweroff

reboot

shutdown

### Other Useful Process Commands

pstree

vmstat

xload

tload

### Environment

Shell maintains lot of information during our shell session called the environment

Programs/applications can use the data stored in this environment to determine info about system’s configuration.

In this series

**printenv**

**set**

**export**

**alias**

#### What gets stored in Environment?

Two types of data gets stored in the environment

environment variables

shell variables

In addition to above mentioned variables shell stores

aliases

shell functions

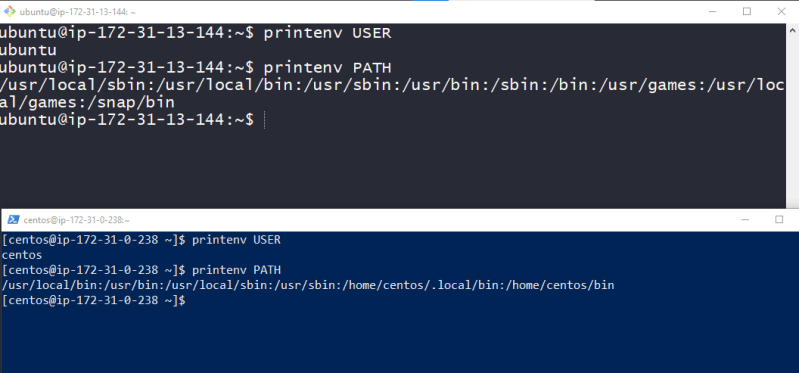
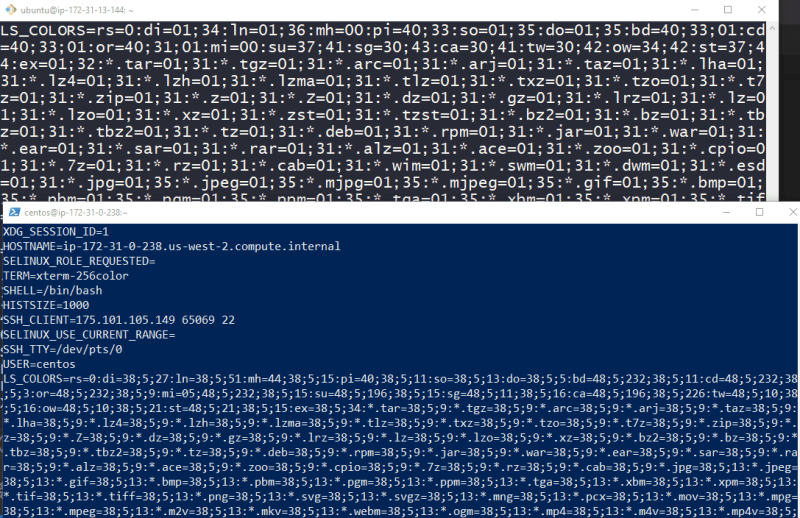
### Examine the Environment

Execute the following command

printenv|less

printenv USER

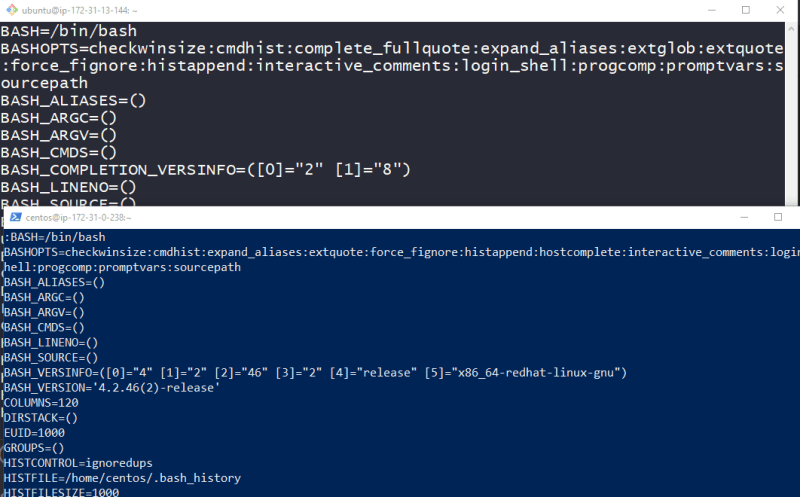
printenv PATH



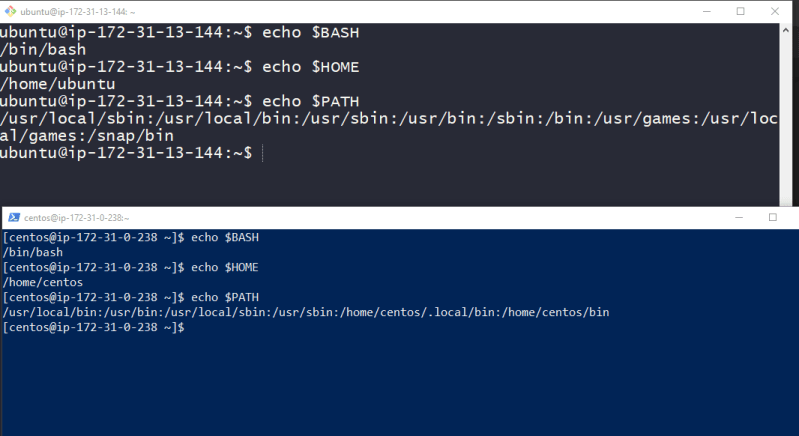
Now lets execute one more comand

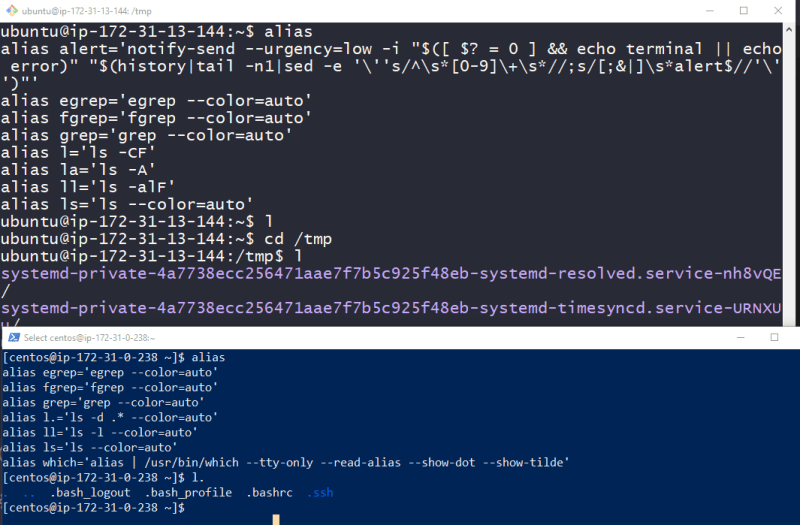
set | less

echo $HOME



printenv command shows only environmental variables where set command displays both environment and shell variables

we can print variables by adding $ to the varaible name 

Neither printenv nor set commands show aliases for that we need use command alias 

#### Some Interesting Variables

SHELL: The name of your shell program

HOME: The path of your home directory

LANG: Defines the character set

OLDPWD: Previous working directory

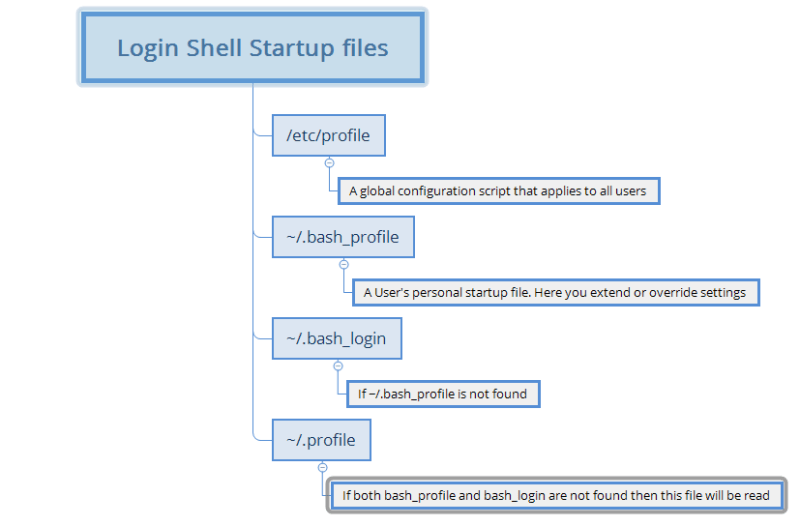
PATH: A colon-separated list of directories that are searched when you enter the name of a executable program

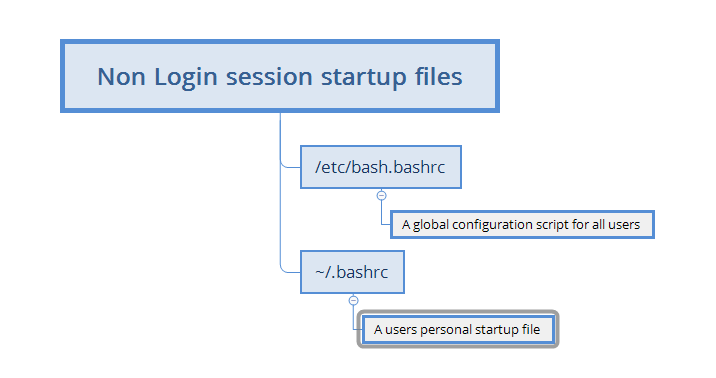
USER: Your username

### How is this Environment established?

When we login into linux system, bash program starts and reads series of configuration scripts called as start-up files

The exact sequence of files depends on type of shell you have started. There are two types

Login shell session: This is the one in which we are prompted for username & password (ssh) 

A non-login shell: This generally occurs when we launch a terminal in GUI. 

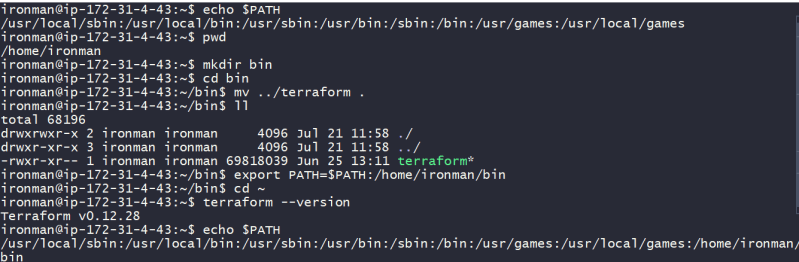
### Setting Environmental Variable Path

Lets try to install software terraform which is used by user ironman

Lets try to install software packer which is used by user thor

Lets try to install software java 8 which should be used by both the user

After downloading terraform, unable to run command Preview

So lets add the folder of terraform to the path using export command 

Now i will logout of ironman, login & recheck & this fails

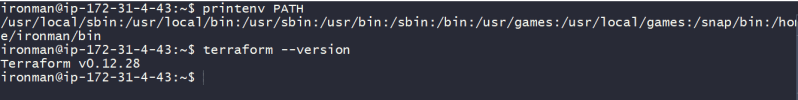
So let create a file

echo 'export PATH=$PATH:/home/ironman/bin' > ~/.bash\_profile

Now logout & relogin & execute

printenv PATH

terraform --version



Now lets do the samething for packer for thor

echo 'export PATH=$PATH:/home/thor/bin' > ~/.bash\_profile

exit

# relogin

packer --version

Packer can be downloaded from [here](https://www.packer.io/downloads) & Terraform from [here](https://www.terraform.io/downloads.html)

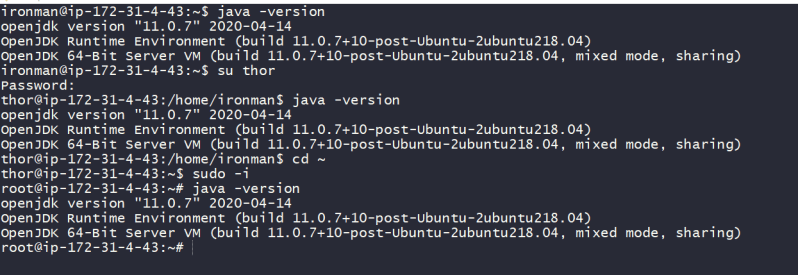
Now lets install Java 8

sudo apt-get install openjdk-8-jdk -y

Java can be used by all the users as java executable is added to folder which is in path for all the linux users.

Now lets assume on the same machine ironman wants java 11 & all other users want java 8

sudo apt-get install openjdk-11-jdk -y



All the users got JAVA 11, now how to solve this problem. If use ENVIRONMENTAL variable called as JAVA\_HOME, the problem can be solved

Add the following lines to /etc/profile

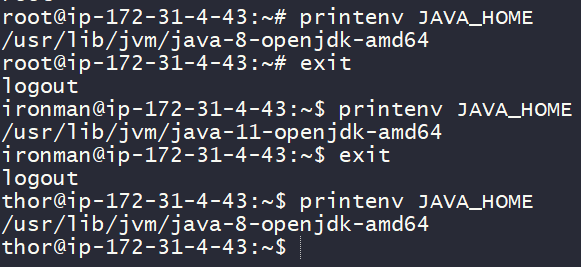
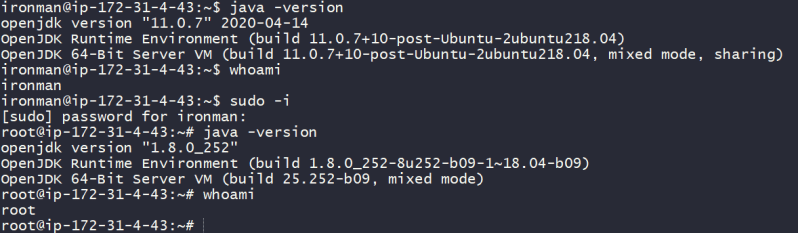
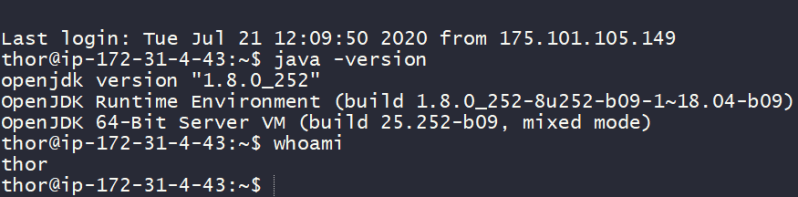
export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export PATH=$JAVA\_HOME/bin:$PATH

Add the following lines to /home/ironman/.bash\_profile

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export PATH=$JAVA\_HOME/bin:$PATH



### Users and Groups in Linux

Linux is a multi user operating system.

To add a new user account, we have two commands

adduser

useradd

Generally whenever a user is created, User gets

username

uid

home directory

A group is created for the user with the same name as username

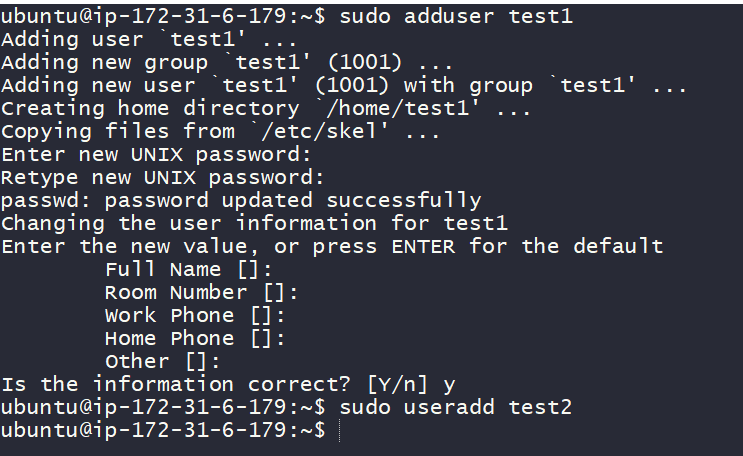
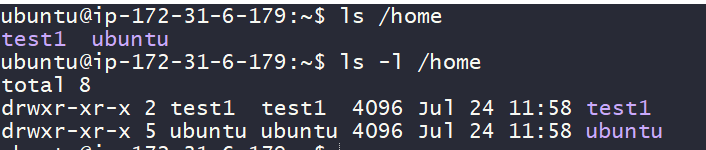
Difference between useradd and adduser

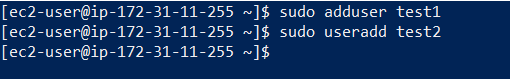
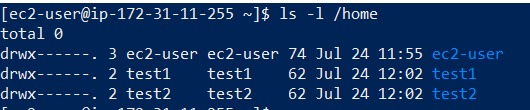
adduser test1

useradd test2

cat /etc/passwd

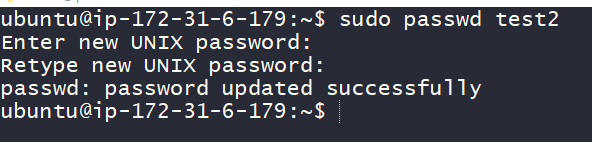
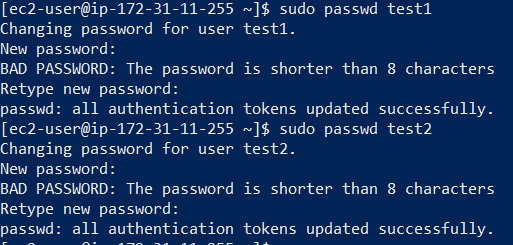
ls /home

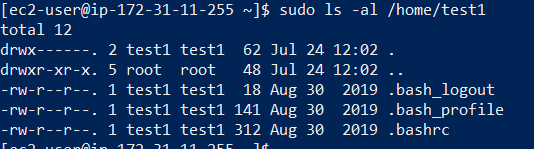
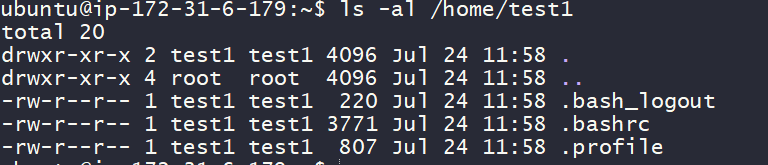
Preview

Lets look at redhat Preview

we can set the password to the user by executing

passwd



In the home directory, few hidden files are created 

If you look into /etc/passwd the entry pattern in this file looks as

[username]:[x]:[UID]:[GID]:[COMMENT]:[HOMEDIRECTORY]:[Defaultshell]

# ubuntu entries

test1:x:1001:1001:,,,:/home/test1:/bin/bash

test2:x:1002:1002::/home/test2:/bin/sh

If you look into /etc/group file, the format for each line

[Group name]:[Group password]:[Gid]:[Group members]

# sample ubuntu entries

test1:x:1001:

test2:x:1002:

For group management

groupadd

groupdel

usermod

Exercise:

Create a user called as ironman

Create a user called as thor

Create a user called as hulk

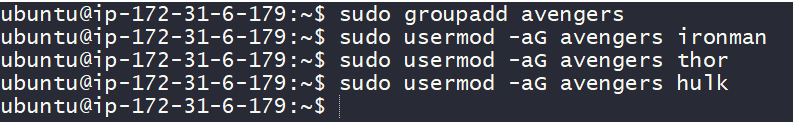
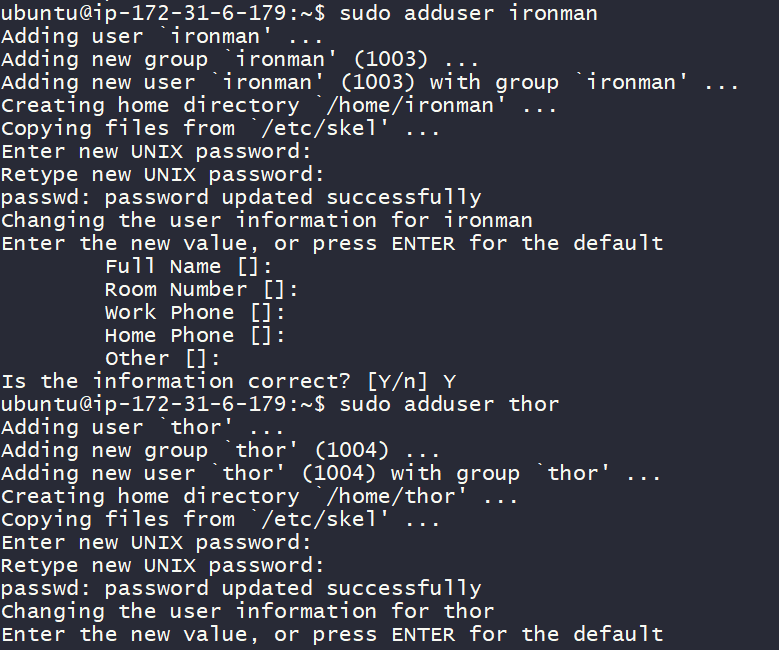
create a user called as superman

Create a user called as batman

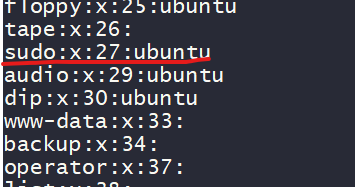
Create a user called as wonderwoman

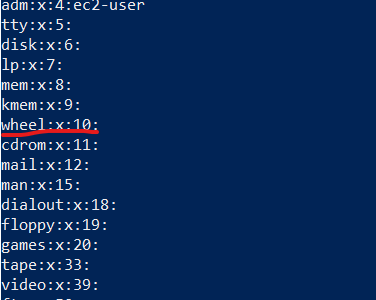
create a group called as Avengers and add ironman, thor and hulk to it

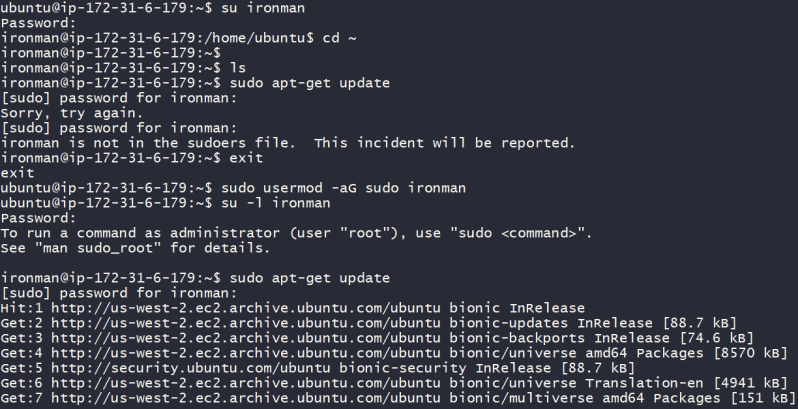
Create a group called as JusticeLeague and add superman, batman and wonderwoman.

Some screenshots 

How can i give sudo permissions to my users

In ubunutu we have a group called sudo, if you add users to this sudo group, then can execute sudo commands. 

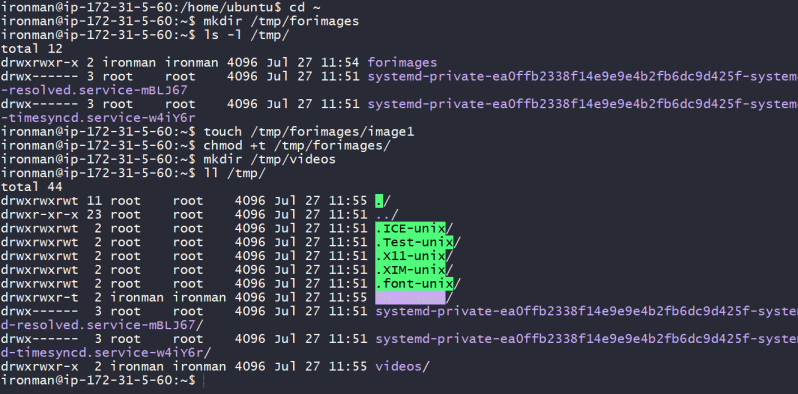
In RHEL family, we have a group called as wheel, adding users to this wheel group can enable users to execute sudo commands 

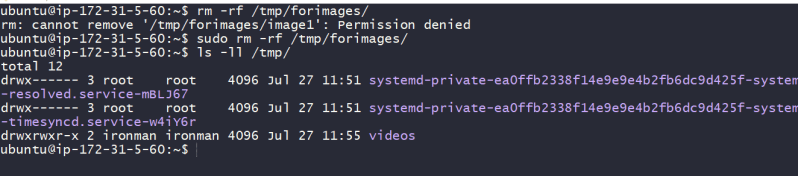
Adding a user ironman to sudo group 

### Sticky bit

Sticky is a special permission that cna be used to prevent users from deleting other users files and directories

Root user can ofcourse delete this, but not others users

Adding stick bit to use +t in chmod 

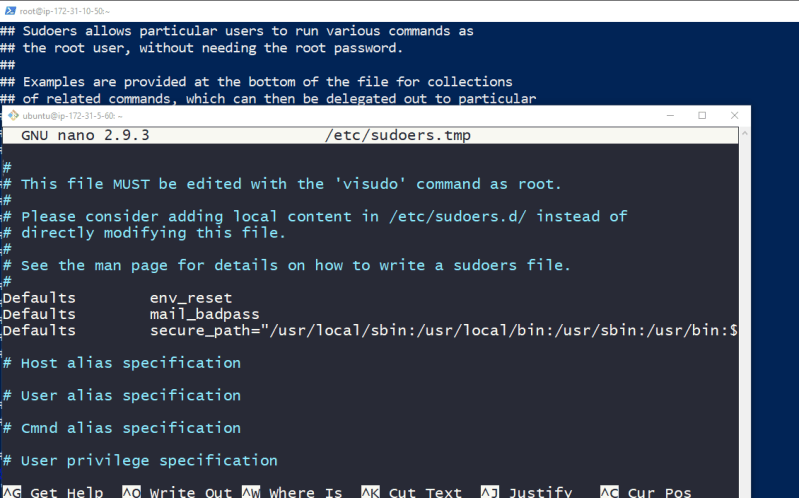
If we add chmod +t to files/directories others users cannot delete it, but if the other user is sudoer or super user they can delete 

#### visudo

visudo is a command that helps in editing sudoers file.

since its not a good idea to change /etc/sudoers file directly, visudo command can help in editing sudoers file

visudo edits the sudoers file, which defines the users and groups with administrative rights

Now execute the command visudo from root user 

any line which starts with # is comment

In this file **root ALL=(ALL:ALL) ALL** states that the user root, may run as any user or group , any command

Syntax of directive

user hostname=(runas-user:runas-group) command

If the user in the directive begins with %, it is group

%admin ALL=(ALL) ALL

%sudo ALL=(ALL:ALL) ALL

%wheel ALL=(ALL) ALL

For sudo users when execute sudo password/key is expected, to ensure linux doesn’t ask passwords when we use sudo command

%sudo ALL=(ALL:ALL) NOPASSWD:ALL

%wheel ALL=(ALL) NOPASSWD:ALL

### Managing Softwares in Linux

A good chunk of time as sys admin/devops will be spent in managing the various softwares/applications.

When we install any software, we would be running them. In this running/executing there are two kinds of softwares

Software which executes only when the user executes the command

Software which run all the time in background. In linux any program that runs in the background is called as daemon.

#### Approaches of installing software in linux

From source code:

Copy the source code in linux

compile the source code (make tools)

copy the executables into the folders of your choice

Add folders to PATH variables (if required)

Example: Look into nagios installation

From executables:

Software executables will be shared to you as tar/zip files

untar the files, copy them to folders of your choice and execute applications

Package Managers:

will be discussed in next class

### Software Installations in Linux Using Package Managers

#### Red Hat Package Manager (RPM)

RPM is a software management system that is used for installation & removal of software packages.

A package typically consists of archive of files and other metadata, includes configuration files, binaries.

RPM is pre-install on all redhat flavors like centos, rhel, fedora, Amazon linux etc…

Following are the primary functions of RPM program

Query, verify, update, install and uninstall software

Maintains a database that stores information about packages

Package other software into RPM form

Querying for all package

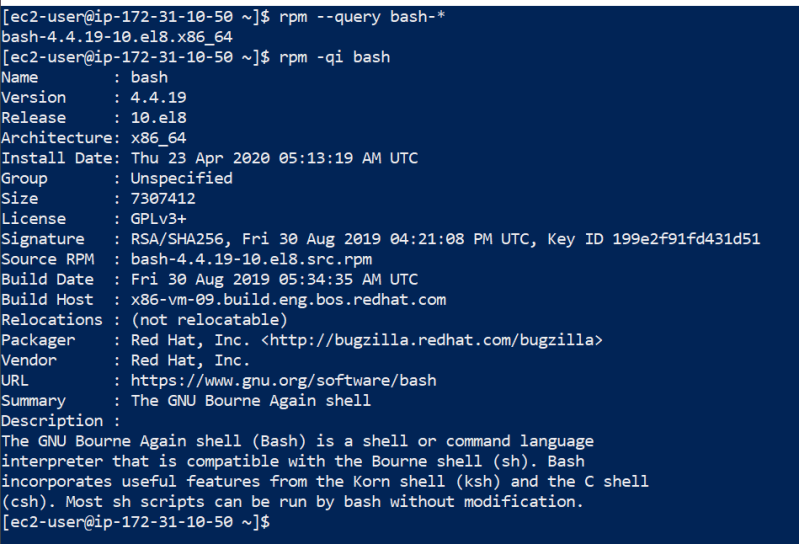
Execute the command below to list all the packages install on your system

rpm --query --all

Querying Details for a specific package

Lets find information about a package called as bash

rpm --query bash-\*



To list all files that come with your package

rpm -ql bash

To list configuration files

rpm -qc bash

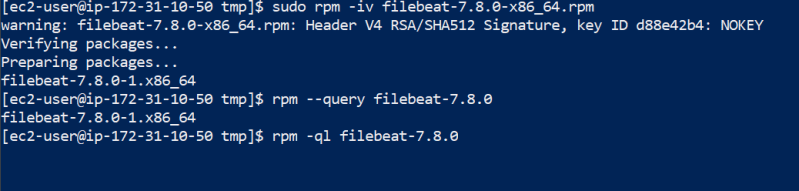
Determining what other packages on the system depend on the bash package

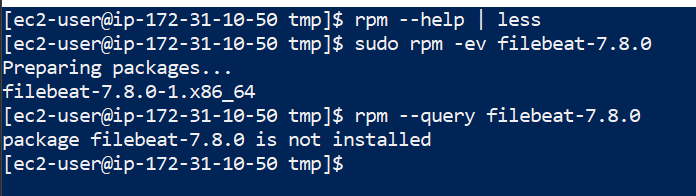
rpm -q --whatrequires bash

Installing software using rpm

Lets use rpm to install file beat available [over here](https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.8.0-x86_64.rpm)

curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.8.0-x86\_64.rpm

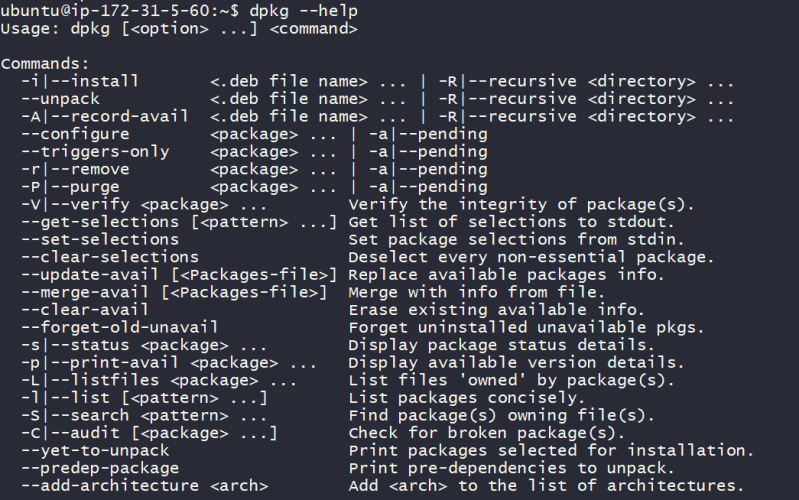


Uninstall software using rpm 

### The Debian Package Management System (DPMS)

DPMS is the framework for managing software on Debian or Debian-like systems

Debian packages end with .deb

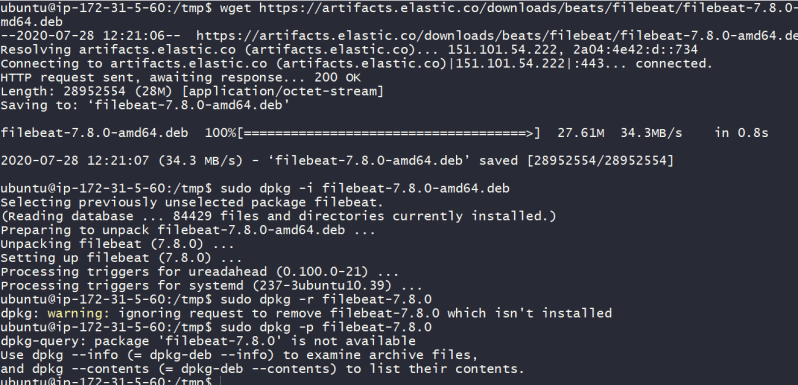
At the core of DPMS is the dpkg application which works with the system providing several command line options 

Lets install and uninstall filebeat from debian package available [over here](https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.8.0-amd64.deb)

cd /tmp

wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.8.0-amd64.deb

sudo dpkg -i <debpackage path>



#### Automatic updates and package installers

We have following popular tools

YUM

DNF

APT

SNAP

### Linux Repository Based Software Management

A linux repository is a storage location from which our system can install applications as well as Operating system updates.

Each repository is collection of software hosted on remote server so that it can be used my many systems for installing/updgrading softwares (packages)

#### YUM

yum is a popular packaging/updating tool for managing software on Linux systems. It is basically a wrapper program for RPM with enhancements.

Yum is pre installed on rhel family like centos, redhat, amazon linux

Recommendataion: Use cheatsheet [Refer Here](https://access.redhat.com/sites/default/files/attachments/rh_yum_cheatsheet_1214_jcs_print-1.pdf)

Sample commands

yum install <package-name>: install the package

yum erase <package-name>: remove the specified package

yum search <package-pattern>: searches the lust of package names

yum deplist <package-name>: List all the libraries and moduels that the package depends on

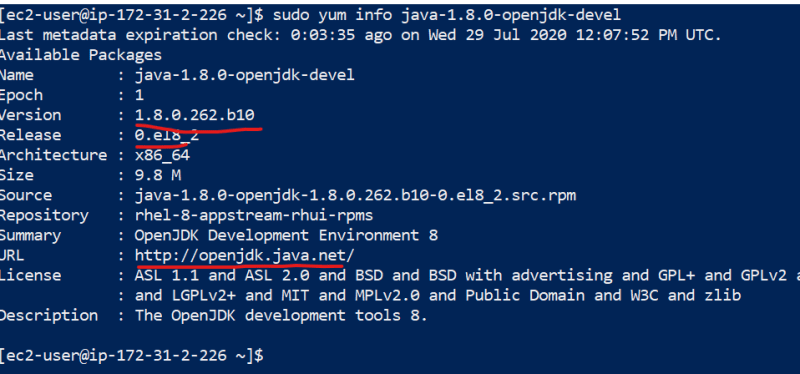
yum check-update: Refreshes the local cache of yum database

yum reinstall <package-name>

yum update <package-name>

yum upgrade: Upgrades all packages install in your system to latest release

Yum configuration is located at **/etc/yum.conf** provides system-wide configuration options for yum.

In yum we can navigate to /etc/yum.repos.d 

If you want to add a repo create a file with .repo extensionin this folder and add the information

[REPO-NAME]

name=REPOSITORY-NAME

mirrorlist=<http-url>

#baseurl

gpgcheck=1

gpgkey

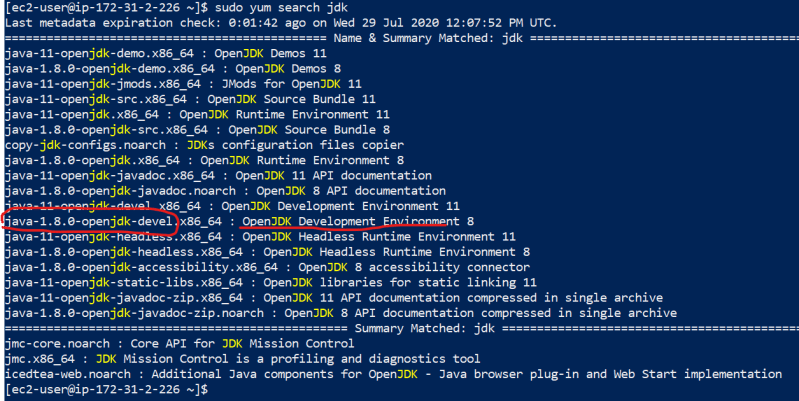
Case study:

Install OpenJDK 8 using yum (Linux has repository already added)

sudo yum search jdk

sudo yum info java-1.8.0-openjdk-devel

sudo yum install java-1.8.0-openjdk-devel -y



Install Logstash (Linux doesnt have a repository where the software is distributed)

Lets add information about a new repository to our redhat system. create a new file called as elastic.repo in /etc/yum.repos.d with following info

[logstash-7.x]

name=Elastic repository for 7.x packages

baseurl=https://artifacts.elastic.co/packages/7.x/yum

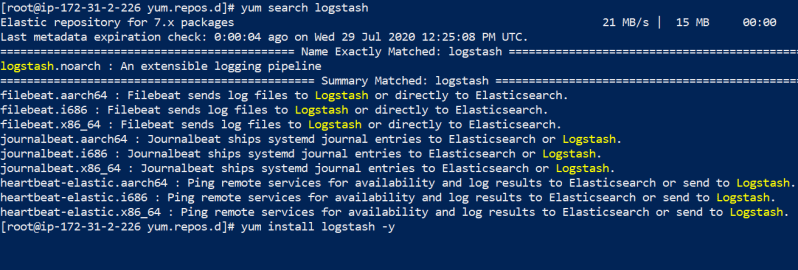
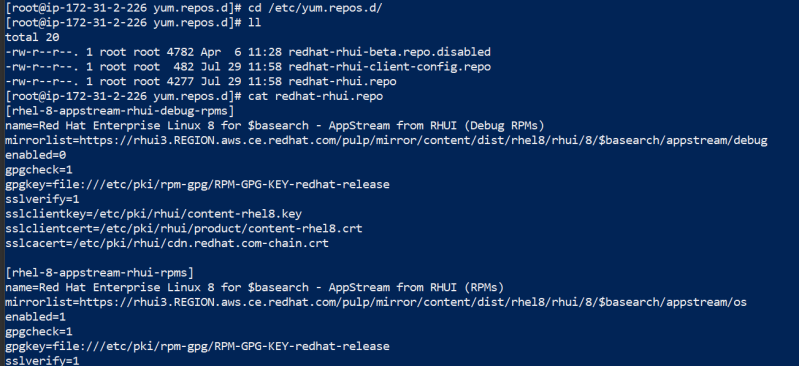
gpgcheck=1

gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch

enabled=1

autorefresh=1

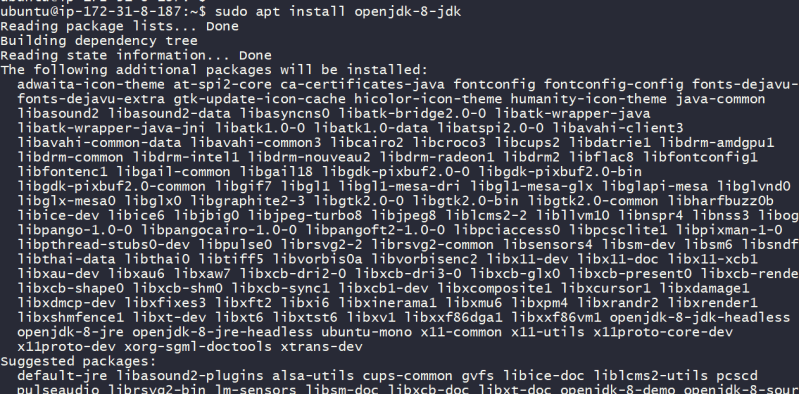
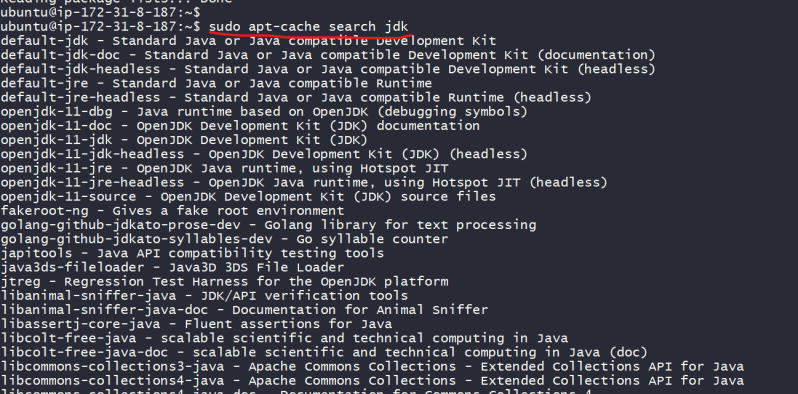
type=rpm-md



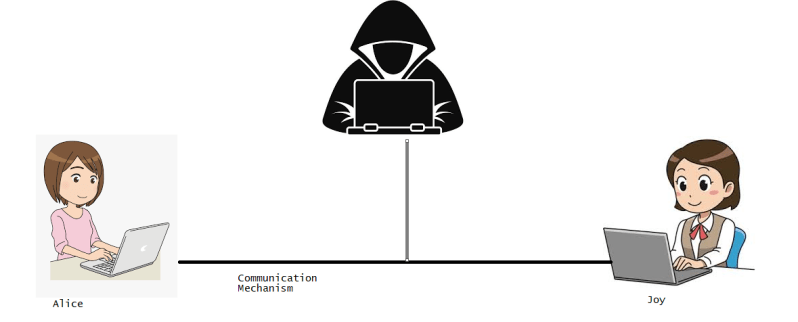
#### APT

APT is a tool for managing packages on ubuntu based systems

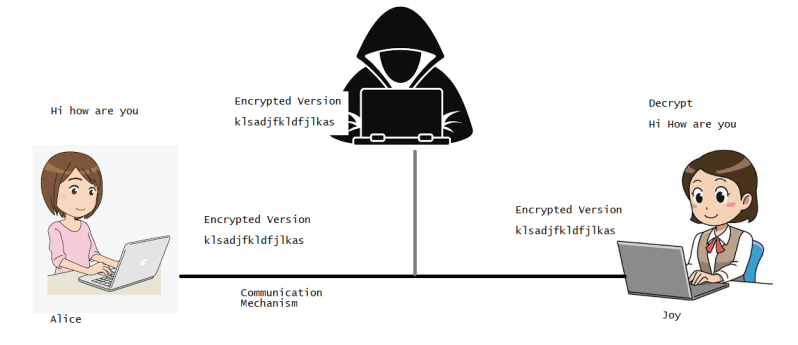
[Refer Here](https://blog.packagecloud.io/eng/2015/03/30/apt-cheat-sheet/#apt-get-high-level-package-handling-utility) for apt based cheat sheet

Lets install Java8 in ubuntu (repository already added) 

### Basics skills for SSH

Lets understand a scenario where alice wants to communicate to Joy. The communication channel which they are using in not secure (Any one can hook & read their messages) 

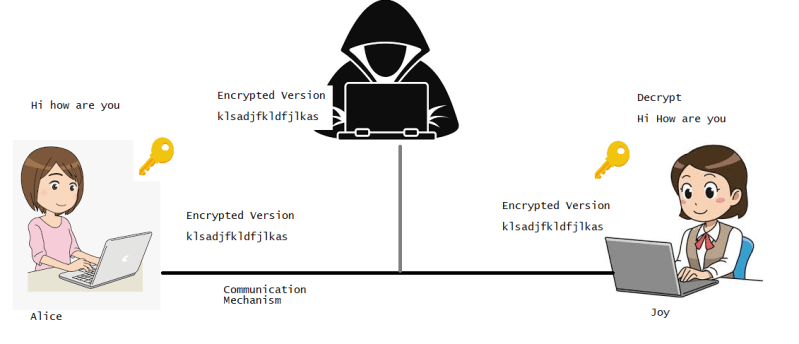
What can be done by Alice & Joy so that they use the same communication channel, still their message are secure.

They encrypt and decrypt the messages 

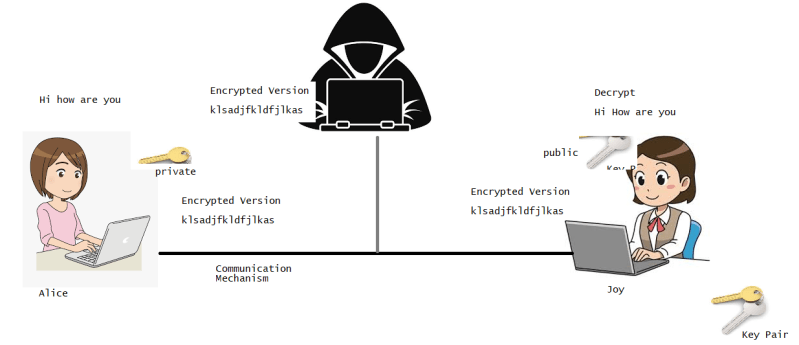
To do this Encryption and decryption we have two major ways

Symmetric encryption

Asymmetric encryption

In Symmetric both Alice and Joy will have the same key to encrypt and do decrypt 

In Asymmetric alice or joy will create a key pair. Alice will have a key and joy will have other key (both are different). One key is called as public & Other is private.

If the message is encrypted with public key it can be decrypted with private and the other way around 

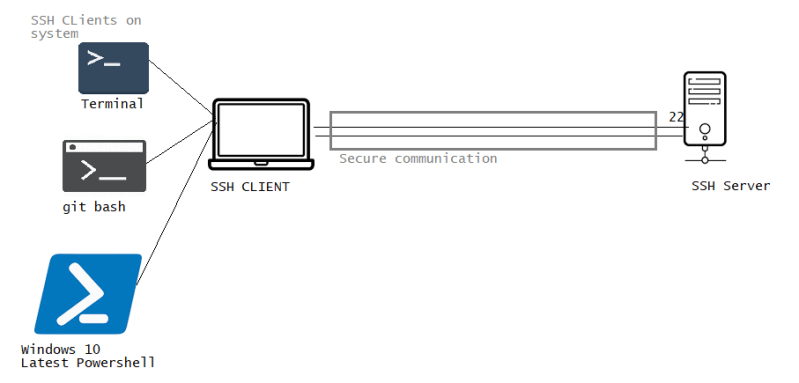
### Secure Shell (SSH)

Telnet was used to communicated with a remote server, Telnet is not a secure communication protocol & it transfers the data over network/internet in a plain text, so to overcome this issue SSH came into existence

**SSH** protocol provides the secure way of accessing remote computers.

IN SSH we will have two parties

SSH Client

SSH Server 

#### HOW SSH works

SSH protocol uses symmetric encryption, asymmetric encryption and hashing in order to secure transmission of information.

SSH Connection b/w client and server happens in 3 stages

Verification of server by the client

Generation of session key to encrypt all the communication

Authentication of the client by server

#### Verification of the server by client

Client initiates ssh connection with server. Server listens to ssh connections by default on port 22. At this point is server identity verified, but we have two cases

Communication b/w client and server for the first time:

client is asked to authenticate the server manually by verifying the public key of the server. Once the key is verified, the server will be added to ~/.ssh/known\_hosts.

If the client is not accessing the server for the first time server’s identity will be matched with previously recorded information on known\_hosts file

#### Generation of SESSION KEY

After the server is verified, both parties negotiate a session key using some version **Diffie-Helmen** algorithm.

This generated session key is used for encrypting and decrypting the message during the session

#### Authentication of the client

Final step involves authentication of the client by server. Authentication is done by using **SSH key pair**

As soon as session key is generated the authentication happens as mentioned in the following steps

Client begins by sending an ID of the key pair it would like to authenticate

Server checks authorized\_keys file

If matching ID is found, server generates a random number and uses the public key to encrypt the number and sends this message

If the client has the correct private key, it will decrypt this message and obtain random

This random is combine with session and hash is generated. This hash is sent to server as an answer.

If the two values match, it proves that client has valid private key and client is authenticated

# Linux Classroom Series – 31/Jul/2020

### Constructing SSH Command

SSH Command is quite simple

# This will ask password

ssh username@<hostname/ip-address>

# If you are using key based authentication

ssh -i <path-to-key> username@<hostname/ip-address>

By default ssh runs on port 22, if ssh runs on any other port (8922)

ssh -p 8922 username@<hostname/ip-address>

ssh -p 8922 -i <path-to-key> username@<hostname/ip-address>

To run ssh command we need ssh client

In linux & MAC machines we already have terminal with ssh clients

Windows:

Latest Windows 10 or windows 2016 server or later: Microsoft has added ssh client and you can use it from powershell

You need to install ssh clients like

Git Bash

Putty

Cygwin

There is one file where ssh server configuration will be present on the linux machine (/etc/ssh/sshd\_config)

Now lets try to create a some user on centos machine

sudo adduser linuxadmin

sudo passwd linuxadmin

sudo usermod -aG wheel linuxadmin

Now lets try to generate key pairs on the ubuntu machine for the new user called as linuxadmin

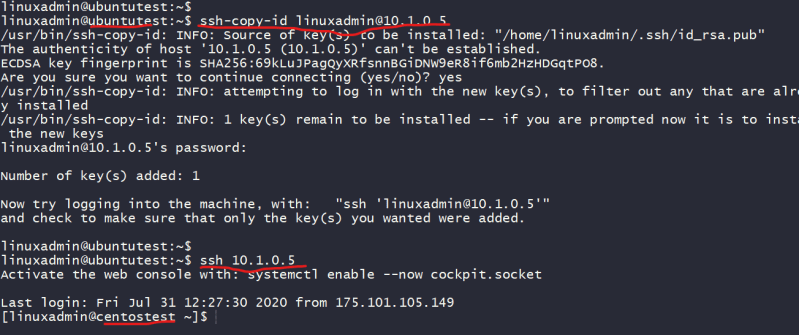
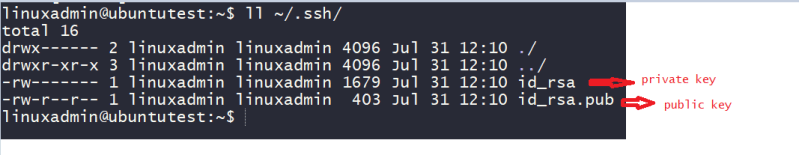
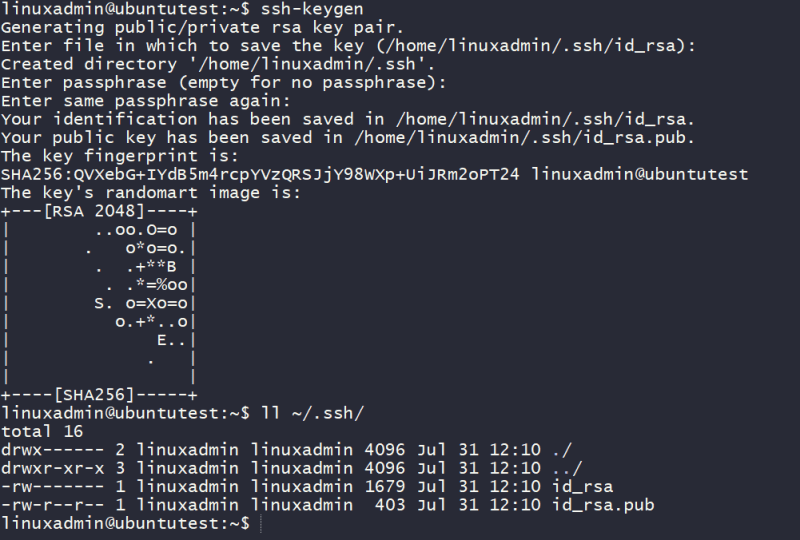
sudo adduser linuxadmin

sudo usermod -aG sudo linuxadmin

ssh-keygen

ssh-copy-id linuxadmin@<centosip>

ssh centosip



### SYSTEMD

At its core **systemd** is a system and startup manager. This has replaced traditional tools and subsystems

init

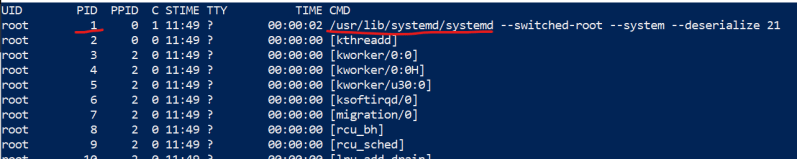
telinit

inittab

SysV

upstart

[Project Page link](https://www.freedesktop.org/wiki/Software/systemd/)

systemd is a suite of basic building blocks for a Linux System. It provides system and service manager that runs as PID1 and starts the rest of the system 

system provides aggressive parallelization capabilities uses socket and D-Bus activation for starting services.

Role:

systemd manages various system startup & shutdown functions. It also maintains startup and shutdown of services (daemon) on Linux OS.

systemd can monitor the services throughout the lifetime.

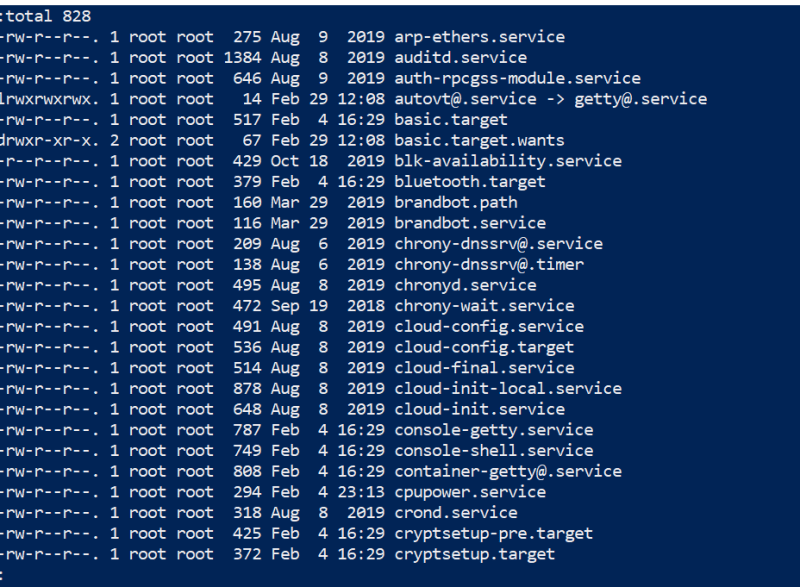
Systemd also no longer uses traditional shell scripts to store the configuration information for services which have been replaced by simple configuration files.

The objects that systemd manages are called as units & they are building blocks of systemd. Objects can be

services or daemons

devices

file system entries

Standard unit configurations are stored under **/usr/lib/systemd/system** 

Any new unit files as well as any needed customizations to existing unit files should be copied over to **/etc/systemd/system** for actual usage.

The following types of units exists

Service units: The units include traditional system daemons or services, These daemons can be started, stopped, restart, reloaded & enabled.

Service units are stored with .service extensions in **/etc/systemd/system**

ls -al \*.service

Socket units: These units consists of local and network sockets that are used for inter process communication in a system.

Socket units are stored with .socket extensions int **/etc/systemd/system**

ls -al \*.socket

Device units: These units allow systemd to see & use kernel devices.

Device units are stored with .device extensions int **/etc/systemd/system**

ls -al \*.device

Mount units: These units are used for mounting and unmounting file systems

Mount units are stored with .mount extensions int **/etc/systemd/system**

ls -al \*.mount

Target units: systemd uses target units for logical grouping of units.

target units are stored with .target extensions int **/etc/systemd/system**

ls -al \*.target

Timer units: These units are used for triggering activations of other units base on timers.

timer units are stored with .timer extensions int **/etc/systemd/system**

ls -al \*.timer

Snapshot unit: These units are used to save the state of set of systemd units temporarily.

snapshot units are stored with .snapshot extensions int **/etc/systemd/system**

ls -al \*.snapshot

Systemctl commandline for viewing units

sudo systemctl list-units --type=target

sudo systemctl list-units --type=mount --all

sudo systemctl list-units --all

sudo systemctl is-enabled sshd.service

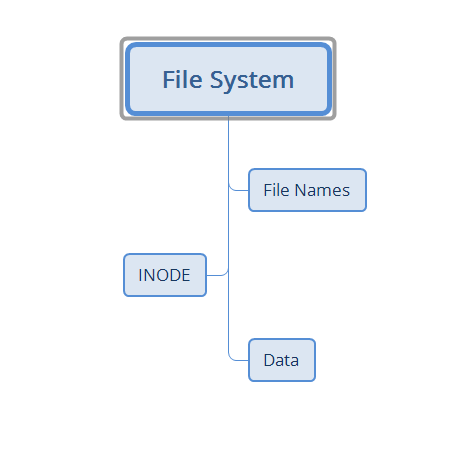
sudo systemctl start NAME.service

sudo systemctl enable NAME.service

sudo systemctl stop NAME.service

sudo systemctl disable NAME.service

#### Basic concepts in File Systems



#### i-Nodes

Inode is a data structure and it defines file or a directory on the filesystem

Inodes point to blocks that makeup a file

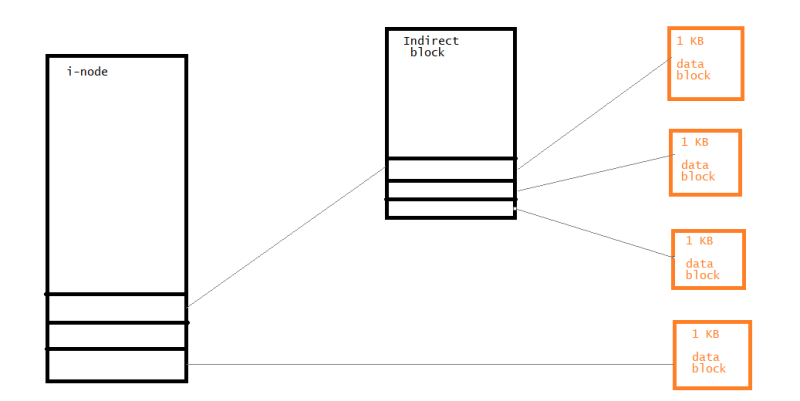
The control information in the i-node includes

file owner

time of last acess

creation time

group Id

Each directory gets an i-node & inode points to the datablocks containing information (filenames & i-nodes) 

#### Block

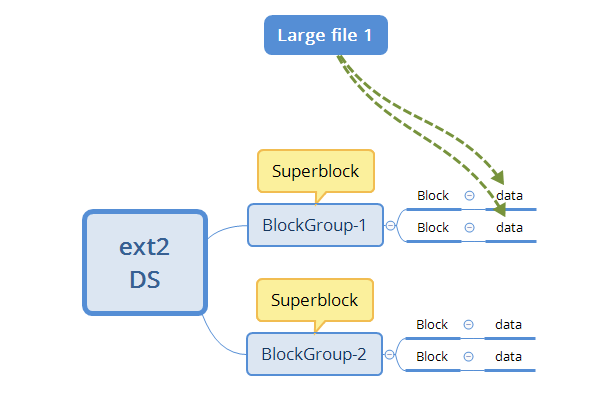
Data on ext\* file system is organized into block, A block is sequence of bytes.

Each block smallest addressable unit in storage device

Blocks are in turn grouped into block groups & each block group consists of superblock.

Superblock is first piece of information read from disk, This small datastructure reveals information such as

available space

location of i-node 

#### Filesystems in linux

The popular filesystems in linux are

ext4:

Supports maximum file system size of 1EB (exabyte)

Supports maximum individual file sizes up to 16TB each.

XFS

Journal base 64 bit file system.

Supports files systems as large as 8EiB (Exbibytes == 8 million terrabytes)

Supports maximum individual file sizes up to 8EiB (Exbibytes == 8 million terrabytes)

Btrfs:

Next generation Linux file system. In addition to all the features of ext4, Btrfs supports

Dynamic i-node allocation and transparent compressing

Online file system checking

Buit-in RAID functions such as mirroring & stripping

#### Commands useful for storage

Disk Utilization:

The du command allows you to determine disk utilization on a directory-by-directory basis

For du examples [Refer Here](https://www.google.com/search?q=linux+command+du+examples&oq=linux+command+du+examples&aqs=chrome..69i57j0.8001j1j7&sourceid=chrome&ie=UTF-8)

To display total amount of space used by all the files & directories in PWD

du -sh .

Disk free:

The df program displays the amount of free space available on the mounted file system.

To show free space for all locally mounted drives use this command

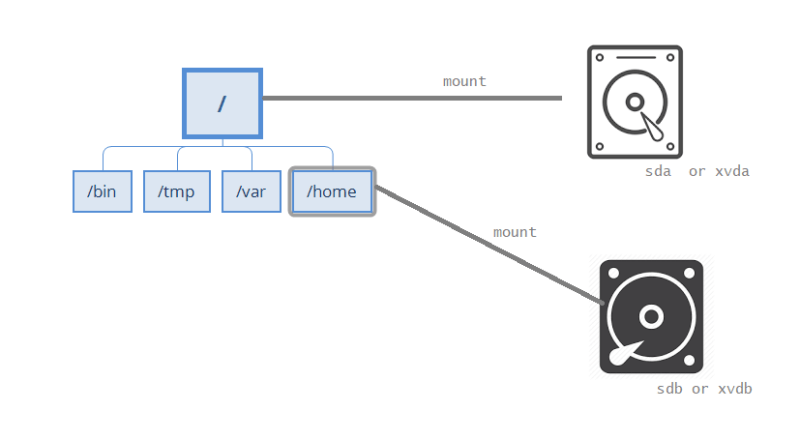
sudo df -l

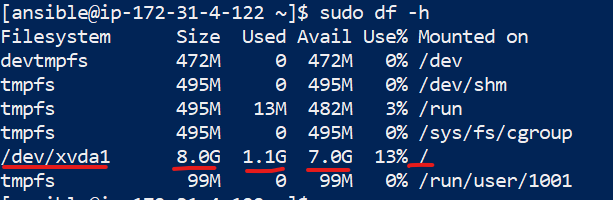
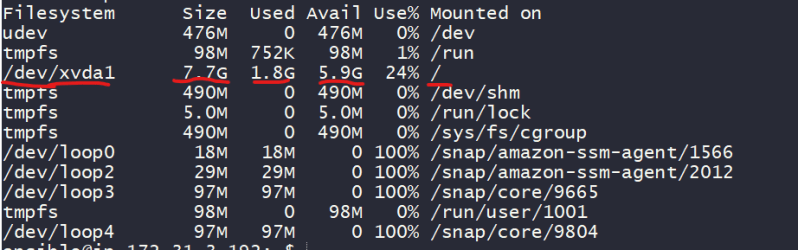
To show the free space in human-readable format

sudo df -h

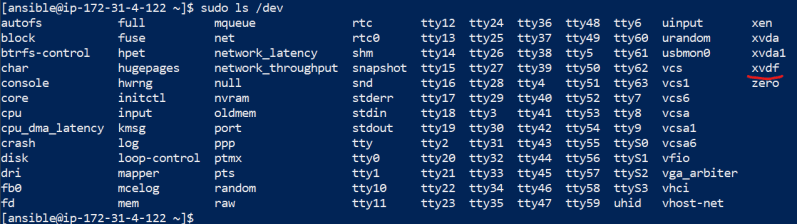
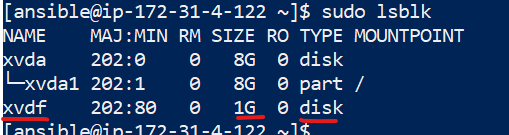
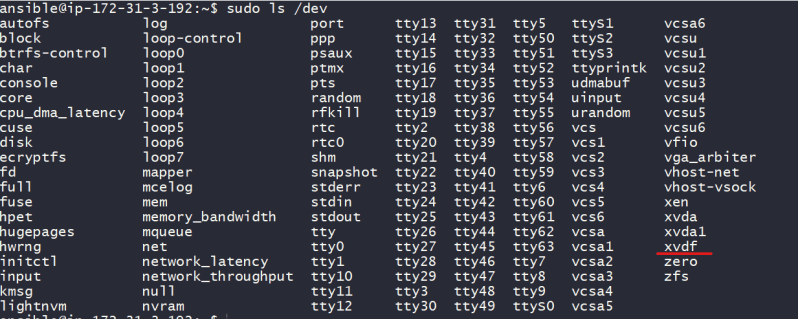
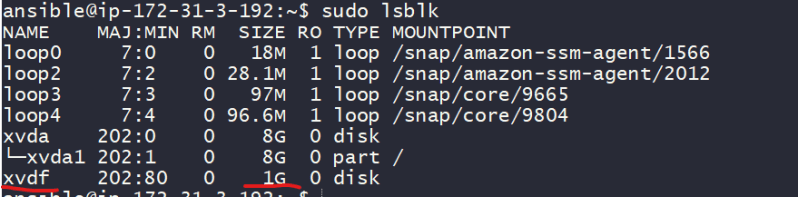
### Managing File Systems

#### Mounting & Unmounting Local disks

Filesystem on a partition or volume is mounted so that it appears as another subdirectory on the system. 

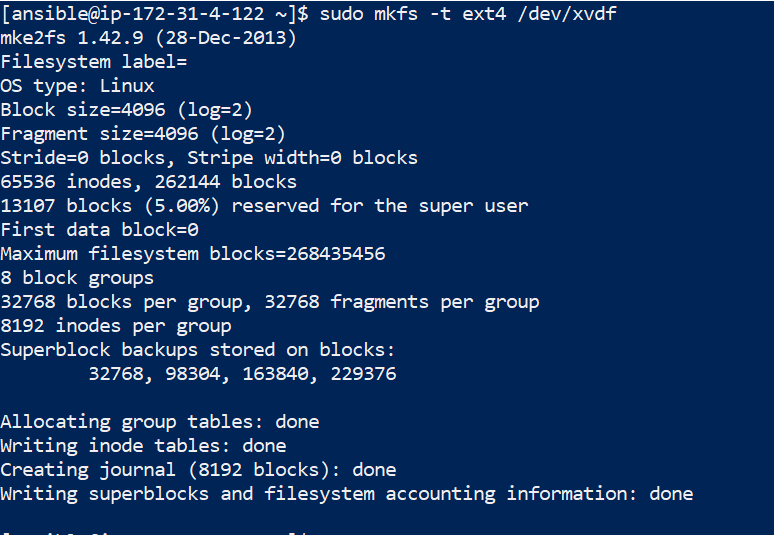
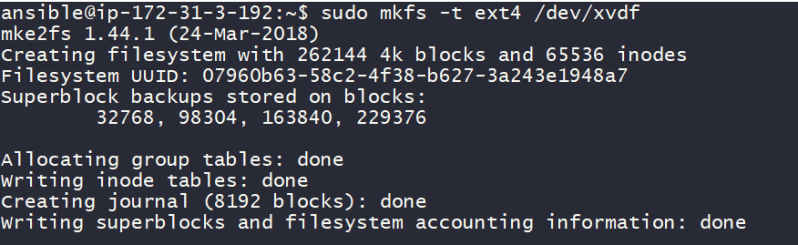
Lets execute a simple command sudo df -h on ubuntu (terminal in black) & centos (terminal in blue) 

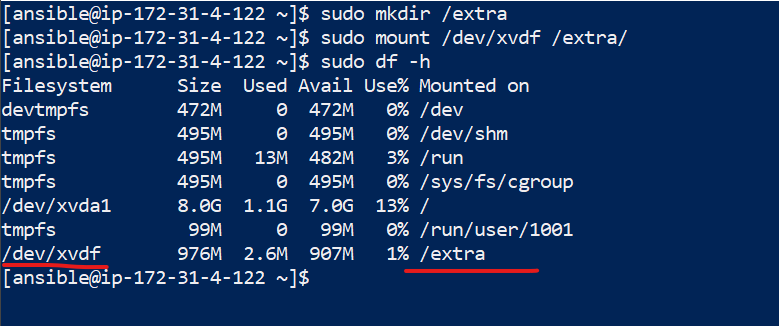
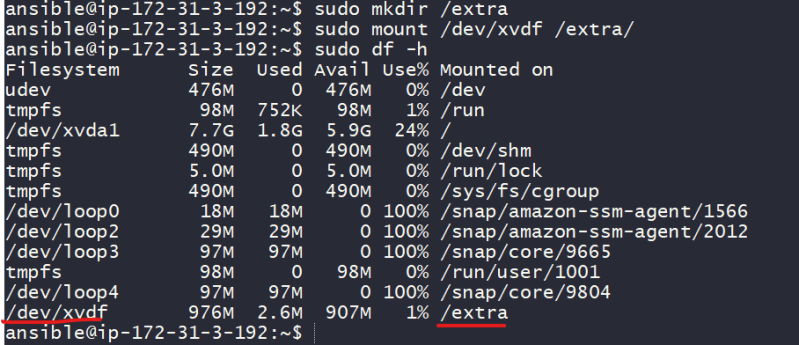
Lets add a virtual disk to an ec2 instance

Now execute lsblk 

Since we need to create file system you can choose any file system of your choice (ext4, xfs, btrfs)

sudo mkfs -t ext4 /dev/xvdf

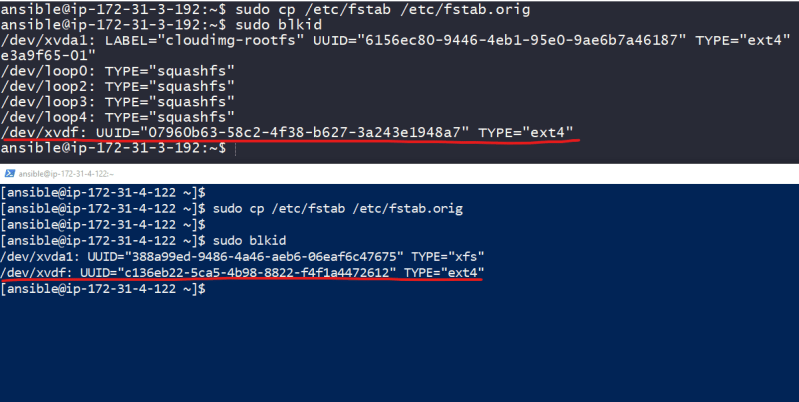


Lets create a dummy directory /extra, mount and list the disk 

Now if we restart the machine and check the mounts, mounts will not appear as linux will mount only the disks which have entries in /etc/fstab

Lets have a backup of current fstab

sudo cp /etc/fstab /etc/fstab.orig

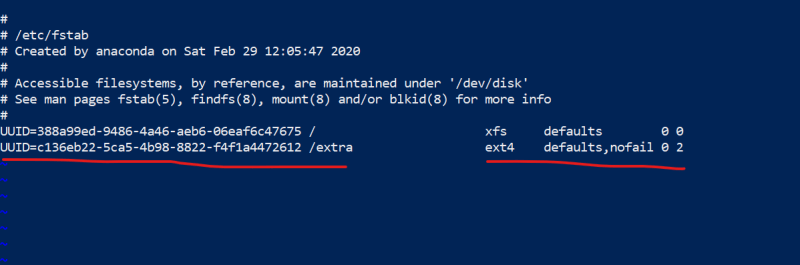
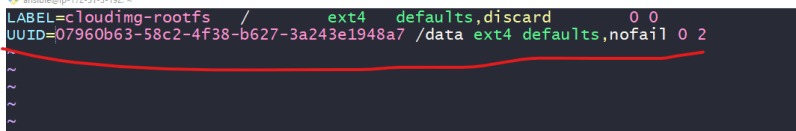
Lets list the block ids 

Now lets edit fstab to add entries, The basic format is

/dev/device /dir/to/mount fstype Paramters fs\_freq fs\_passno

# or

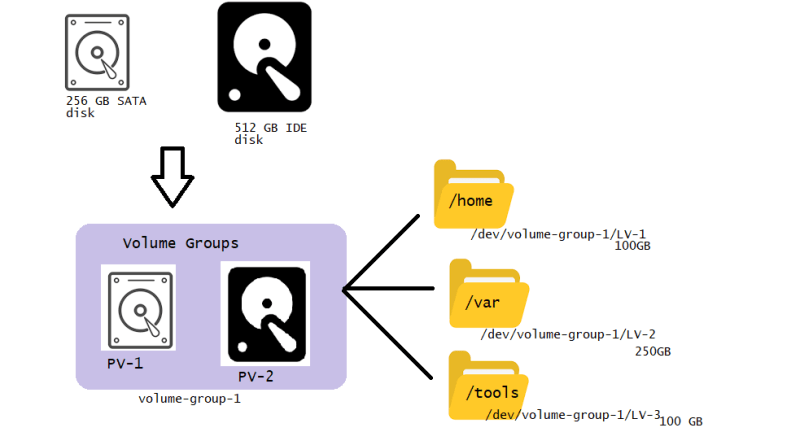
UUID=<uuid> <directory to mount> <filesystem> Paramters fs\_freq fs\_passno



Now even after restarts your mounts appear.

Exercise findout what defaults,nofail 0 2 is

### Logical Volume Management (LVM)

The Below illustration shows the relationships between disks, physical volumes (PVs), volume Groups (VGs) and Logical volumes in LVM: 

Volume management is a approach dealing with disks & partitions. The new approach to deal with this partitions is called as Logical Volume Management (LVM)

Benefits of using LVM

Greater flexibility for disk partitioning

Easier online resizing of volumes

Use of Snapshots

Terms:

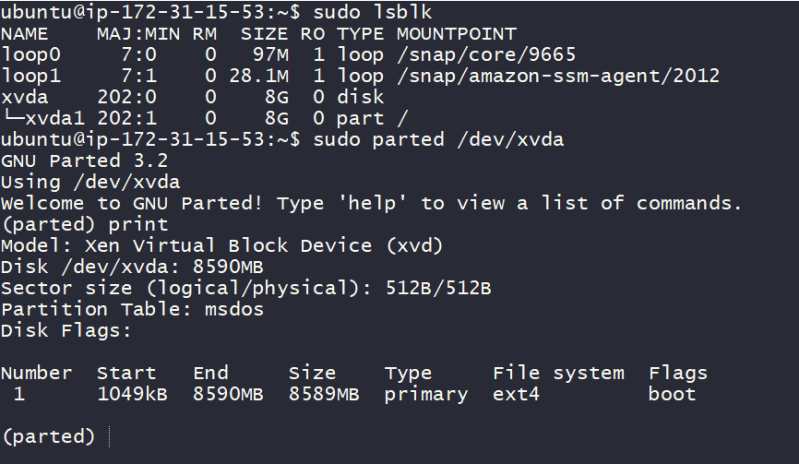
Physical Volume(PV): This represents physical disk.

Volume Group (VG): VGs are used to house one or more physical volumes & logical volumes into single administrative unit.

Logical Volume (LV): LVs are equivalent to disk partitions. LV appears as standard block device. LV can be mounted on to directories

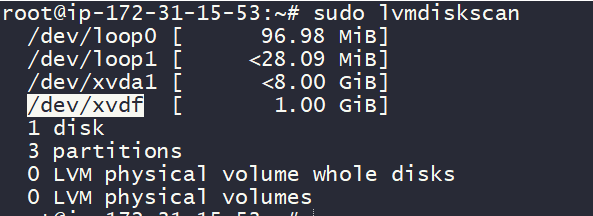
Extents: Two kinds of extents physical extents & logical extents. PVs can be divided into physical extents. LVs can be divided into logical extents

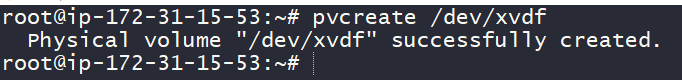
#### Examining Disk Layout

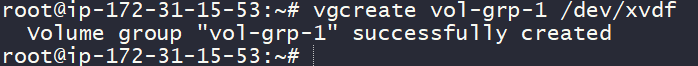
Open the parted prompt & execute print statement 

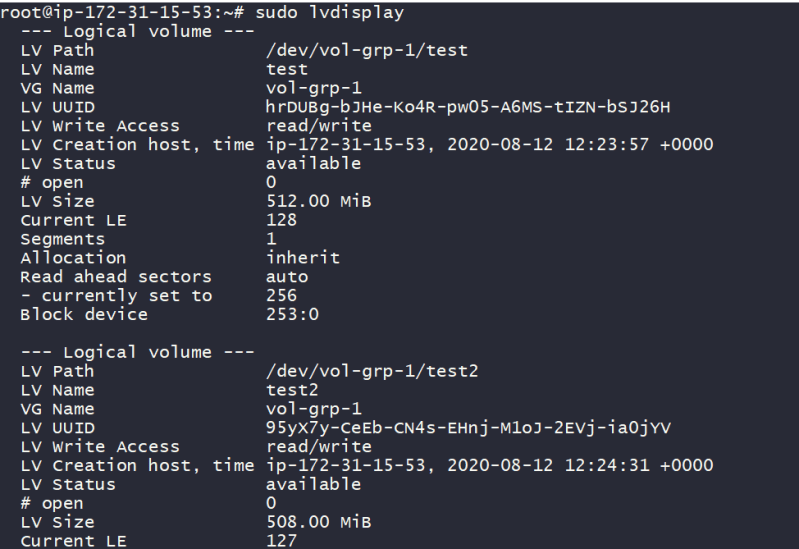
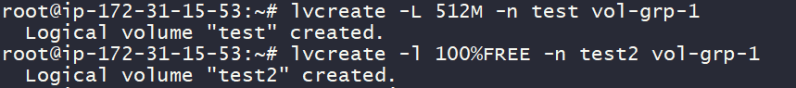
Physical volumes can be explored using pvdisplay & volume groups can be explored using vgdisplay & Logical volumes using lvdisplay

Lets experiment

scan disks 

Lets create a physical volume using pvcreate 

Now lets create volume groups using vgcreate 

Now lets create a Logical volume using lvcreate 

Now create a file system in two lvs and mount it asper your needs