



# LINUX 7&8

## WITH SHELL SCRIPTING

## REFRENCE CUM LAB MANUAL

By Musabuddin Syed  
Redhat certified

### VirtualPath Techno Solutions

#101, A Block  
Balaji Towers, Aster Prime Hospital Lane.  
Ameerpet, Hyd -38

E-Mail: [info@virtualpathtech.com](mailto:info@virtualpathtech.com)

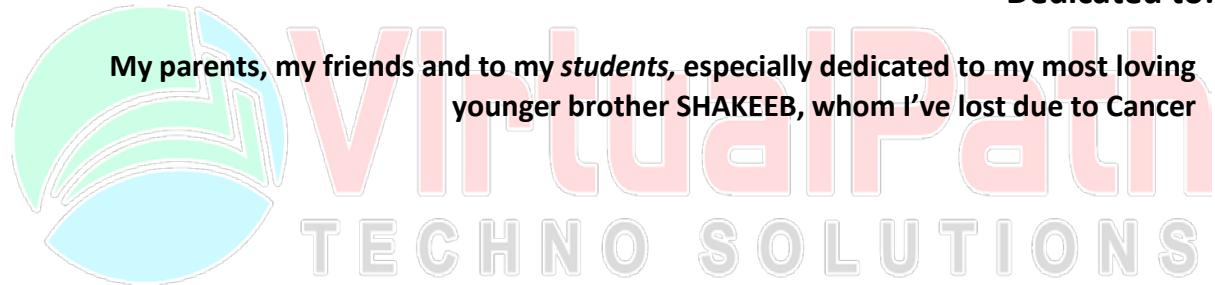
Phone:+91 799 309 6092, 040-6666 6092  
website: [www.virtualpathtech.com](http://www.virtualpathtech.com)  
website: [www.musab.in](http://www.musab.in)



**LINUX 6&7 COMPARITIVE STUDY GUIDE CUM LAB MANUAL:**

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied

**Dedicated to:**



**Our Address:**

**VirtualPath Techno Solutions:**

#101 -A Block,  
Balaji Towers,  
Prime Hospital Lane,  
Ameerpet, Hyderabad-500038

Phone:+91 799 309 6092, 040-6666 6092

Email: [info@virtualpathtech.com](mailto:info@virtualpathtech.com)

website: [www.virtualpathtech.com](http://www.virtualpathtech.com)

website: [www.musab.in](http://www.musab.in)

### **Trademarks**

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of Red Hat or other companies

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, is trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

XFS® is a trademark of Silico n Graphics International Corp. or its subsidiaries in the United States and/or other countries

## FOREWORD

### About the Author:

Musabuddin Syed is a highly acclaimed trainer, author and solutions provider. He regularly trains students in in-house, online and corporate at VirtualPath Techno Solutions. He has an experience of more than 10 years in industry and Training, where he has delivered more than 400 batches successfully in various technologies.

### I'm Highly Indebted To:

*Almighty God*

*My Family*

*KernelSphere Technologies*

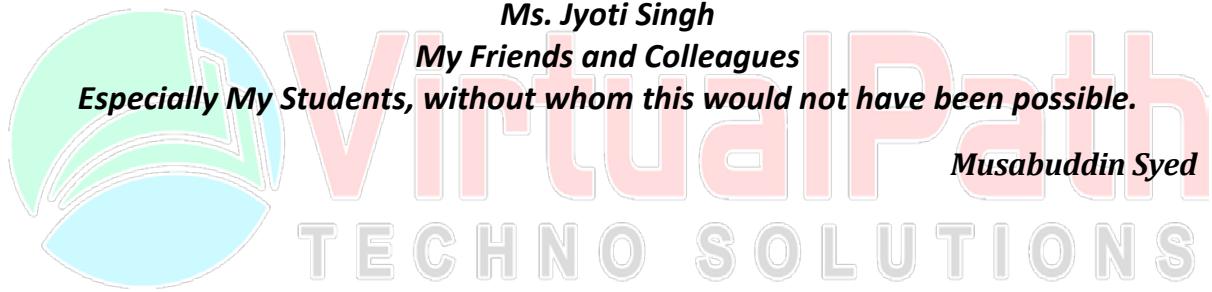
*Mr. Vinod Kumar*

*Ms. Jyoti Singh*

*My Friends and Colleagues*

*Especially My Students, without whom this would not have been possible.*

*Musabuddin Syed*



### Words to the Students

Though we have taken utmost efforts to present you this book error free, but still it may contain some errors or mistakes. Students are encouraged to bring, if there are any mistakes or errors in this document to our notice. So that it may be rectified in the next edition of this document.

This document provides good information on every topic and lab practices. This could become more effective if equally good practice is done. I urge the readers/students to do rigorous practice to polish your skill sets.

You can reach us on the following email address

[info@virtualpathtech.com](mailto:info@virtualpathtech.com)

[musab@virtualpathtech.com](mailto:musab@virtualpathtech.com)

[musabsyd@gmail.com](mailto:musabsyd@gmail.com)

Go through our website and blog

[www.virtualpathtech.com](http://www.virtualpathtech.com)

[www.musab.in](http://www.musab.in)

**MUSAB.IN**   
Training. Solutions & more.

## OTHER COURSES AT VIRTUALPATH

### LINUX CLUSTERS

DEVOPS

AWS

OPENSTACK

IBM & EMC SAN

NETAPP &  
NETAPP CLUSTER

VMWARE

ORACLE DBA, RAC  
& GOLDEN GATE

IBM AIX, POWER  
HA, LPAR VIO

SHELL SCRIPTING

And Many More...

# Table of Contents

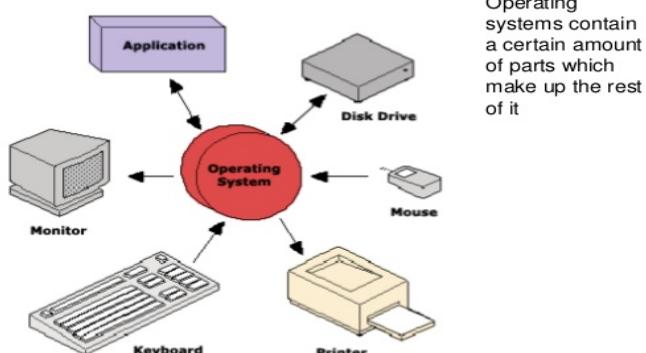
1. Introduction to Linux .....	01-07
2. Basic Commands.....	08-26
3. RHEL 8 Basic Installation .....	27-38
4. Managing File Systems and Partitions .....	39-49
5. Swap Spaces Management .....	50-53
6. Logical Volume Management (LVM) .....	54-67
7. RHEL 7 LVM based Installation .....	68-70
8. User and Group Administration .....	71-81
9. Controlling Access to the Files .....	82-89
10. Enhanced User Security with SUDO .....	90-97
11. Network Configuration and Troubleshooting.....	98-110
12. Managing SELinux (Basics) .....	111-120
13. Booting Procedure of RHEL7/8 and Troubleshooting .....	121-129
14. Manage Installed Services .....	130-133
15. Introduction to Firewalld .....	134-137
16. Introduction to Cokpit in RHEL8 .....	138-140
17. Backup and Restore (tar&gzip)	141-143
18. Job Automation with Cronjobs .....	144-149
19. Administrating Remote System .....	150-158
20. Software Management .....	159-179
21. Managing Processes .....	180-194
22. FTP (File Transfer Protocol) Server .....	195-207
23. NFS (Network File System) Server .....	208-216
24. Samba Server .....	217-224
25. DNS (Domain Name System) Server .....	225-234
26. Mail Server .....	235-239
27. Web Server (Apache) .....	240-250
28. Kickstart and Network Installations .....	251-261
 <u>BONUS SERVERS &amp; TOPICS FOR SELF-LEARNING</u>	
29. Squid Proxy Server .....	263-269
30. DHCP Server .....	270-275
<b>SHELL SCRIPTING .....</b>	<b>276</b>



***This page has been left blank intentionally***

## INTRODUCTION TO UNIX & LINUX

### What is an operating system?



Operating systems contain a certain amount of parts which make up the rest of it

### What is Operating System?

Operating system is an interface between user and the computer hardware. The hardware of the computer cannot understand the human readable language as it works on binaries i.e. 0's and 1's. Also it is very tough for humans to understand the binary language, in such case we need an interface which can translate human language to hardware and vice-versa for effective communication.

### Types of Operating System:

- Single User - Single Tasking Operating System
- Single User - Multitasking Operating System
- Multi User - Multitasking Operating System

### Single User - Single Tasking Operating System

In this type of operating system only one user can log into system and can perform only one task at a time.

E.g.: MS-DOS

### Single User - Multi tasking operating System

This type of O/S supports only one user to log into the system but a user can perform multiple tasks at a time, browsing internet while playing songs etc.

E.g.: Windows -98, XP, vista, 7,8,10 etc.

### Multi User - Multi Tasking Operating System

This type of O/S provides multiple users to log into the system and also each user can perform various tasks at a time. In a broader term multiple users can logged in to system and share the resources of the system at the same time.

E.g.: UNIX, LINUX etc.

## HISTORY OF UNIX

### In the beginning, there was AT&T.

Bell Labs' Ken Thompson developed UNIX in 1969 so he could play games on a scavenged DEC PDP-7. With the help of Dennis Ritchie, the inventor of the "C" programming language, Ken rewrote UNIX entirely in "C" so that it could be used on different computers. In 1974, the OS was licensed to universities for educational purposes. Over the years, hundreds of people added and improved upon the system, and it spread into the commercial world. Dozens of different UNIX "flavors" appeared, each with unique qualities, yet still having enough similarities to the original AT&T version. All of the "flavors" were based on either AT&T's System V or Berkeley System Distribution (BSD) UNIX, or a hybrid of both.

### During the late 1980's there were several of commercial implementations of UNIX:

- Apple Computer's A/UX
- AT&T's System V Release 3
- Digital Equipment Corporation's Ultrix and OSF/1 (renamed to DEC UNIX)
- Hewlett Packard's HP-UX
- IBM's AIX
- Lynx's Real-Time UNIX
- NeXT's NeXTStep
- Santa Cruz Operation's SCO UNIX
- Silicon Graphics' IRIX
- SUN Microsystems' SUN OS and Solaris
- and dozens more.

The Open Standards Foundation is a UNIX industry organization designed to keep the various UNIX flavors working together. They created operating systems guidelines called POSIX to encourage inter-operability of applications from one flavor of UNIX to another. Portability of applications to different gave UNIX a distinct advantage over its mainframe competition.

Then came the GUIs. Apple's Macintosh operating system and Microsoft's Windows operating environment simplified computing tasks, and made computers more appealing to a larger number of users. UNIX wizards enjoyed the power of the command line interface, but acknowledged the difficult learning curve for new users. The Athena Project at MIT developed the X Windows Graphical User Interface for UNIX computers. Also known as the X11 environment, corporations developed their own "flavors" of the UNIX GUIs based on X11. Eventually, a GUI standard called Motif was generally accepted by the corporations and academia.

During the late 1990's Microsoft's Windows NT operating system started encroaching into traditional UNIX businesses such as banking and high-end graphics. Although not as reliable as UNIX, NT became popular because of the lower learning curve and its similarities to Windows 95 and 98. Many traditional

UNIX companies, such as DEC and Silicon Graphics abandoned their OS for NT. Others, such as SUN, focused their efforts on niche markets, such as the Internet.

Linus Torvalds had a dream. He wanted to create the coolest operating system in the world that was free for anyone to use and modify. Based on an obscure UNIX flavor called MINIX, Linus took the source code and created his own flavor, called Linux. Using the power of the Internet, he distributed copies of his OS all over the world, and fellow programmers improved upon his work. In 1999, with a dozen versions of the OS and many GUIs to choose from, Linux is causing a UNIX revival. Knowing that people are used to the Windows tools, Linux developers are making applications that combine the best of Windows with the best of UNIX.

## UNIX Principles

- **Everything is a file:-** UNIX system have many powerful utilities designed to create and manipulate files. The UNIX security model is based around the security of files. By treating everything as a file, you can secure access to hardware in the same way as you secure access to a document.
- **Configuration data stored in text:** - Storing configuration in text allows an administrator to move a configuration from one machine to another easily, provide the ability to roll back a system configuration to a particular date and time.
- **Small, Single-Purpose Programs:** - UNIX provides many utilities.
- **Avoid captive user interfaces:-**
- **Ability to chain programs together to perform complex tasks:-** A core design feature of UNIX is that output of one program can be the input for another. This gives the user the flexibility to combine many small programs together to perform a larger, more complex task.

## GNU Project/ FSF

- GNU project started in 1984
  - a) Goal: Create 'free' UNIX clone
  - b) By 1990, nearly all required user space application created.  
Example:-gcc, emacs, etc.
- Free Software Foundation
  - a) Non-Profit organization that manages the GNU project.

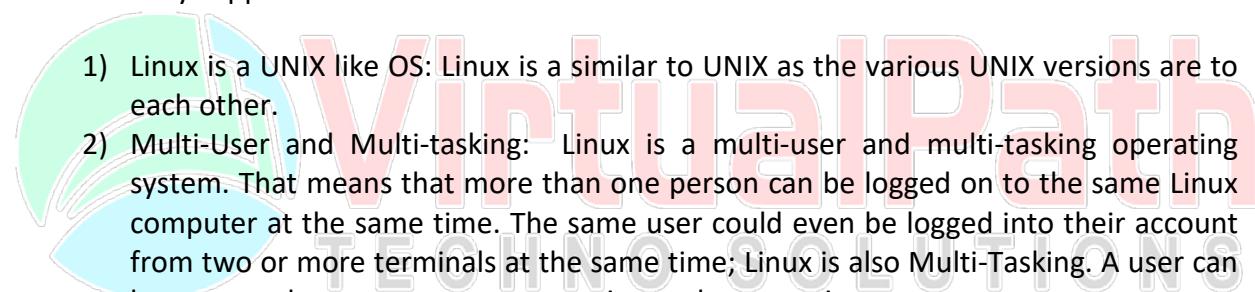
## GPL – GNU (General Public License)

- primary license for open source software
- encourages free software
- All enhancements and changes to GPL software must also be GPL
- Often called 'copy left' (All rights reversed)

## Linux Origins

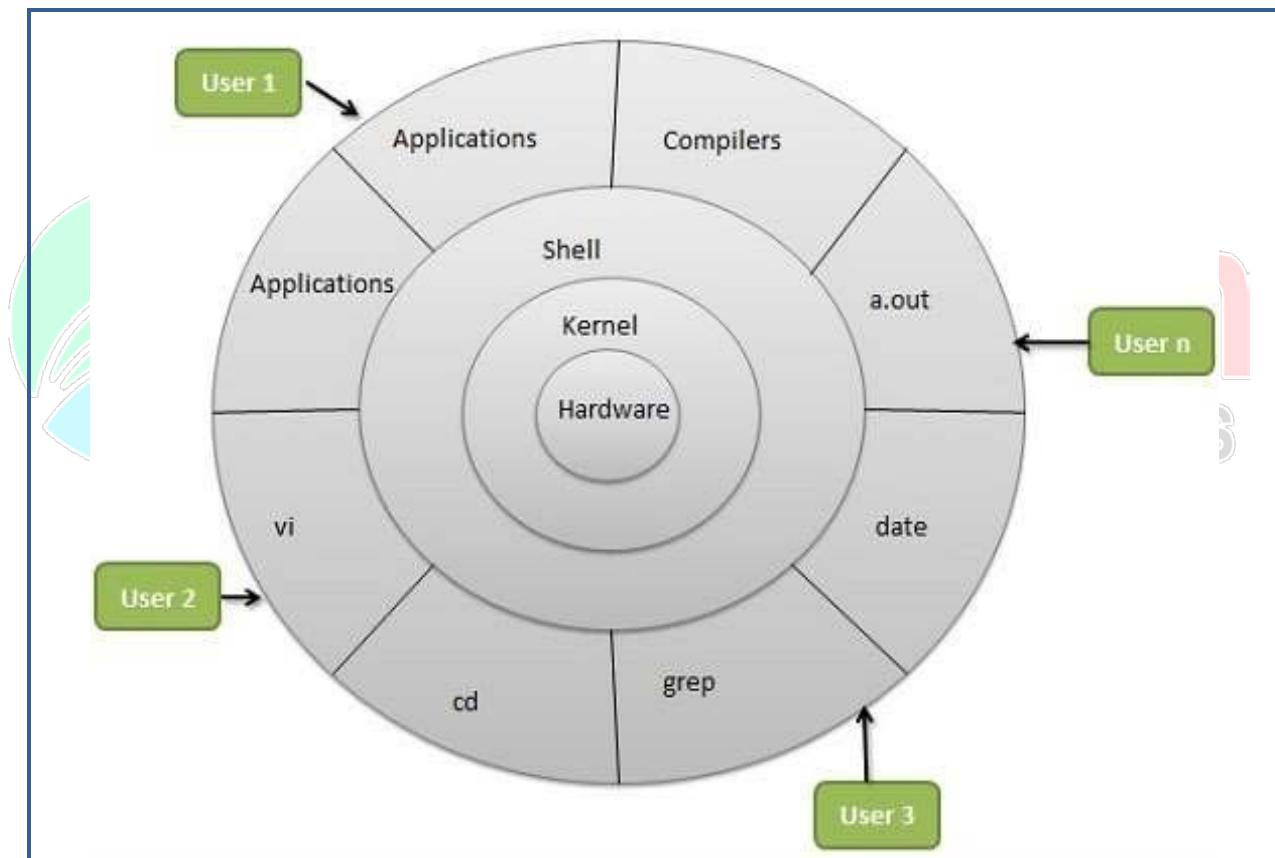
- **LINUS TORVALDS**
  - a) Finnish college student in 1991
  - b) Created Linux Kernel
- When Linux Kernel combined with GNU applications, complete free UNIX like OS was developed.

## Why Linux?

- Fresh implementation of UNIX APIs
  - Open source development model
  - Supports wide variety of hardware
  - Supports many networking protocols and Configurations
  - Fully supported
- 
- 1) Linux is a UNIX like OS: Linux is a similar to UNIX as the various UNIX versions are to each other.
  - 2) Multi-User and Multi-tasking: Linux is a multi-user and multi-tasking operating system. That means that more than one person can be logged on to the same Linux computer at the same time. The same user could even be logged into their account from two or more terminals at the same time; Linux is also Multi-Tasking. A user can have more than one program executing at the same time.
  - 3) Wide hardware support: Red Hat Linux support most pieces modern x86 compatible PC hardware
  - 4) Fully Supported: Red Hat Linux is a fully supported distribution Red Hat Inc. provides many support programs for the smallest to the largest companies.

## ARCHITECTURE OF UNIX

The architecture of UNIX can be divided into three levels of functionality, as shown in Figure. The lowest level is the *kernel*, which schedules tasks, manages resources, and controls security. The next level is the *shell*, which acts as the user interface, interpreting user commands and starting applications. The highest level is *utilities*, which provides utility functions. In other words it is the USER level, as user is the one who operates those utilities.



## FILESYSTEM HIERARCHY

Linux uses single rooted, inverted tree like file system hierarchy

/

This is top level directory  
It is parent directory for all other directories  
It is called as ROOT directory  
It is represented by forward slash (/)  
C:\ of windows

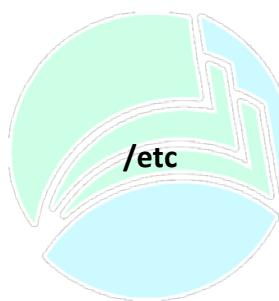
/root

it is home directory for root user (super user)  
It provides working environment for root user  
C:\Documents and Settings\Administrator

/home

it is home directory for other users  
It provide working environment for other users (other than root)  
c:\Documents and Settings\username

/boot



it contains bootable files for Linux  
Like vmlinuz (kernel).... ntoskrnl  
Initrd (INITial Ram Disk)and  
GRUB (GRand Unified Boot loader).... boot.ini, ntldr

it contains all configuration files  
Like /etc/passwd..... User info  
/etc/resolv.conf... Preferred DNS  
/etc/dhcpd.conf.... DHCP server  
C:\windows\system32\dirvers\

/usr

by default soft wares are installed in /usr directory  
(UNIX Sharable Resources)  
c:\program files

/opt

It is optional directory for /usr  
It contains third party softwares  
c:\program files

/bin

it contains commands used by all users  
(Binary files)

/sbin

it contains commands used by only Super User (root)  
(Super user's binary files)

/dev

it contains device files  
Like /dev/hda ... for hard disk  
/dev/cd rom ... for cd rom  
Similar to device manager of windows

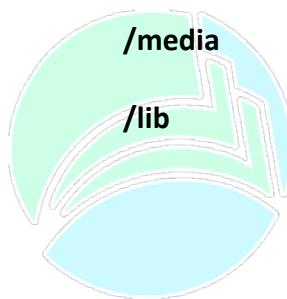
**/proc** it contain process files  
Its contents are not permanent, they keep changing  
It is also called as Virtual Directory  
Its file contain useful information used by OS  
Like /proc/meminfo... information of RAM/SWAP  
/proc/cpuinfo... information of CPU

**/var** it contains variable data like mails, log files

**/tmp** contains the temporary files for small period of time

**/mnt** it is default mount point for any partition  
It is empty by default

**/media** it contains all of removable media like CD-ROM, pen drive  
**/lib** it contains library files which are used by OS  
It is similar to dll files of windows  
Library files in Linux are SO (shared object) files



**VirtualPath**  
**TECHNO SOLUTIONS**

## UNIX BASIC COMMANDS

### Creating, Removing, Copying, Moving files & Directories

#### Creating a file in Linux

##### Using cat command:

- cat (Concatenate) command is used to create a file and to display and modify the contents of a file.
- **To create a file**

```
# cat > filename (say myfile)
```

Hello World

Ctrl+d (To save the file)

```
[root@musab1 ~]# cat > myfile
HELLO WORLD
[root@musab1 ~]#
```

##### To display the content of the file

```
# cat filename (say myfile)
```

```
[root@musab1 ~]# cat myfile
HELLO WORLD
[root@musab1 ~]#
```

##### To append the data in the already existing file

```
# cat >> <filename>
```

```
# cat >> myfile
```

Ctrl+d (to exit back)

```
[root@musab1 ~]# cat >> myfile
WELCOME TO LINUX
[root@musab1 ~]#
```

#### Creating multiple files at same time using touch command

```
#touch <filename> <filename> <filename>
```

```
#touch file1 file2 file3
```

Note: to check the files use # ls command

```
[root@musab1 ~]# touch file1 file2 file3
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file1  file3
Desktop          Downloads  file2  install.log
```

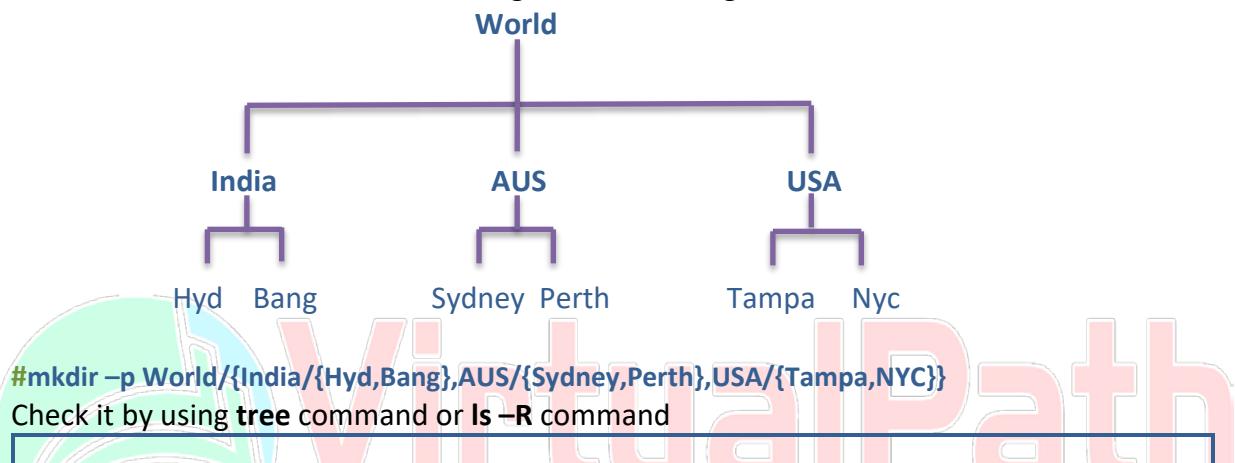
## Creating a Directory

```
#mkdir <dir name>
#mkdir mydir
```

```
[root@musab1 ~]# mkdir mydir
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file1  file3      install.log.syslog  mydir
Desktop          Downloads  file2  install.log  Music           myfile
```

## Making multiple directories inside a directory

Let us make some directories according to the following architecture in one command.



```
[root@musab1 ~]# mkdir -p World/{India/{Hyd,Bang},AUS/{Sydney,Perth},USA/{Tampa, NYC}}
[root@musab1 ~]# tree World
World
├── AUS
│   ├── Perth
│   └── Sydney
├── India
│   ├── Bang
│   └── Hyd
└── USA
    ├── NYC
    └── Tampa

9 directories, 0 files
[root@musab1 ~]#
```

## Copying files into directory

```
#cp <source filename> <destination directory in which to paste the file>
#cp file1 mydir
```

```
[root@musab1 ~]# cp file1 mydir
[root@musab1 ~]# cd mydir
[root@musab1 mydir]# ls
file1
```

### Copying directories from one location to other

```
# cp -rvfp <dir name> <destination name>
#cp -rvfp mydir2 mydir
```

```
[root@musab1 ~]# cp -rvfp mydir2 mydir
`mydir2' -> `mydir/mydir2'
`mydir2/file5' -> `mydir/mydir2/file5'
`mydir2/file2' -> `mydir/mydir2/file2'
`mydir2/file1' -> `mydir/mydir2/file1'
`mydir2/file4' -> `mydir/mydir2/file4'
`mydir2/file3' -> `mydir/mydir2/file3'
[root@musab1 ~]# cd mydir
[root@musab1 mydir]# ls
file1  mydir2
```

### Moving files from one location to other (cut and Paste)

```
#mv <filename> <Destination directory>
#mv myfile mydir
```

```
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file1  file3      install.log.syslog  mydir  myfile
Desktop          Downloads   file2  install.log  Music           mydir2 -p
[root@musab1 ~]# mv myfile mydir
[root@musab1 ~]# cd mydir
[root@musab1 mydir]# ls
file1  mydir2  myfile
```

### Moving a Directory from one location to other

```
#mv <dir name> <destination dir name>
#mv mydir mydir2
```

```
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file1  file3      install.log.syslog  mydir  -p
Desktop          Downloads   file2  install.log  Music           mydir2  Pictures
[root@musab1 ~]# mv mydir mydir2
[root@musab1 ~]# cd mydir2
[root@musab1 mydir2]# ls
file1  file2  file3  file4  file5  mydir
[root@musab1 mydir2]#
```

### Renaming a File

```
#mv <old name> <new name>
#mv file1 newfile
```

```
[root@musab1 mydir]# ls
file1  mydir2  myfile
[root@musab1 mydir]# cat myfile
HELLO WORLD
WELCOME TO LINUX
[root@musab1 mydir]# mv myfile new
[root@musab1 mydir]# ls
file1  mydir2  new
[root@musab1 mydir]# cat new
HELLO WORLD
WELCOME TO LINUX
[root@musab1 mydir]#
```

## Renaming a Directory

- The procedure and command for renaming the directory is exactly same as renaming a file.

```
#mv old name new name
#mv mydir newdir
```

```
[root@musab1 mydir2]# ls
file1 file2 file3 file4 file5 mydir
[root@musab1 mydir2]# ls mydir
file1 mydir2 new
[root@musab1 mydir2]# mv mydir newdir
[root@musab1 mydir2]# ls
file1 file2 file3 file4 file5 newdir
[root@musab1 mydir2]# ls newdir
file1 mydir2 new
```

## Removing a File

#rm filename or #rm -f filename (without prompting)

```
[root@musab1 mydir2]# ls
file1 file2 file3 file4 file5 newdir
[root@musab1 mydir2]# rm file1
rm: remove regular empty file `file1'? y
```

*Without prompting:*

```
[root@musab1 mydir2]# rm -f file1
[root@musab1 mydir2]# ls
file2 file3 file4 file5 newdir
[root@musab1 mydir2]#
```

## Removing an Empty directory

#rmdir dirname

```
[root@musab1 ~]# ls
anaconda-ks.cfg Documents file2 install.log Music newdir -p
Desktop Downloads file3 install.log.syslog mydir2 newfile Pictures
[root@musab1 ~]# ls newdir
[root@musab1 ~]# rmdir newdir
[root@musab1 ~]# ls
anaconda-ks.cfg Documents file2 install.log Music newfile Pictures
Desktop Downloads file3 install.log.syslog mydir2 -p Public
[root@musab1 ~]#
```

### Removing a directory with files or directories inside

A dir which is having some contents inside it cannot be removed by **rmdir** command. There are two ways to delete the directory with contents.

- i. Remove the contents inside the directory and then run **rmdir** command
- ii. Run **#rm -rf dirname** (where r stands for recursive and f stands for forcefully).

```
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file2  install.log      Music
Desktop          Downloads  file3  install.log.syslog mydir2
[root@musab1 ~]# ls mydir2
file2  file3  file4  file5  newdir
[root@musab1 ~]# rm -rf mydir2
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file2  install.log      Music
Desktop          Downloads  file3  install.log.syslog newfile
[root@musab1 ~]#
```



It has 3 modes:

- 1 Command Mode
- 2 Insert mode (edit mode)
- 3 extended command mode

**Note:** When you open the vim editor, it will be in the command mode by default.

In the command mode the cursor's can be used as  
h/l/k/j to move cursor left/right/up/down

**Insert Mode:**

i	To begin insert mode at the cursor position
I	To insert at the beginning of line
a	To append to the next word's letter
A	To Append at the end of the line
o	To insert a new line below the cursor position
O	To insert a new line above the cursor position

### Command Mode:

gg	To go to the beginning of the file
G	To go to end of the file
w	To move the cursor forward, word by word
b	To move the cursor backward, word by word
nw	To move the cursor forward to n words (5W)
nb	To move the cursor backward to n words (5B)
u	To undo last change (word)
U	To undo the previous changes (entire line)
Ctrl+R	To redo the changes
yy	To copy a line
nyy	To copy n lines (5yy or 4yy)
p	To paste line below the cursor position
P	To paste line above the cursor position
dw	To delete the word letter by letter (like Backspace)
x	To delete the world letter by letter (like DEL Key)
dd	To delete entire line
ndd	To delete n no. of lines from cursor position(5dd)
/	To search a word in the file

### Extended Mode: (Colon Mode)

Extended Mode is used for save and quit or save without quit using "Esc" Key with ":"

Esc+:w	To Save the changes
Esc+:q	To quit (Without saving)
Esc+:wq	To save and quit
Esc+:w!	To save forcefully
Esc+wq!	To save and quit forcefully
Esc+:x	To save and quit
Esc+:X	To give password to the file and remove password
Esc+:20(n)	To go to line no 20 or n
Esc+: se nu	To set the line numbers to the file
Esc+:se nonu	To Remove the set line numbers

To open multiple files in vim editor

#vim -o file1 file2

To switch between files use Ctrl +w

### Listing files and directories:

#ls	list the file names
#ls -l	long listing of the file
#ls -l filename	to see the permissions of a particular file
#ls -al	shows the files in ascending order of modification.
#ls p*	All the files start with p.

#ls ?ample	Files with any first character and has ample
#ls -l*	Directory listing only
#ls -l directory name	to see the permissions of a particular directory
#ls [ae]*	First character of the filename must be a or e.
# ls [!ae]*	! Symbol complements the condition that follows. The characters must not be a or e.
#ls [a-m][c-z][4-9]	list all the files in specific range

### Types of Files:

Symbol	Type of File
-	Normal file
d	Directory
l	Link file (shortcut)
b	Block file (Harddisk, Floppy disk)
c	Character file (Keyboard, Mouse)

### Symbolic Link

There are two types of Links:-

	Soft Link	Hard link
1	Size of link file is equal to no. of characters in the name of original file	Size of both file is same
2	Can be created across the Partition	Can't be created across the partition
3	inode no. of source and link file is different	inode no. of both file is same
4	if original file is deleted, link is broken and data is lost	If original file is deleted then also link will contain data
5	SHORTCUT FILE	BACKUP FILE

### Creating a soft link:

# ln -s <source file> <destination>

```
[root@musabi ~]# ls
anaconda-ks.cfg  Documents  file2  install.log      Music      -p
Desktop          Downloads  file3  install.log.syslog newfile   Pictures
[root@musabi ~]# ln -s newfile softlink
[root@musabi ~]# ls -li newfile softlink
1445159 -rw-r--r--. 1 root root 0 Feb 13 13:55 newfile
1439034 lrwxrwxrwx. 1 root root 7 Feb 13 15:16 softlink -> newfile
[root@musabi ~]#
```

### Creating a Hard link:

```
#ln <source file> <Destination>
```

```
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file2  install.log      Music
Desktop          Downloads   file3  install.log.syslog newfile
[root@musab1 ~]# ln newfile hardlink
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file2  hardlink      install.log.syslog
Desktop          Downloads   file3  install.log    Music
[root@musab1 ~]# ls -li newfile hardlink
1445159 -rw-r--r--. 2 root root 0 Feb 13 13:55 hardlink
1445159 -rw-r--r--. 2 root root 0 Feb 13 13:55 newfile
[root@musab1 ~]#
```

## Regular Expressions, Pipelines & I/O Redirections

### Grep:

Grep stands for **Global Regular Expression Print**. It is used to pick out the required expression from the file and print the output. If grep is combined with another command it can be used to pick out the selected word, phrase from the output of first command and print it.

### Examples of Grep:

Let us pick the information about **root** from the file **/etc/passwd** (/etc/passwd contains information about all the users present in the system)

```
#grep root /etc/passwd
```

```
[root@ linux ~]# grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[root@ linux ~]#
```

To avoid case sensitivity of the word (i.e. the word may be uppercase or lowercase) use **-i**

**#grep -i linux test** (lets grep the word **linux** whether upper or lower case in the file **test**)

```
[root@musab1 ~]# grep -i linux test
LINUX is freedom
linux is freedom
Linux is freedom
[root@musab1 ~]#
```

To display a word and 2 lines after the word:

```
#grep -nA2 wheel /etc/group
```

```
[root@ linux ~]# grep -nA2 wheel /etc/group
11:wheel:x:10:root
12-mail:x:12:mail,postfix
13-uucp:x:14:uucp
[root@ linux ~]#
```

To display a word and 2 lines after the word:

```
#grep -nB2 wheel /etc/group
```

```
[root@ linux ~]# grep -nB2 wheel /etc/group
9-mem:x:8:
10-kmem:x:9:
11:wheel:x:10:root
```

To display the things except the given word

```
#grep -v world test
```

```
[root@musab1 ~]# cat test
linux is freedom
Hello world
Welcome to my world
[root@musab1 ~]# grep -v world test
linux is freedom
```

To display the searched word in color

```
#grep --color root /etc/passwd
```

#### Combining grep with other commands

# cat myfile | grep -i linux (pipe | is used to combine to commands)

#ls -l | grep -i myfile

# ifconfig |grep -i eth0

Like this we can combine grep with many commands which we will see in later chapters

**Filter Commands:**

- Filter commands are used to filter the output so that the required things can easily be picked up. The commands which are used to filter the output are

**#less  
#more  
#head  
#tail  
#sort  
#cut  
#sed**

- less:**

The **less** command is used to see the output line wise or page wise.

Ex: less /etc/passwd

```
root:x:0:0:root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin.sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

**Note:** -press **Enter** key to scroll down line by line (or)

Use **d** to go to next page

Use **b** to go to previous page

Use **/** to search for a word in the file

Use **v** to go vi mode where you can edit the file and once you save it you will back to less command

### more:

**more** is exactly same like **less**

Ex: #more /etc/passwd

**Note:** -press **Enter** key to scroll down line by line (or)

Use **d** to go to next page

Use **/** to search for a word in the file

Use **v** to go vi mode where you can edit the file and once you save it you will back to more command

### head:

It is used to display the top **10 lines** of the file.

Ex:# head /etc/passwd

```
[root@ linux ~]# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin.sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

### To display the custom lines

#head -n /etc/passwd (where n can be any number)

```
[root@ linux ~]# head -5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

### tail:

It is used to display the **last 10** lines of the file

#tail /etc/passwd

```
[root@ linux ~]# tail /etc/passwd
apache:x:48:48:Apache:/var/www:/sbin/nologin
nslcd:x:65:55:LDAP Client User:::/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
pulse:x:496:494:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72:::/sbin/nologin
visitor:x:500:500:visitor:/home/visitor:/bin/bash
```

### To display the custom lines

#tail -n /etc/passwd (where n can be any number)

```
[root@ linux ~]# tail -5 /etc/passwd
user:x:500:500: user:/home/ user:/bin/bash
amit:x:501:501::/home/amit:/bin/bash
vivek:x:502:502::/home/vivek:/bin/bash
musab:x:503:503::/home/musab:/bin/bash
rahul:x:504:504::/home/rahul:/bin/bash
```

### Sort:

It is used to sort the output in numeric or alphabetic order

#sort filename

```
[root@musab1 ~]# cat test
Linux is freedom
Linux is freedom
Welcome to my world
Welcome to my world
Hello world
Hello world
[root@musab1 ~]# sort test
Hello world
Hello world
Linux is freedom
Linux is freedom
Welcome to my world
Welcome to my world
[root@musab1 ~]#
```

**To sort the file according to numbers**

#sort -d test or #sort -h test

```
[root@musab1 ~]# cat test
6. Linux is freedom
3. Linux is freedom
1. Welcome to my world
2. Welcome to my world
4. Hello world
7. Hello world
[root@musab1 ~]# sort -d test
1. Welcome to my world
2. Welcome to my world
3. Linux is freedom
4. Hello world
6. Linux is freedom
7. Hello world
```

**To remove the duplicate entries from the output**

#sort -u test

```
[root@musab1 ~]# cat test
Linux is freedom
Linux is freedom
Welcome to my world
Welcome to my world
Hello world
Hello world
[root@musab1 ~]# sort -u test
Hello world
Linux is freedom
Welcome to my world
```



**cut command:**

The cut command is used to pick the given expression (in columns) and display the output.

# cut -d -f filename (where d stands for delimiter ex. :, " " etc and f stands for field)

```
[root@ linux ~]# cut -d: -f1 /etc/passwd
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
uucp
```

**To delimit spaces and print the field**

#cut -d " " -f1 filename

To delimit commas and print the field

**#cut -d, -f1 filename**

```
[root@linux ~]# cat hello
hello,how,are,you
[root@linux ~]# cut -d, -f1 hello
hello
```

### **sed command:**

**sed** stands for **stream editor**, which is used to search a word in the file and replace it with the word required to be in the output

**Note:** it will only modify the output, but there will be no change in the original file.

**#sed 's/searchfor/replacewith/g' filename**

```
[root@musab1 ~]# cat test
Linux is freedom
Linux is freedom
Welcome to my world
Welcome to my world
Hello world
Hello world
[root@musab1 ~]# sed 's/Linux/LINUX/g' test
LINUX is freedom
LINUX is freedom
Welcome to my world
Welcome to my world
Hello world
Hello world
```

### **I/O Redirection:**

Redirection is a process where we can copy the output of any command(s), file(s) into a new file. There are two ways of redirecting the output into a file.

Using **>** or **>> filename** after the command, and

Using **tee** command

**Let's see the > and >> option first**

Syn: command > new file

**Note:** if the given name of the file is not available a new file will be created automatically. If the file already exists then it will overwrite contents of that file.

```
[root@musab1 ~]# cat test
Linux is freedom
[root@musab1 ~]# sed 's/Linux/LINUX/g' test > test2
[root@musab1 ~]# cat test2
LINUX is freedom
```

### Appending another output in same the same file

```
[root@musab1 ~]# cat file2
Welcome to MyWorld
[root@musab1 ~]# cat file2 >>test2
[root@musab1 ~]# cat test2
LINUX is freedom
Welcome to MyWorld
```

Likewise there are many options where we can use redirections

Ex:

**Copying contents of two files in a new file**

```
#cat file1 file2 > file3
```

### Using tee command:

The above options of redirections will not display any output, but directly save the output in a file. Using tee command will not only redirect the output to new file but it will also display the output.

Syn: cat <filename> | tee <new file name>

Note: if the given name of the file (newfile) is not available a new file will be created automatically. If the file already exists then it will overwrite contents of the file.

```
#cat file2 |tee file3
```

```
[root@musab1 ~]# cat file2 |tee file3
Welcome to MyWorld
[root@musab1 ~]# cat file3
Welcome to MyWorld
```

Appending data in the same file using tee command

Syn: cat filename |tee -a filename2

```
#cat test | tee -a file2
```

```
[root@musab1 ~]# cat test |tee -a file2
Linux is freedom
[root@musab1 ~]# cat file2
Welcome to MyWorld
Linux is freedom
```

## Find command:

**find** command is used to find the files or directory's path, it is exactly like the find option in windows where you can search for a file.

**Syntax: find / (under root) –option filename**

**Options that can be used with find command:**

Option	Usage
<b>-name</b>	For searching a file with its name
<b>-inum</b>	For searching a file with particular inode number
<b>-type</b>	For searching a particular type of file
<b>-user</b>	For files whose owner is a particular user
<b>-group</b>	For files belonging to particular group

### Finding a File with name

#find / -name newfile

```
[root@musab1 ~]# find / -name newfile
/root/newfile
```

### Finding a file with its inode number

#find / -inum 1445159

```
[root@musab1 ~]# find / -inum 1445159
find: `/proc/28735/task/28735/fd/5': No such file or directory
find: `/proc/28735/task/28735/fdinfo/5': No such file or directory
find: `/proc/28735/fd/5': No such file or directory
find: `/proc/28735/fdinfo/5': No such file or directory
/root/hardlink
/root/newfile
[root@musab1 ~]#
```

### Finding the files, whose owner is a user called "ktuser"

#find / -user myuser

```
[root@musab1 ~]# find / -user myuser
find: `/proc/28763/task/28763/fd/5': No such file or directory
find: `/proc/28763/task/28763/fdinfo/5': No such file or directory
find: `/proc/28763/fd/5': No such file or directory
find: `/proc/28763/fdinfo/5': No such file or directory
/var/spool/mail/myuser
/home/myuser
/home/myuser/.bash_profile
/home/myuser/.gnome2
/home/myuser/.bash_logout
/home/myuser/testfile
```

Finding the files whose group is “myuser”

#find / -group myuser

```
[root@musab1 ~]# find / -group myuser
find: `/proc/28780/task/28780/fd/5': No such file or directory
find: `/proc/28780/task/28780/fdinfo/5': No such file or directory
find: `/proc/28780/fd/5': No such file or directory
find: `/proc/28780/fdinfo/5': No such file or directory
/home/myuser
/home/myuser/.bash_profile
/home/myuser/.gnome2
/home/myuser/.bash_logout
/home/myuser/testfile
```

## File Permissions:

Permissions are applied on three levels:

- Owner or User level
- Group level
- Others level

Access modes are of three types:

- r read only
- w write/edit/delete/append
- x execute/run a command

Access modes are different on file and directory:

Permissions	Files	Directory
r	Open the file	'ls'/list the contents of directory
w	Write, edit, append, delete file	Add/Del/Rename contents of directory
x	To run a command/shell script	To enter into directory using 'cd'

```
[root@musab1 ~]# ls -l myfile
-rw-r--r--. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]# ls -ld mydir
drwxr-xr-x. 2 root root 4096 Feb 13 16:43 mydir
```

**Filetype+permission, links, owner, group name of owner, size in bytes, date of modification, file name**

Permission can be set on any file/dir by two methods:

1 Symbolic method (ugo)

2 Absolute methods (numbers)

### 1 Symbolic method (ugo):

- Symbolic mode: General form of symbolic mode is:  
**# chmod [who] [+/-/=] [permissions] file**  
 who → To whom the permissions to be assigned  
 User/owner (u); group (g); others (o)

#### **Example:**

**Assigning different permissions to the file (user=rwx, group=rw and others=r)**

#chmod u=rwx,g=rw,o=r myfile (where myfile is the name of the file)

```
[root@musab1 ~]# chmod u=rwx,g=rw,o=r myfile
[root@musab1 ~]# ls -l myfile
-rwxrw-r--. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]#
```

**Assigning full permission to the file i.e. rwx to all**

#chmod ugo=rwx <file name>

```
[root@musab1 ~]# chmod ugo=rwx myfile
[root@musab1 ~]# ls -l myfile
-rwxrwxrwx. 2 root root 0 Feb 13 13:55 myfile
```

Likewise you can add or remove permissions from any file for anyone (user group or other)

- #chmod u+x myfile (Adding execute permission to user only)
- #chmod go-wx myfile (Removing write and execute permissions from group and other)
- #chmod go+wx myfile (Adding write and execute permissions from group and other)
- #chmod go=r myfile (Giving only read permission to group and other)

### **2 Absolute Method (numbers)**

In Absolute method we use numbers instead of using symbols i.e.

- Read=4
- Write=2
- Execute=1

**Assigning different permissions to the file (user=rwx, group=rw and others=r)**

#chmod 764 myfile (where 7 means rwx i.e. 4+2+1, rw=6 i.e. 4+2 and 1 indicates x)

```
[root@musab1 ~]# ls -l myfile
-rwxrwxrwx. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]# chmod 764 myfile
[root@musab1 ~]# ls -l myfile
-rwxrw-r--. 2 root root 0 Feb 13 13:55 myfile
```

Assigning full permission to the file i.e. rwx to all

#chmod 777 myfile

```
[root@musab1 ~]# ls -l myfile
-rwxrw-r--. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]# chmod 777 myfile
[root@musab1 ~]# ls -l myfile
-rwxrwxrwx. 2 root root 0 Feb 13 13:55 myfile
```

Likewise you can give different permissions according to your requirement

**Removing all permissions from others**

#chmod 770 myfile (where 0 indicates no permissions)

**Note:** All the above permissions and procedure is same for files and directories.

#### Umask:

When we create any file using touch, cat or vi commands they get created with default file permissions as stored in umask (**User file creation mask**).umask is a 4 digit octal number which tells Unix which of the three permissions are to be denied rather than granted. Umask will decide that what should be the default permissions for a file and directory when it is created.

The default umask value is 0022

#umask

```
[root@musab1 ~]# umask
0022
```

Calculation of default permissions for file and directory, basing upon the umask value

**Note:** For a file by default it cannot have the execute permission, so the maximum full permission for a file at the time of creation can be **666** (i.e. 777 -111 = 666), whereas a directory can have full permissions i.e. **777**

- The full permission for the file 666
- Minus the umask value - 022
- The default permission for file is 644 (rw-,r--,r--)

```
[root@musab1 ~]# umask
0022
[root@musab1 ~]# touch test
[root@musab1 ~]# ls -l test
-rw-r--r--. 1 root root 0 Feb 13 16:55 test
```

- The full permission for the directory 777
- Minus the umask value - 022
- The default permission for file is 755 (rwx, r-x, r-x)

```
[root@musab1 ~]# umask
0022
[root@musab1 ~]# mkdir testdir
[root@musab1 ~]# ls -ld testdir
drwxr-xr-x. 2 root root 4096 Feb 13 16:56 testdir
```

### Modifying the umask value:

#umask 002

The Modified default Permission for a file will be **666-002=664** i.e. **rw,rw,r**, and for the directory it will be **777-002=775** i.e. **rwx,rwx,r-x**.

```
[root@musab1 ~]# umask  
0022  
[root@musab1 ~]# umask 002  
[root@musab1 ~]# umask  
0002
```

Note: Create a file and a directory and check for the default permissions.

Visit [www.musab.in](http://www.musab.in) for all basics video tutorial by Musab Syed

These were the few things amongst the basics; keep working to furnish your basics. After All, "*if the foundation is good then only the building can stand still*"



## RHEL 8 BASIC INSTALLATION

### Minimum and Recommended Requirements to install RHEL7/8

MINIMUM	RECOMMENDED
Intel/AMD Dual core processor	Intel/AMD Quad Core Processor
4GB RAM	16GB RAM
25GB HARD DISK SPACE	50GB HARD DISK SPACE

Note: RHEL7/8 comes only in 64 bit version

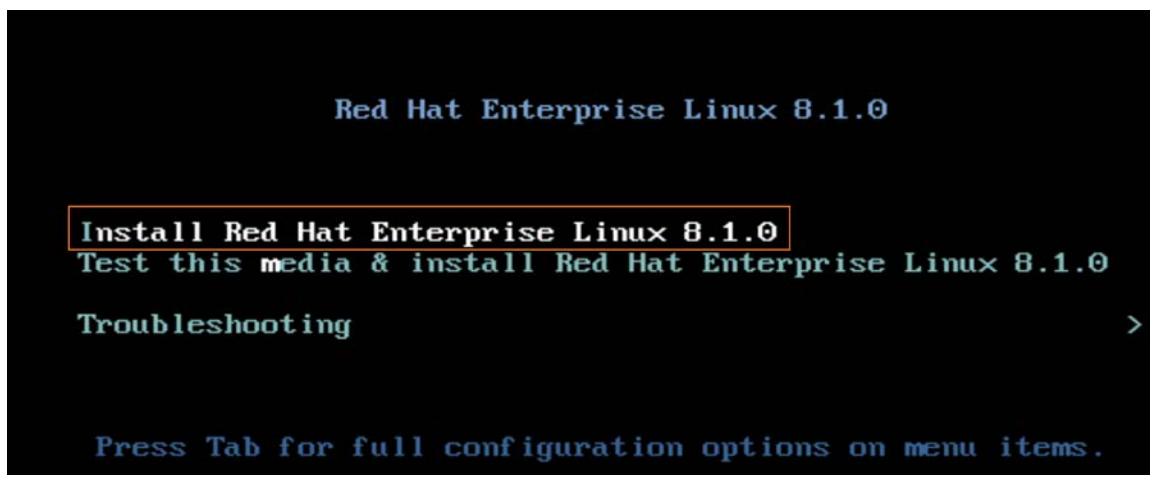
### Minimum Partition creation and sizes for basic installation

Partitions	Sizes
/ {root}	20-30 GB APPROX
/boot	1 GB
SWAP	Twice of RAM approx

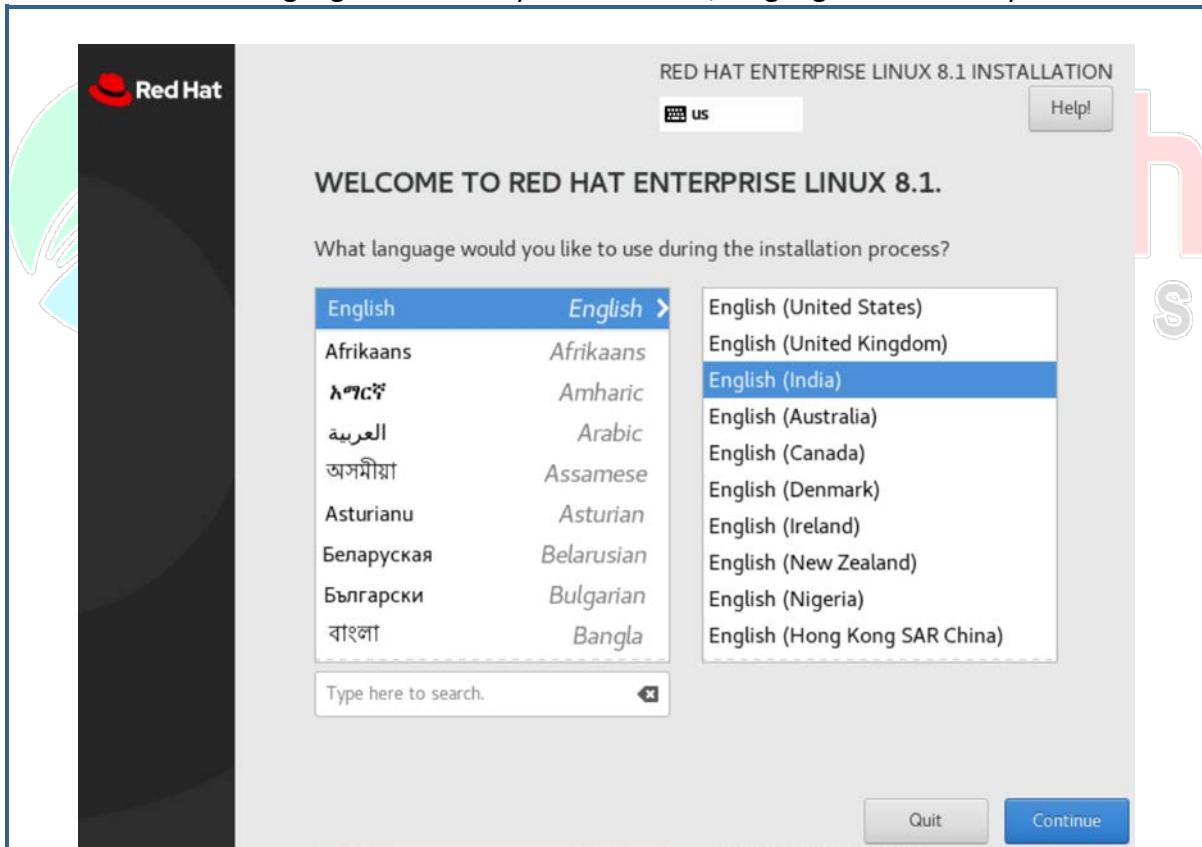
### Installing RHEL7/8 with above specification

- Enter into BIOS setting and make CD/DVD Drive as first boot device
- Make sure that VT {Virtual Technology} is enabled in BIOS/UEFI in 64 bit systems
- Insert the RHEL 7/8 CD/DVD into CD/DVD drive and boot the system
- If booted from CD/DVD Rom the following screen will be displayed

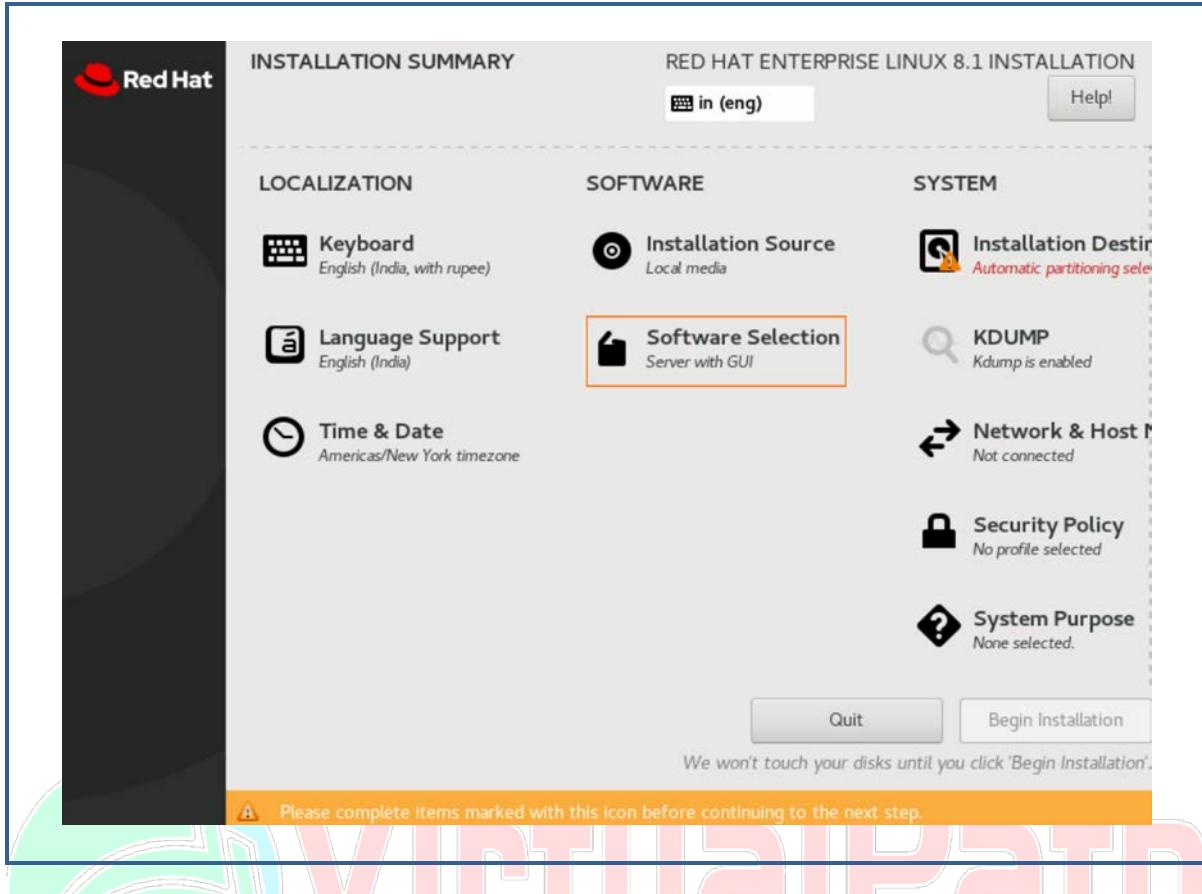
- Move the cursor to **install Red Hat Enterprise linux 8.x**, hit Enter



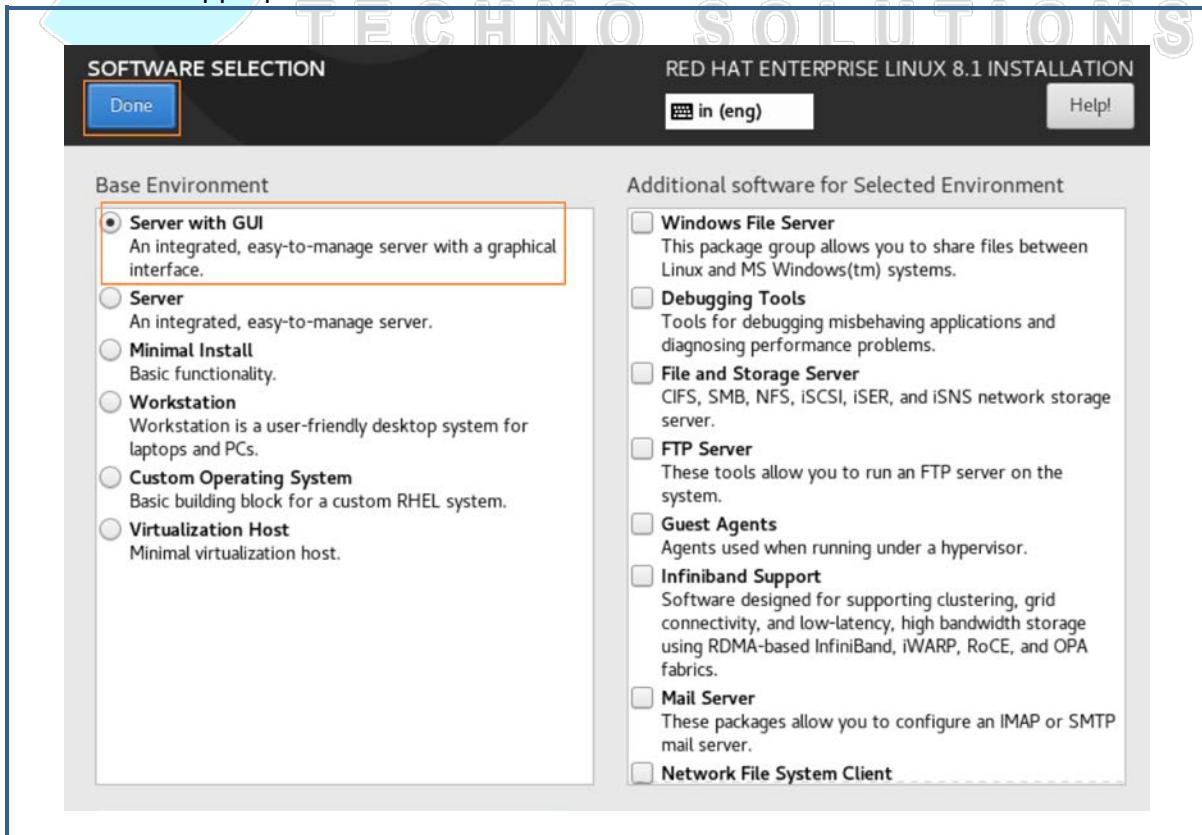
- Select the Language and Country for time zone, language and currency



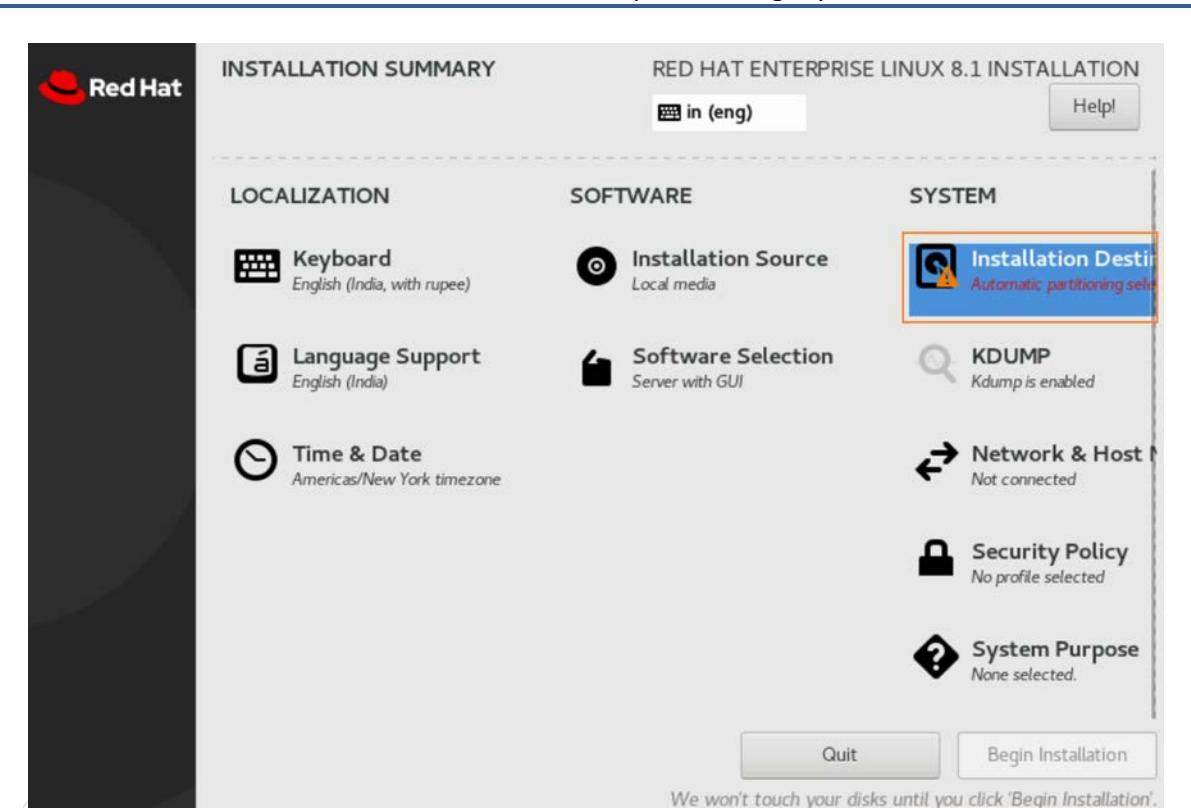
- Select **Software Selection** for selecting software to install during installation



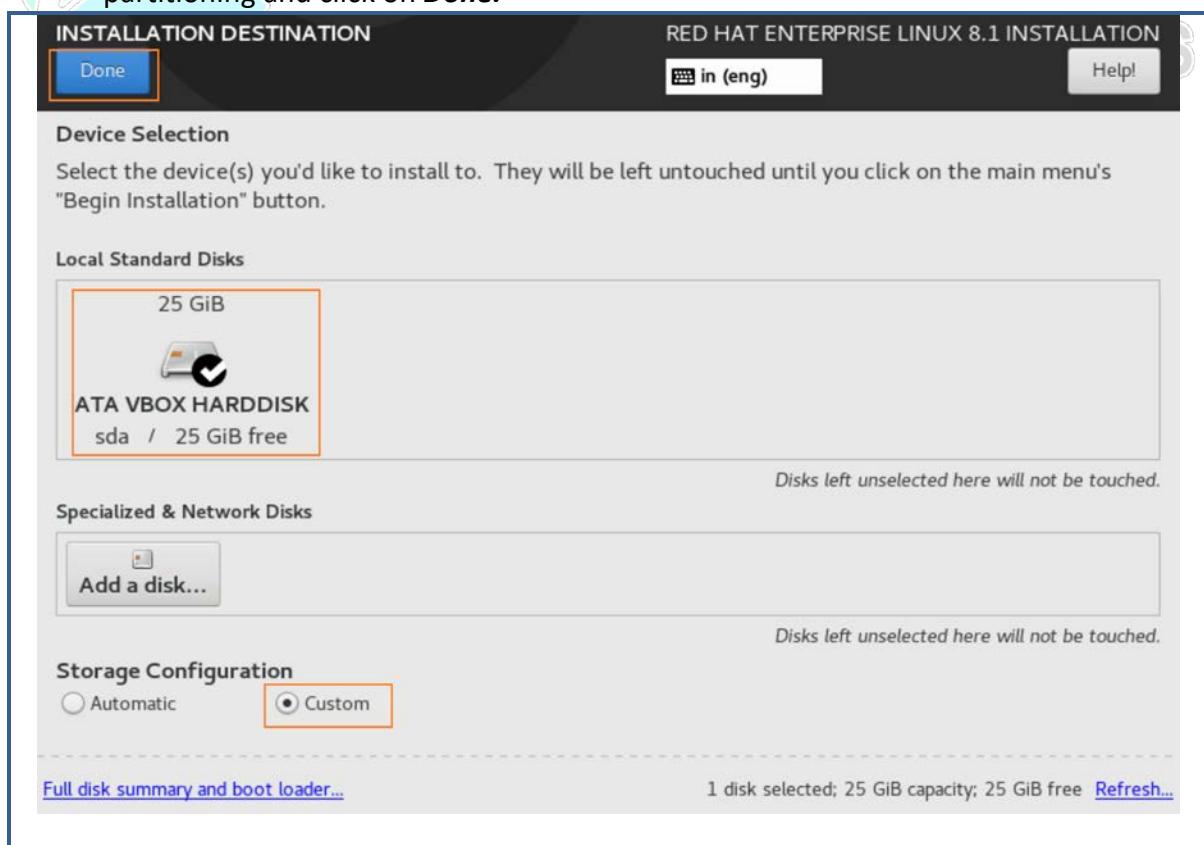
- Select appropriate environment for installation and click on **done**



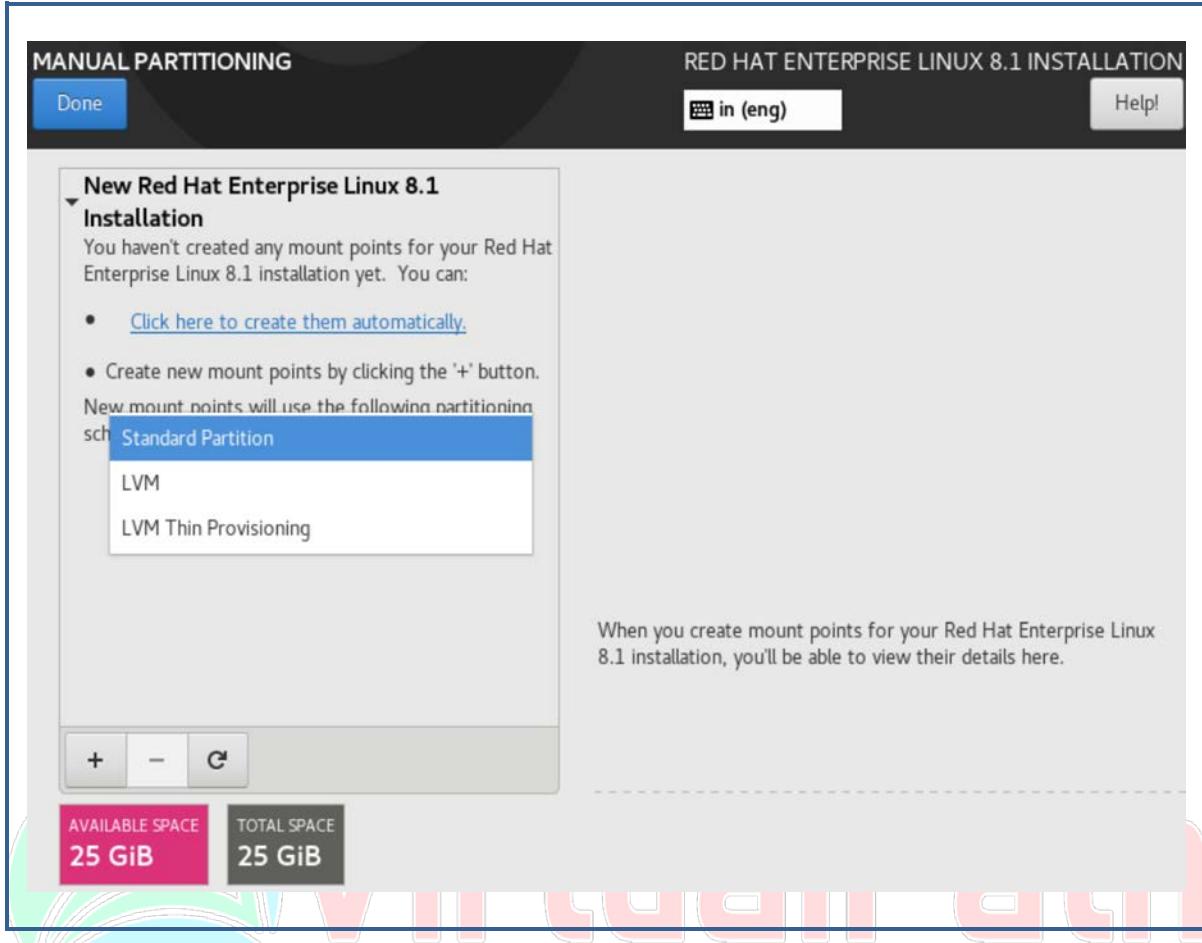
- Click on **INSTALLATION DESTINATION** for partitioning layout



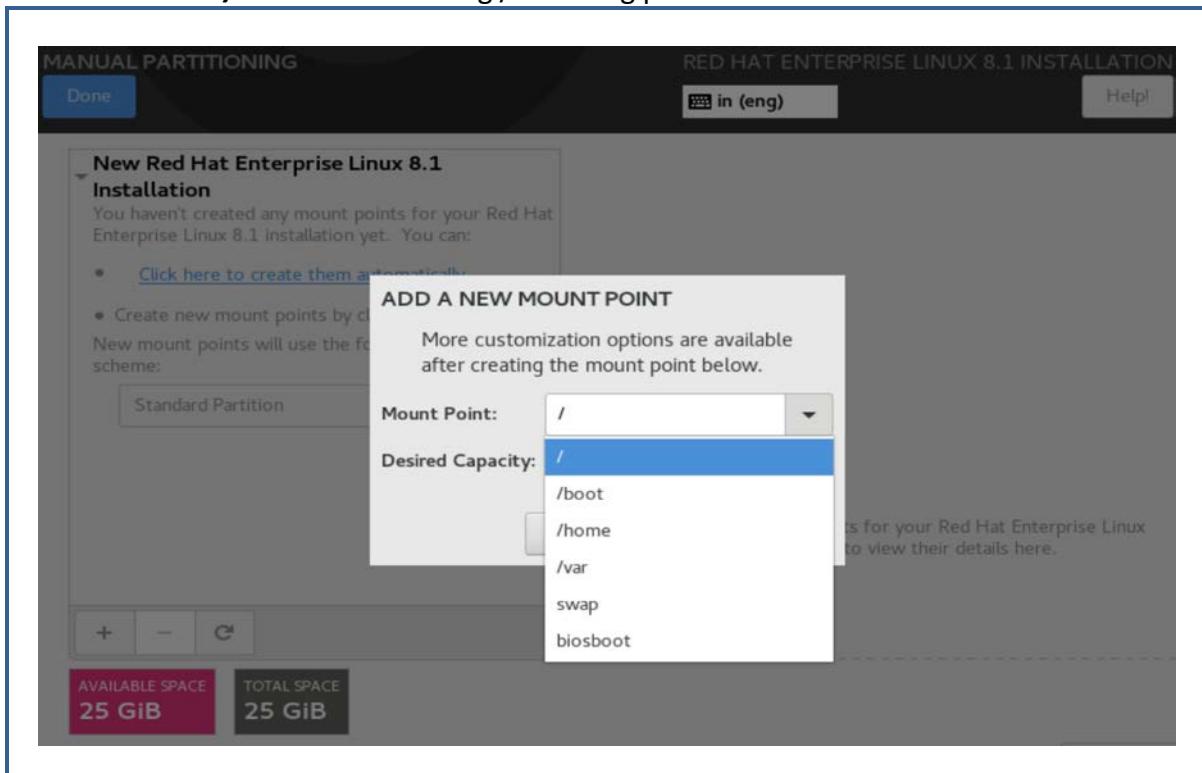
- Select the Disk which you want to use for installation, Select **Custom**, for manual partitioning and click on **Done**.



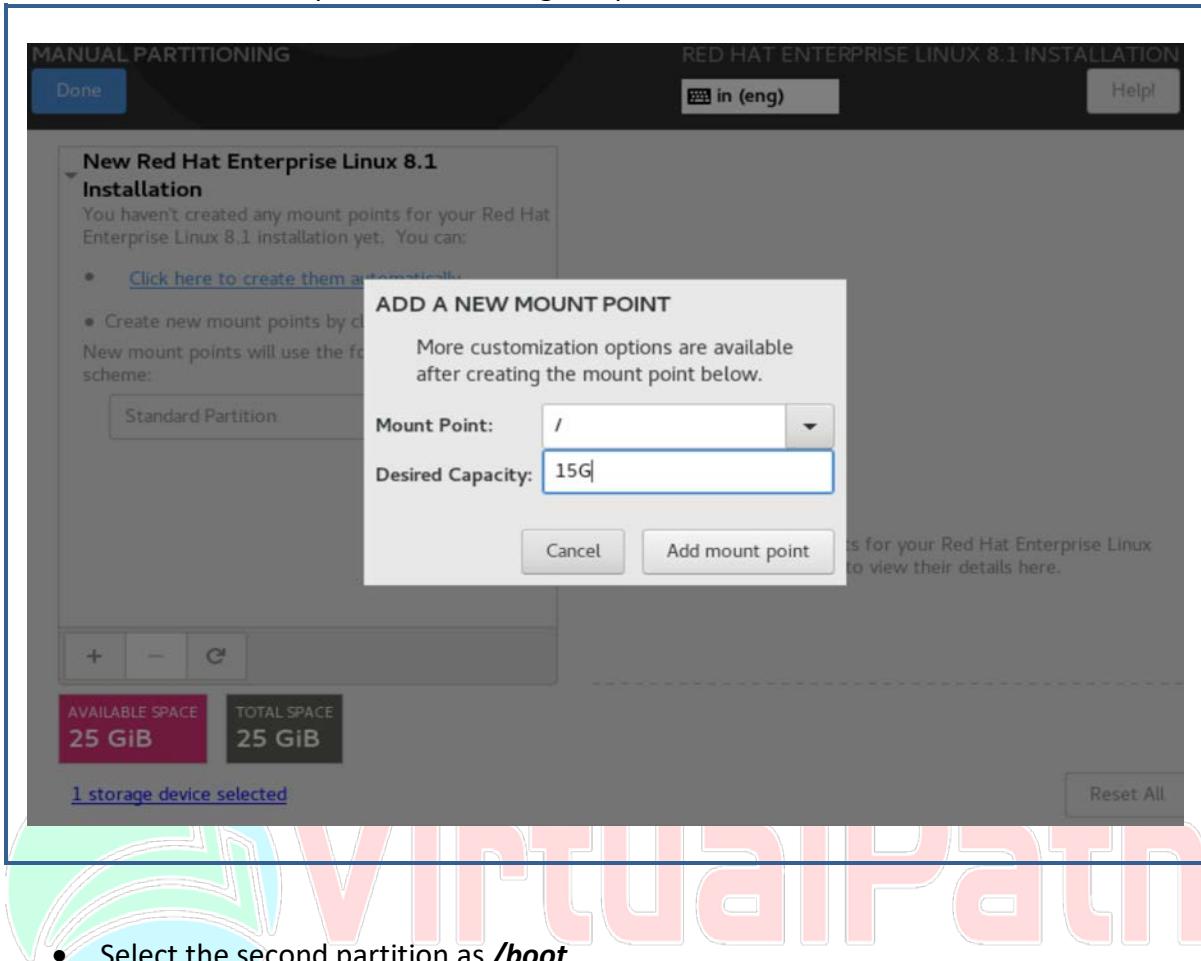
- Select the style of partitioning type preferable



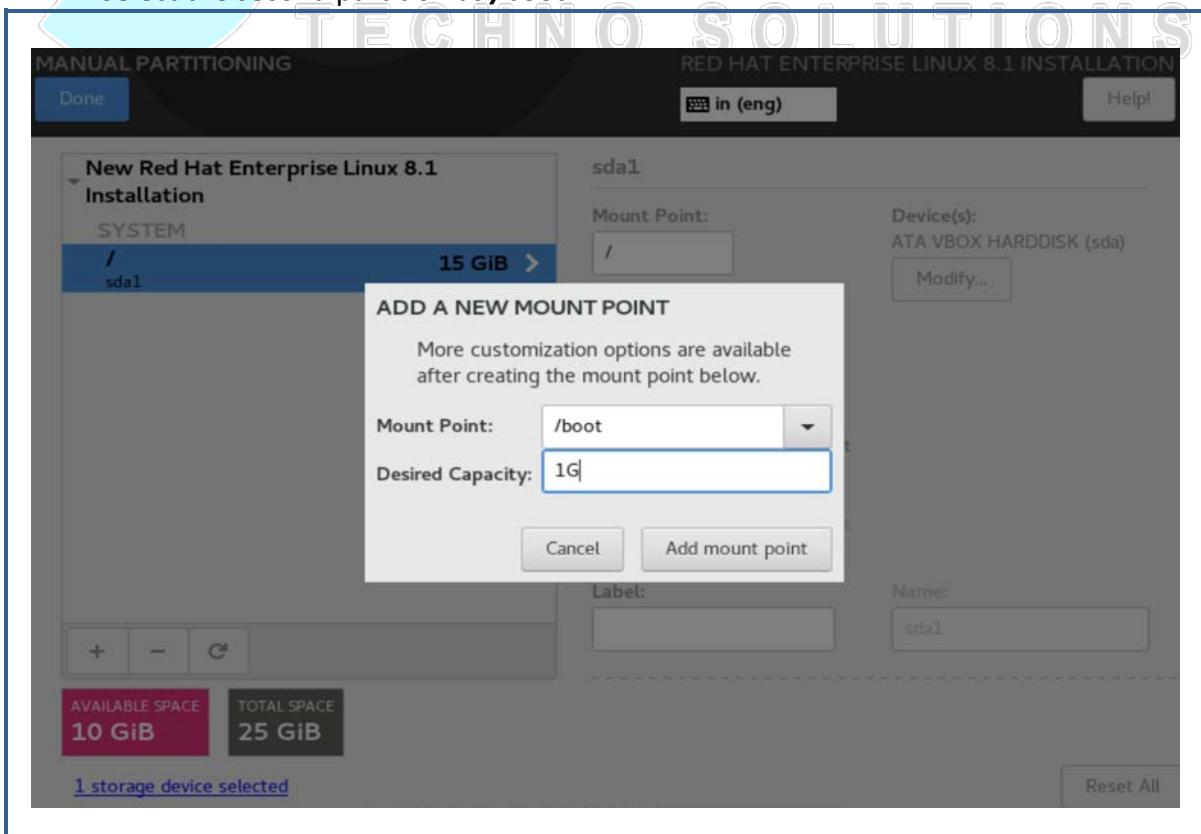
- Click on +/- button for adding /removing partitions



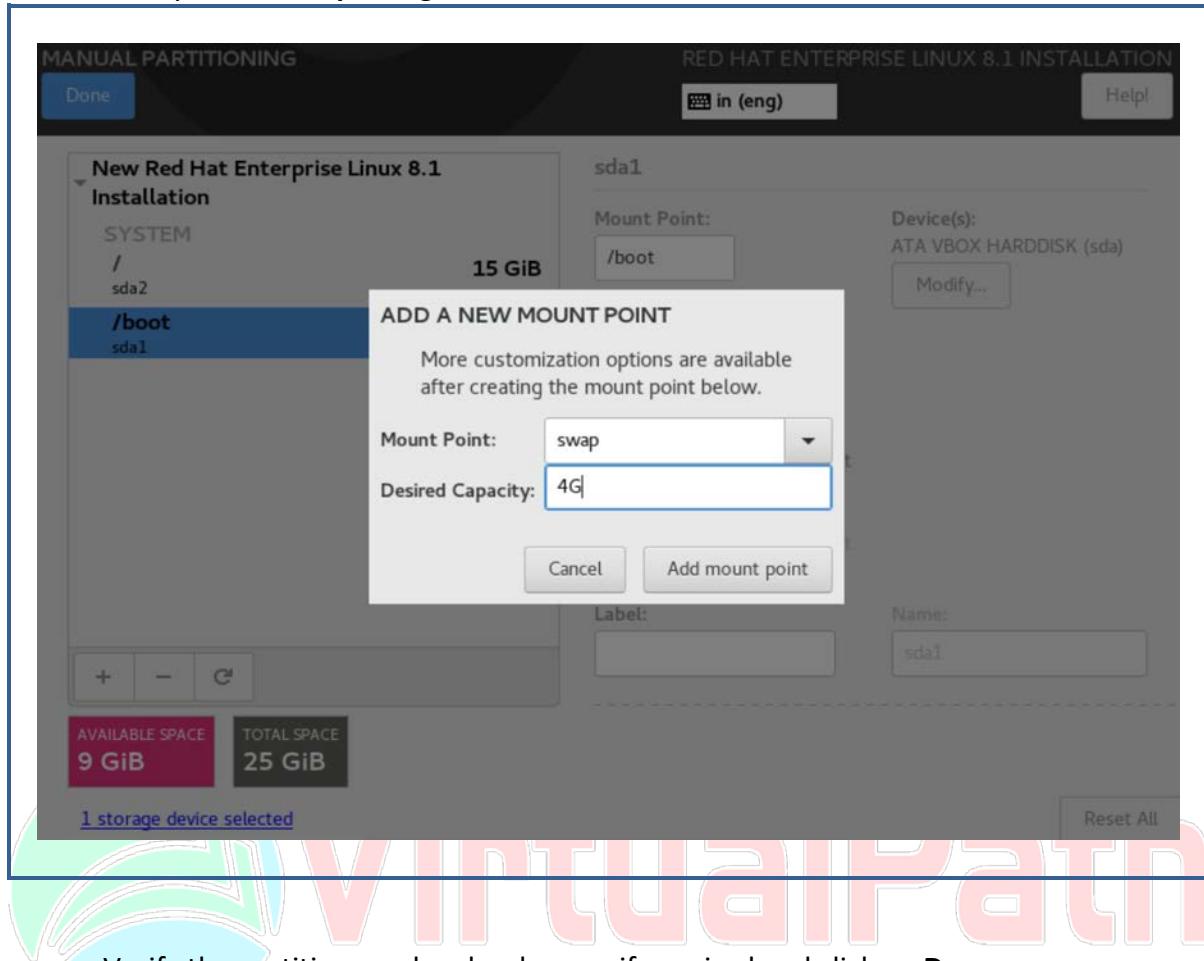
- Select **/** for root partition and assign required size in KB, MB, GB



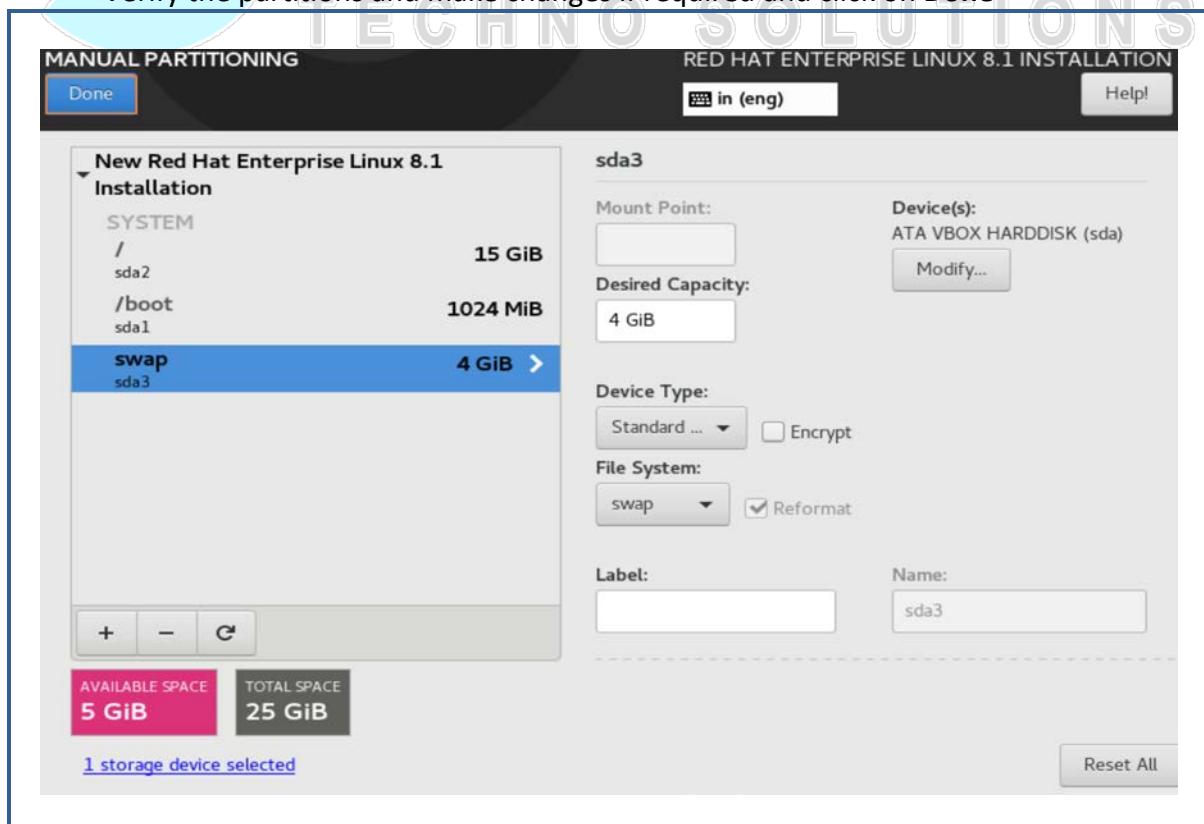
- Select the second partition as **/boot**



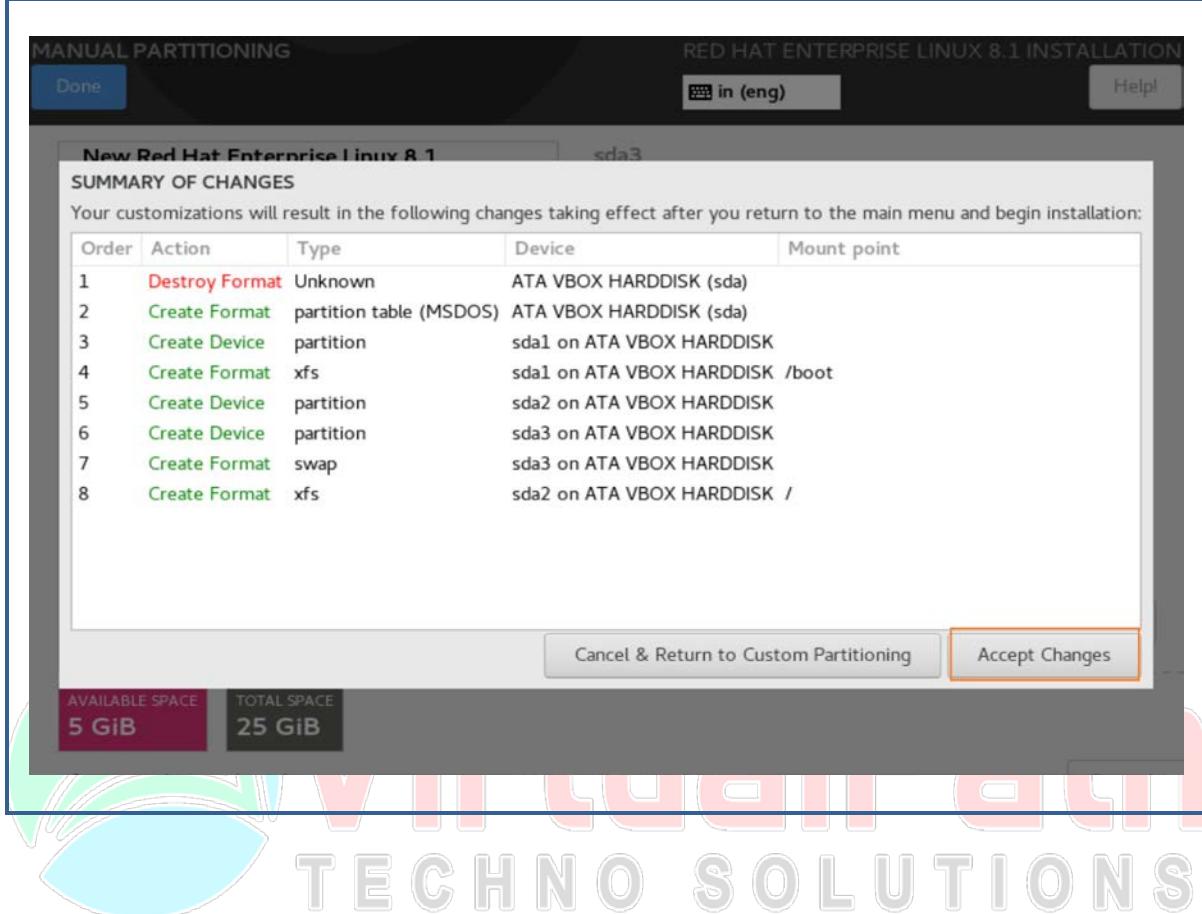
- Finally, select **swap** and give the size



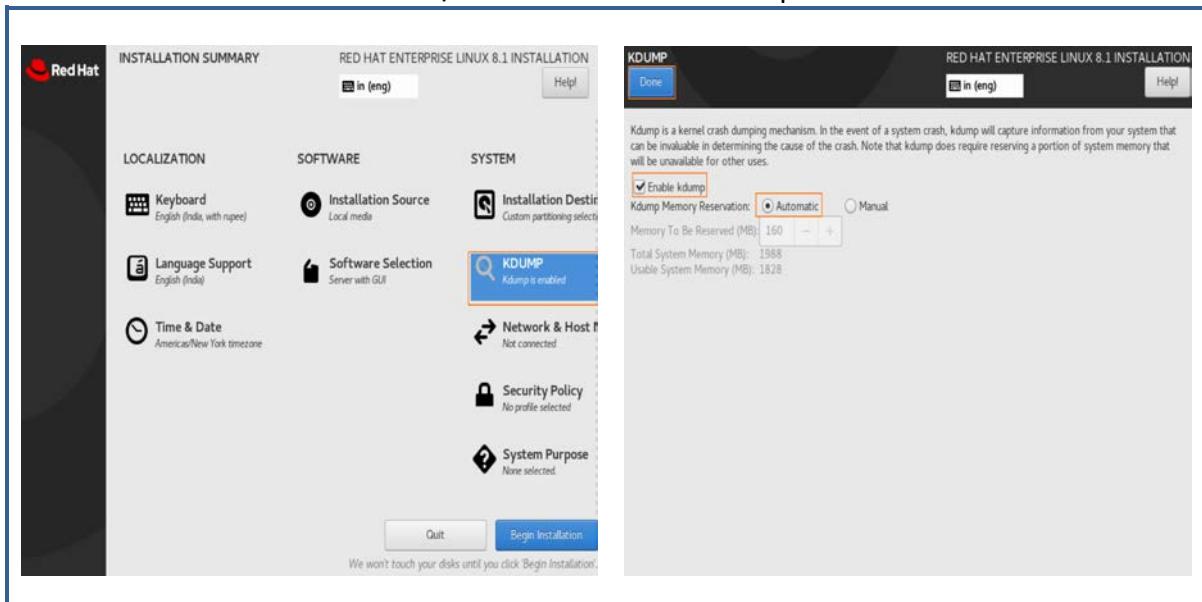
- Verify the partitions and make changes if required and click on **Done**



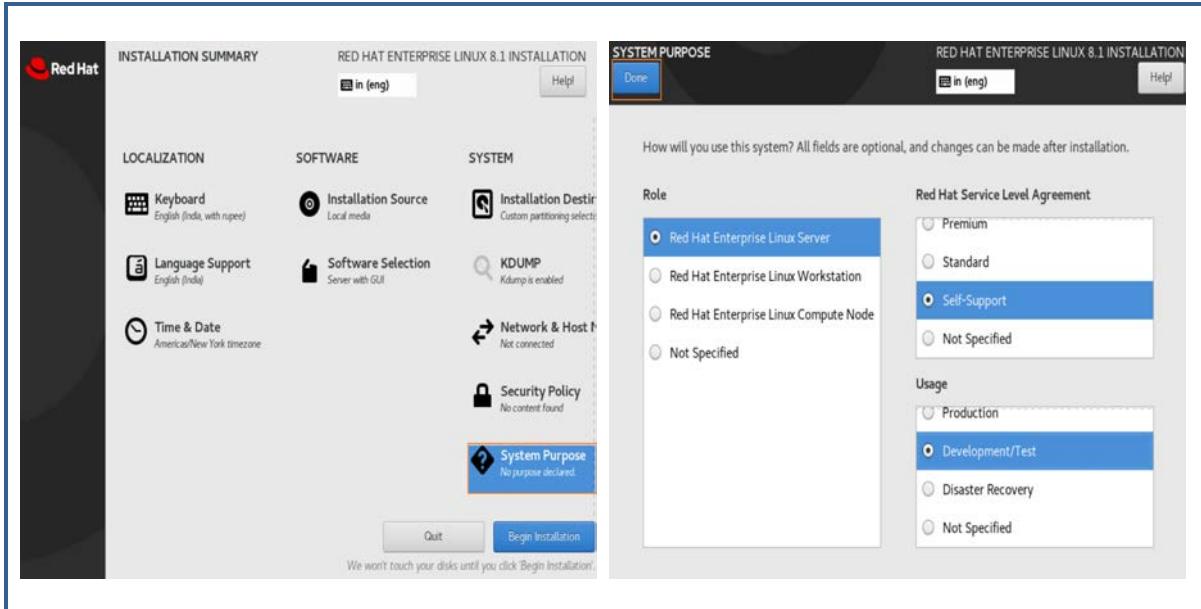
- Accept the changes for continue



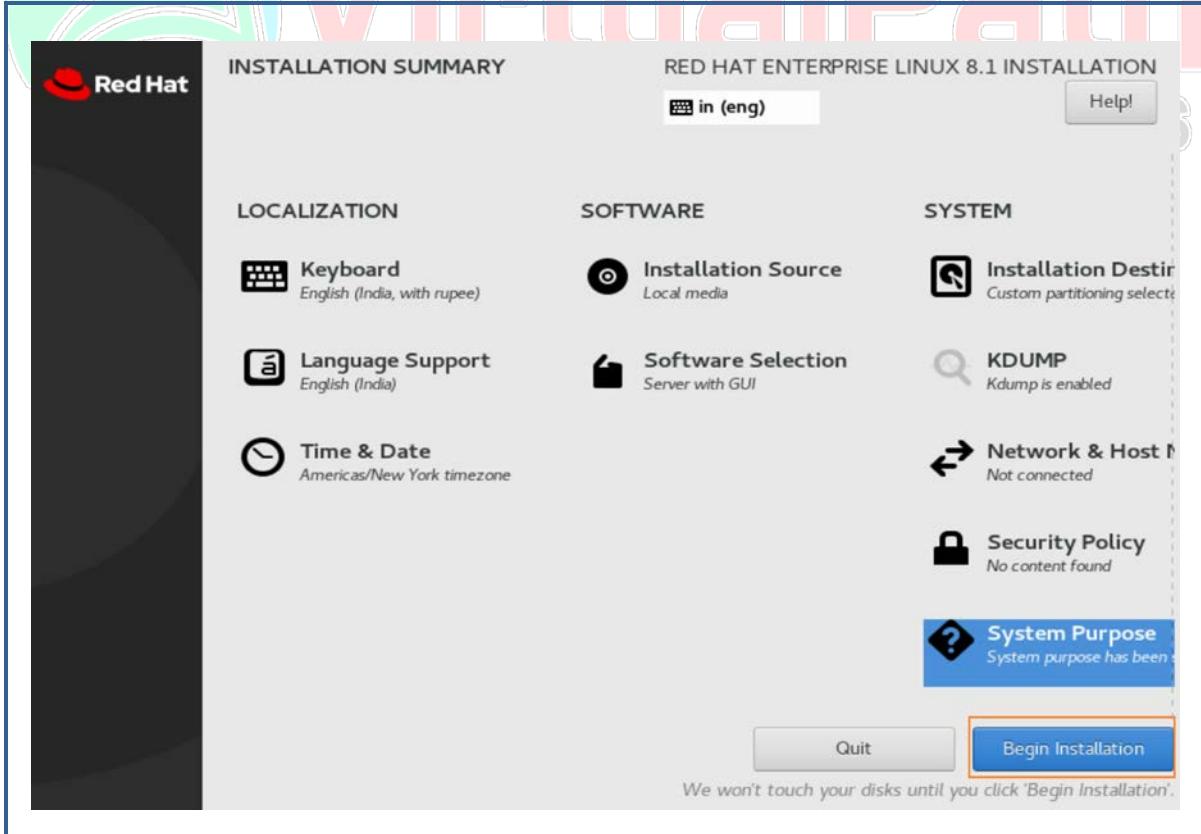
- Click on **KDUMP** to enable/disable kernel crash dump mechanism.



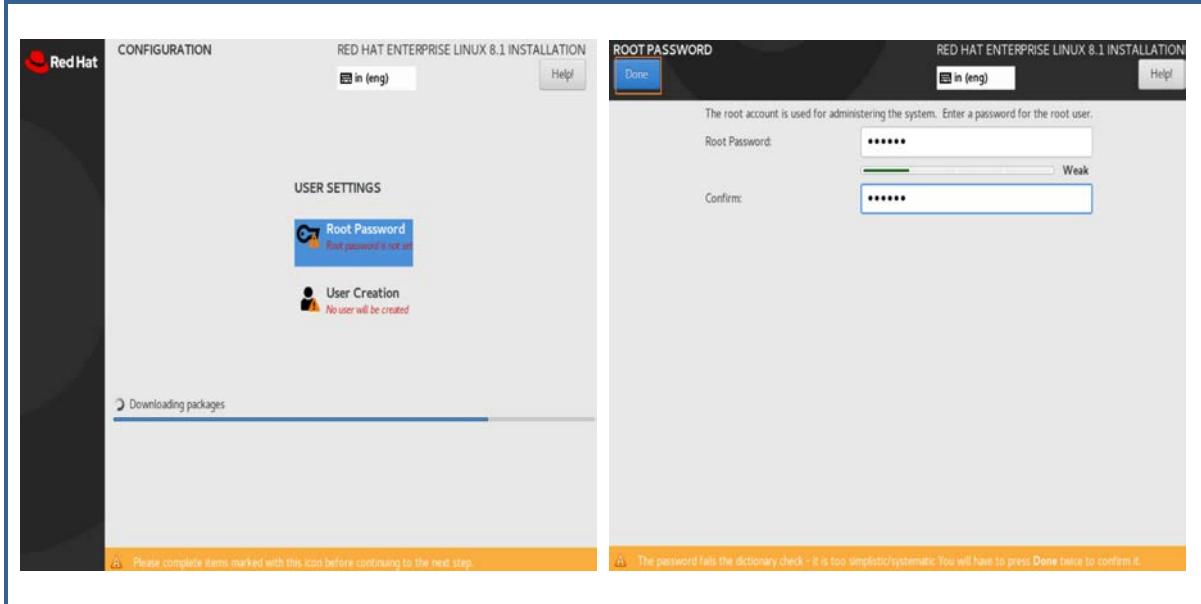
- Click on **System Purpose** to set usage, which is optional and can be skipped.



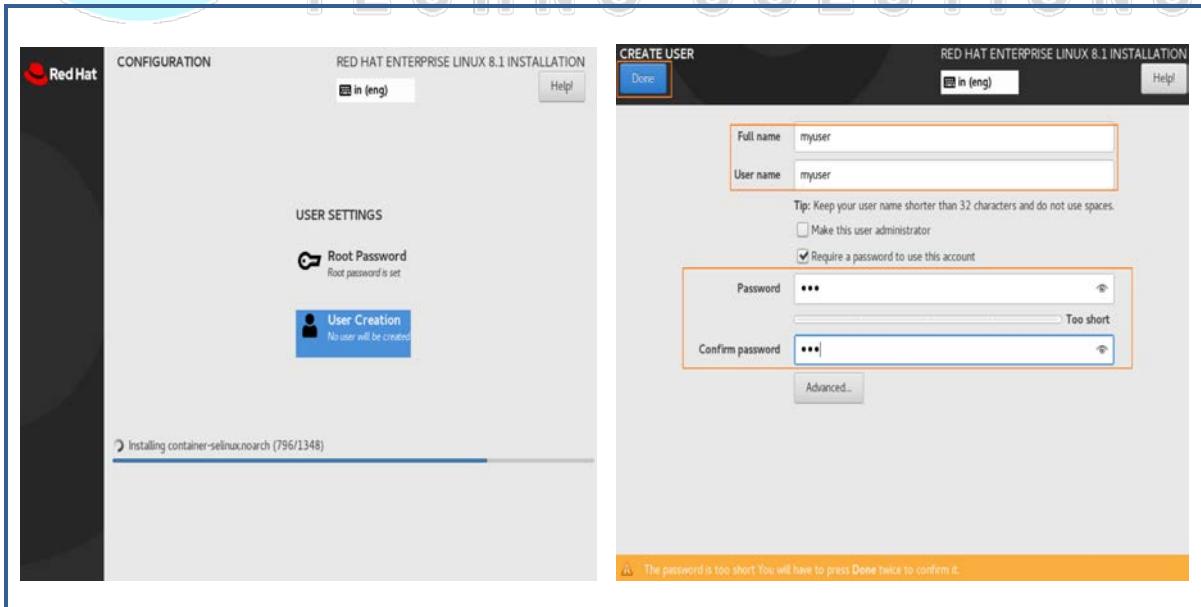
- Click on **Begin Installation** to start the installation



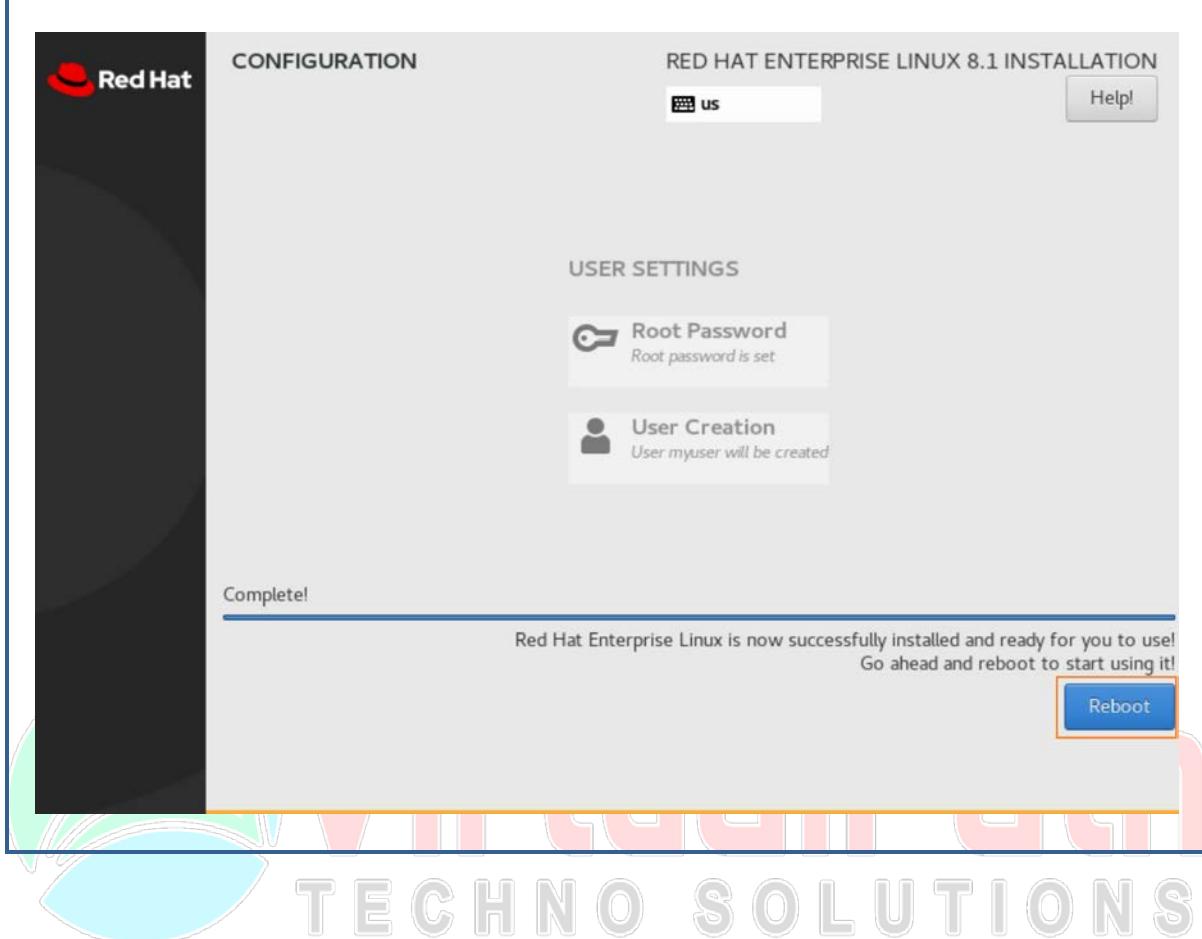
- While the installation is in progress, root password can be set. Click on **ROOT PASSWORD**. Type the password twice and click on done.



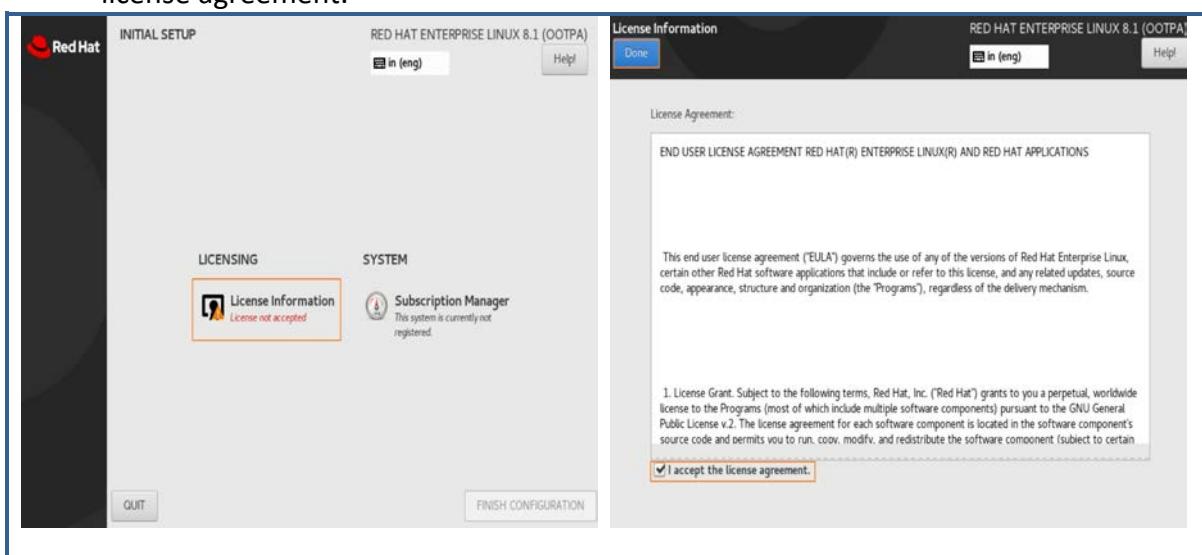
- Optionally you can add a normal user. Click on **USER CREATION**, type the password twice and click on **Done**.



- Once the installation is completed, reboot the machine while clicking on **Reboot**

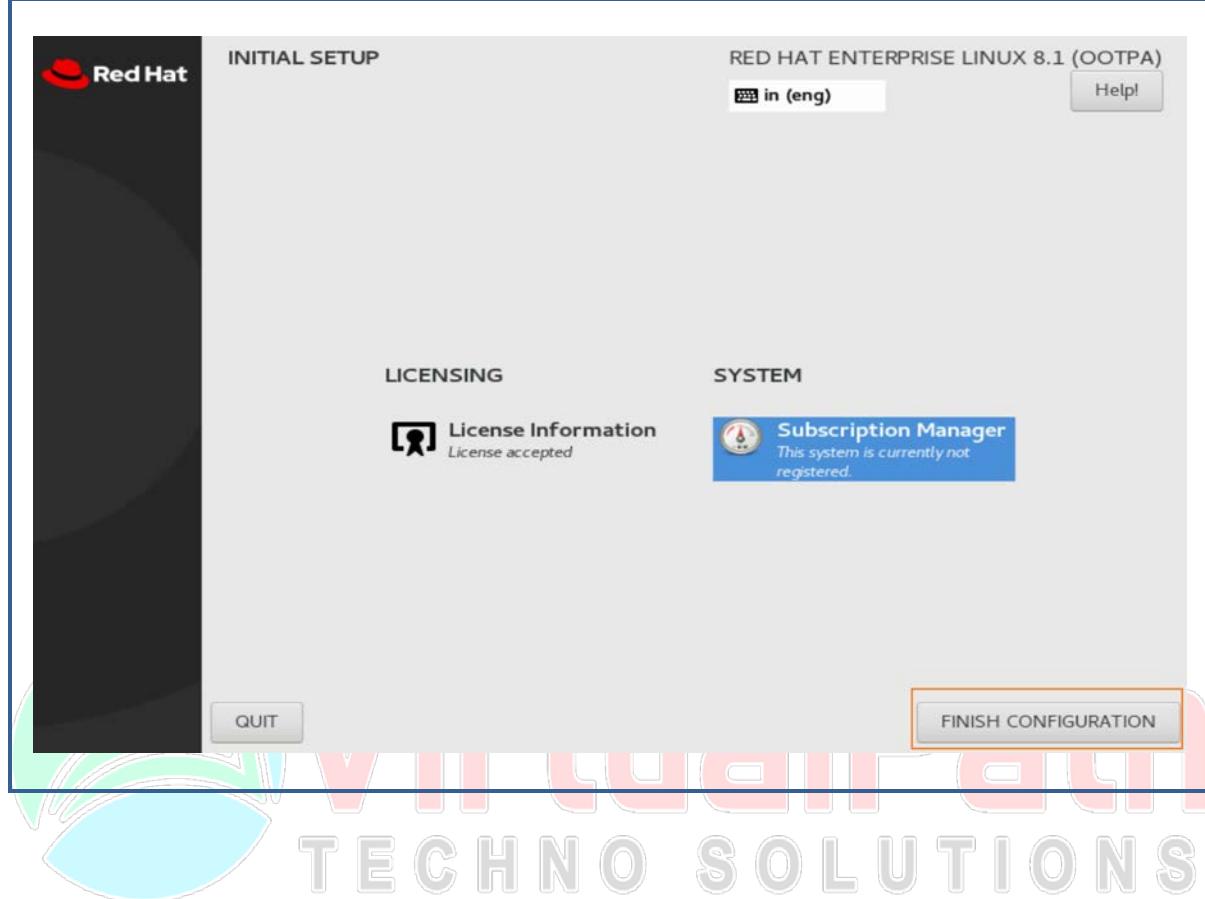


- After booting first time, click on **LICENSE INFORMATION** to see and accept the license agreement.



- Read and Accept the EUL Agreement and click on done

- If you have subscription, register your machine with **Redhat Network** for updates and support.
- If you want to skip registration, click on **FINISH CONFIGURATION**



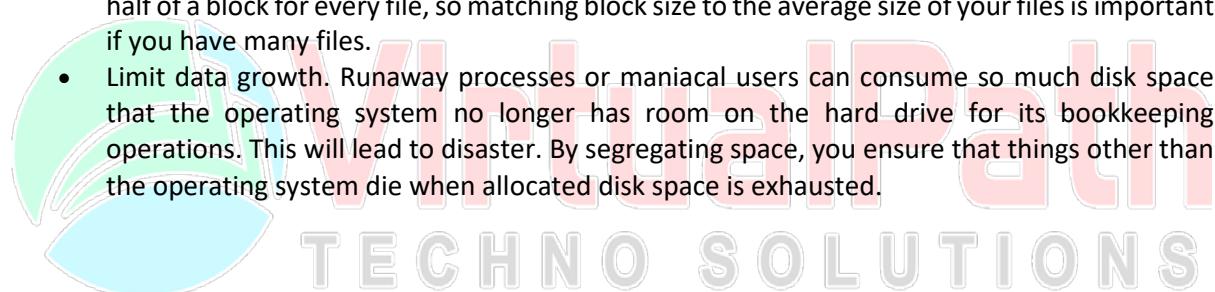
## MANAGING PARTITIONS & FILE SYSTEMS

### What is a partition?

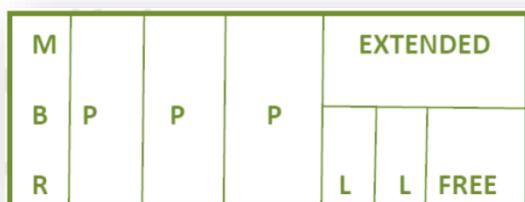
Partitioning is a means to divide a single hard drive into many logical drives. A partition is a contiguous set of blocks on a drive that are treated as an independent disk. A partition table is an index that relates sections of the hard drive to partitions.

### Why have multiple partitions?

- Encapsulate your data. Since file system corruption is local to a partition, you stand to lose only some of your data if an accident occurs.
- Increase disk space efficiency. You can format partitions with varying block sizes, depending on your usage. If your data is in a large number of small files (less than 1k) and your partition uses 4k sized blocks, you are wasting 3k for every file. In general, you waste on average one half of a block for every file, so matching block size to the average size of your files is important if you have many files.
- Limit data growth. Runaway processes or maniacal users can consume so much disk space that the operating system no longer has room on the hard drive for its bookkeeping operations. This will lead to disaster. By segregating space, you ensure that things other than the operating system die when allocated disk space is exhausted.



### Disk Partitioning Criteria:



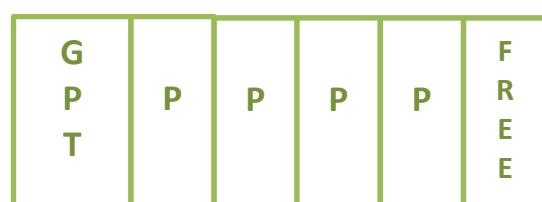
**MBR = MASTER BOOT RECORD**

**P= PRIMARY PARTITION**

**EXTENDED= EXTENDED PARTITION**

**L= LOGICAL PARTITION**

**FREE= FREE SPACE**



**GPT = GUID PARTITION TABLE**

**P= PARTITION**

**FREE= FREE SPACE**

## The Structure of Disk Partition

- On the disk where O/S is installed, will have the first partition as **MBR/GPT**.
- **MBR** is a Master Boot Record, which contains two important utilities, **IPL** (Initial Program Loader) and **PTI** (Partition Table information). Which supports up to 2TB of disk size
- **GPT** is **GUID Partition Table**, which also contains **IPL** as well as **PTI**, supports up to **2 Zettabyte (1024 EB (Exabyte)=1 ZB (zettabyte))**.
- **IPL** is responsible for booting the operating system, because it contains the **boot loader**.
- In earlier versions of Linux i.e. up to **RHEL 4**, the default boot loader was **LILO** (Linux Loader). But, since **RHEL5** onwards it has been changed to **GRub** (Grand Unified Boot loader), which is far more superior to **LILO**. In **RHEL 7 GRub2** has been introduced.
- The **PTI** (Partition Table information) is the information about the number of partitions on the disk, sizes of the partition and types of partitions.

### THE CRITERIA OF DISK PARTITIONING IN MBR:

- Every disk can have only **4 Primary Partitions** (3 Primary + 1 Extended).
- **Primary Partition** is a partition which usually holds the **operating system**.
- **Extended Partition** is a special type of primary partition which can be subdivided into multiple logical partitions. As there can be only 3 primary partitions per disk, and if the user is required to make further partitions then all the space remaining on the disk should be allocated to extended partition, which can be used to create the logical partitions later. There can be only **one extended partition** per disk.
- With the help of extended partition, you can create up to **60 partitions**, but redhat says you can create max up to **15-25 max** no. of partitions in rhel7 and 8.
- **Logical partitions** are the partitions which are created under extended partition, all the space in the extended partition can be used to create any number of logical partitions.

### THE CRITERIA OF DISK PARTITIONING IN GPT:

- It supports up to 128 partitions per disk

### Disk Identification:

Different type of disks will be having different initials in Linux

- **IDE** drive will be shown as **/dev/hd** (ex: **hda, hdb, hdc**)
- **SCSI, SATA etc.**, drive will be shown as **/dev/sd** (ex: **sda, sdb, sdc**)

### FILE SYSTEM:

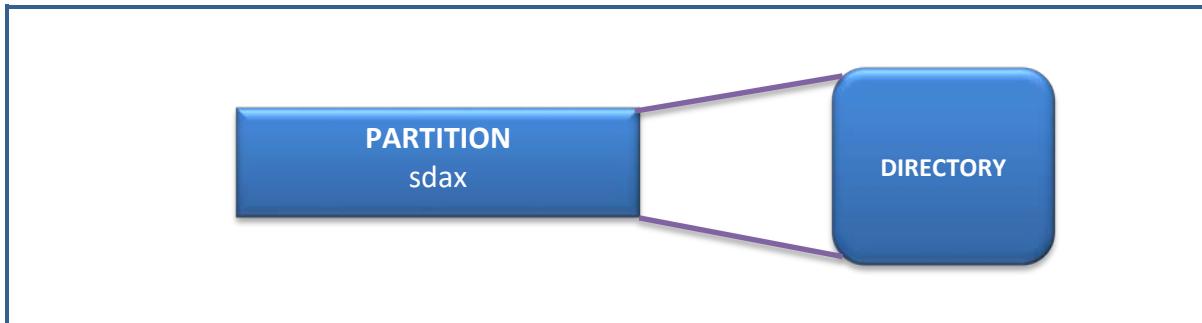
- It is method of storing the data in an organized fashion on the disk. Every partition on the disk except **MBR** and **Extended partition** should be assigned with some file system in order to make them store the data. File system is applied on the partition by formatting it with a particular type of file system.

### Types of file systems supported in RHEL 6 & 7:

- The file systems supported in Linux are **ext2, ext3 ext4, vfat xfs** in RHEL 6, 7&8 etc.
- Ext/xfs** file system is the widely used file system in Linux, whereas vfat is the file system to maintain a common storage between **Linux and windows** ( in case of multiple o/s)

S.NO	EXT2	EXT3	EXT4	XFS
1.	Stands for Second Extended File System	Stands for Third Extended File System	Stands for Fourth Extended File System	Xtents File system/ X File System
2.	It was introduced in 1993	It was introduced in 2001	It was introduced in 2008.	It was introduced in 1993.
3.	Does not have journaling feature.	Supports Journaling Feature.	Supports Journaling Feature.	Supports Journaling Feature.
4.	Maximum File size can be from <b>16 GB to 2 TB</b>	Maximum File Size can be from <b>16 GB to 2 TB</b>	Maximum File Size can be from <b>16 GB to 2 TB</b>	Maximum File Size can be from <b>16TB to 8 EB</b>
5.	Maximum ext2 file system size can be from <b>2 TB to 32 TB</b>	Maximum ext3 file system size can be from <b>2 TB to 32 TB</b>	Maximum ext4 file system size is <b>1 EB (Exabyte)</b> . 1 EB = <b>1024 PB (Petabyte)</b> . <b>1 PB = 1024 TB (Terabyte)</b> .	Maximum xfs file system size is <b>16 EB (Exabyte)</b> . 1 EB = <b>1024 PB (Petabyte)</b> . <b>1 PB = 1024 TB (Terabyte)</b> .
6.	Cannot convert ext file system to ext2.	You can convert an ext2 file system to ext3 file system directly (without backup/restore).	All previous ext file systems can easily be converted into ext4 file system. You can also mount an existing ext3 f/s as ext4 f/s (without having to upgrade it).	N/A

### MOUNTING:-



- It is a method of attaching a directory to the file system in order to access the partition and its file system is known as mounting.
- The mount point is the directory (usually an empty one) in the currently accessible file system to which an additional file system is mounted.
- The /mnt directory exists by default on all Unix-like systems. Its, or usually its subdirectories (such as /mnt/floppy and /mnt/usb), are intended specifically for use as mount points for removable media such as CDROMs, USB key drives and floppy disks.

#### Files which is related to mounting in Linux:

- **/etc/mtab** is a file which stores the information of all the currently mounted file systems; it is dynamic and keeps changing.
- **/etc/fstab** is the file which keeps information about the permanent mount point. If you want to make your mount point permanent, so that it will be mounted even after reboot, then you need to make an appropriate entry in this file.

#### LAB WORK:-

##### To view the existing partitions (RHEL6)

#fdisk -l or parted -l

```
[root@ linux1 ~]# fdisk -l

Disk /dev/sda: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000405e4

      Device Boot      Start        End      Blocks   Id  System
/dev/sda1  *           1          26       204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2            26         2576     20480000   83  Linux
/dev/sda3         2576         2837     2102378+   82  Linux swap / Solaris

[root@ linux1 ~]# parted -l
Model: ATA QEMU HARDDISK (scsi)
Disk /dev/sda: 42.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start    End     Size   Type      File system    Flags
 1      1049kB  211MB   210MB  primary   ext4          boot
 2      211MB   21.2GB  21.0GB  primary   ext4
 3      21.2GB  23.3GB  2153MB primary   linux-swap(v1)
```

**Note:** Observe in the above picture that the device name is **/dev/sda**.

## To view the existing partitions (RHEL7/8)

#fdisk -l or parted -l

```
[root@ rh7 ~]# fdisk -l
Disk /dev/sda: 85.9 GB, 85899345920 bytes, 167772160 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x00082084

Device Boot Start End Blocks Id System
/dev/sda1 * 2048 1026047 512000 83 Linux
/dev/sda2 1026048 52226047 25600000 83 Linux
/dev/sda3 52226048 56420351 2097152 82 Linux swap / Solaris

[root@ rh7 ~]# parted -l
Model: ATA QEMU HARDDISK (scsi)
Disk /dev/sda: 85.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number Start End Size Type File system Flags
1 1049kB 525MB 524MB primary xfs boot
2 525MB 26.7GB 26.2GB primary xfs
3 26.7GB 28.9GB 2147MB primary linux-swap(v1)
```

## Partition Administration using fdisk

To enter into disk utility, the syntax is

#fdisk <disk name>

#fdisk /dev/sda

```
[root@ktcl5 Desktop]# fdisk /dev/sda

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): m
Command action
  a  toggle a bootable flag
  b  edit bsd disklabel
  c  toggle the dos compatibility flag
  d  delete a partition
  l  list known partition types
  m  print this menu
  n  add a new partition
  o  create a new empty DOS partition table
  p  print the partition table
  q  quit without saving changes
  s  create a new empty Sun disklabel
  t  change a partition's system id
  u  change display/entry units
  v  verify the partition table
  w  write table to disk and exit
  x  extra functionality (experts only)
```

Command (m for help):

- Use **m** to list out various options that can be used in fdisk.

## Creating a new partition

#fdisk /dev/sda

- Use **p** to list out the partition information first and
- Use **n** to create a new partition.

```
[root@ tcl5 Desktop]# fdisk /dev/sda
```

```
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').
```

```
Command (m for help): p
```

```
Disk /dev/sda: 32.2 GB, 32212254720 bytes
64 heads, 32 sectors/track, 30720 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00090a50

      Device Boot      Start        End      Blocks   Id  System
/dev/sda1  *           2         201     204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2            202        8201    8192000   83  Linux
Partition 2 does not end on cylinder boundary.
/dev/sda3            8202       12201   4096000   83  Linux
Partition 3 does not end on cylinder boundary.
/dev/sda4            12202      30720   18963456   5   Extended
Partition 4 does not end on cylinder boundary.
/dev/sda5            12204      15203   3072000   83  Linux
/dev/sda6            15205      17204   2048000   82  Linux swap / Solaris
```

```
Command (m for help): l
```

**Now use *n* to create a new partition and verify it again with *p*.**

```
Command (m for help): n
```

```
First cylinder (12202-30720, default 12202): 17205
```

```
Last cylinder, +cylinders or +size{K,M,G} (17205-30720, default 30720): +500M
```

```
Command (m for help): p
```

```
Disk /dev/sda: 32.2 GB, 32212254720 bytes
64 heads, 32 sectors/track, 30720 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00090a50
```

```
      Device Boot      Start        End      Blocks   Id  System
/dev/sda1  *           2         201     204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2            202        8201    8192000   83  Linux
Partition 2 does not end on cylinder boundary.
/dev/sda3            8202       12201   4096000   83  Linux
Partition 3 does not end on cylinder boundary.
/dev/sda4            12202      30720   18963456   5   Extended
Partition 4 does not end on cylinder boundary.
/dev/sda5            12204      15203   3072000   83  Linux
/dev/sda6            15205      17204   2048000   82  Linux swap / Solaris
/dev/sda7            17205      17705   513008   83  Linux
```

## Deleting a partition

Let's delete the partition we've created above i.e. /dev/sda7

- Use **d** to delete a partition and specify the device name, in our case it is **7**.

```
Command (m for help): d
Partition number (1-7): 7
```

**Note:** Never delete the system partitions i.e. **1-3**

## Saving the partition changes

Every time you make a partition or delete a partition, the changes made has to be saved using **w**, otherwise the creation and deletion will not be considered to be happen. For practice purpose you can make any no. of partition and delete it and just quit using **q** so that it will not be saved.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
[root@ cl5 Desktop]#
```

## Updating the partition table without restarting the system

After creating or deleting a partition the changes will be effected in the partition table only after the restart of the system. But there is a way to avoid this circumstance. We can use **partprobe** or **partx** command to update the partition information without restarting the system. **Note: in RHEL8 it is not needed to manually update, as it gets update by default.**

```
#partprobe /dev/sda
Or
#partx -a /dev/sda (while adding), #partx -d /dev/sda (while deleting)
Or
#kpartx /dev/sda
```

**Note:** In **RHEL6** **partprobe** is not functioning properly, so it is recommended to use **partx**, whereas in **RHEL7** it is functioning perfectly.

- Read the following file to check status of partition table
- **#cat /proc/partitions**

**RHEL6**

```
[root@musabl ~]# partx -a /dev/sda
BLKPG: Device or resource busy
error adding partition 1
BLKPG: Device or resource busy
error adding partition 2
BLKPG: Device or resource busy
error adding partition 3
[root@musabl ~]# cat /proc/partitions
major minor #blocks name
8 0 41943040 sda
8 1 204800 sda1
8 2 25600000 sda2
8 3 2097152 sda3
8 4 31 sda4
8 5 515977 sda5
```

**RHEL7**

```
[root@ rh7 ~]#
[root@ rh7 ~]# partprobe /dev/sda
[root@ rh7 ~]# cat /proc/partitions
major minor #blocks name
11 0 3655680 sr0
8 0 83886080 sda
8 1 512000 sda1
8 2 25600000 sda2
8 3 2097152 sda3
8 4 1 sda4
8 5 512000 sda5
```

**"Now then we have learnt creating a partition. Let's see how to format a partition with a particular file system"**

### Formatting a partition with ext4 filesystem in rhel6, 7 or 8

After creating a partition we need to assign some file system to it so that we can start storing the data into it. To format a partition the following syntax is used.

```
# mkfs.<file system type> <partition name>
#mkfs.ext4 /dev/sda7 (where sda7 is our newly created partition)
```

```
[root@mlinux6 ~]# mkfs.ext4 /dev/sda5
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
129024 inodes, 515976 blocks
25798 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
63 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409
```

- Likewise you can format the different partitions with different file systems like
- #mkfs.ext3 /dev/sdax
- #mkfs.vfat /dev/sdax

### Formatting a partition with xfs in rhel7

```
#mkfs.xfs /dev/sda5
```

```
[root@ rh7 ~]# mkfs.xfs /dev/sda5
meta-data=/dev/sda5              isize=256    agcount=4, agsize=32000 blks
                                =          sectsz=512  attr=2, projid32bit=1
                                =
                                =
data     =                      bsize=4096   blocks=128000, imaxpct=25
                                =
                                =
naming  =version 2             bsize=4096   ascii-ci=0 fttype=0
log      =internal log         bsize=4096   blocks=853, version=2
                                =
realtime =none                 sectsz=512   sunit=0 blks, lazy-count=1
                                extsz=4096   blocks=0, rtextents=0
```

**Note:** Even after formatting the partition, we cannot add the data into the partition. In order to add the data in the partition it is required to be mounted.

## Mounting a partition

Mounting is a procedure where we attach a directory to the file system. There are two types of mounting which will be used in Linux or any UNIX.

- **Temporary Mounting**
- **Permanent Mounting**

### Temporary Mounting

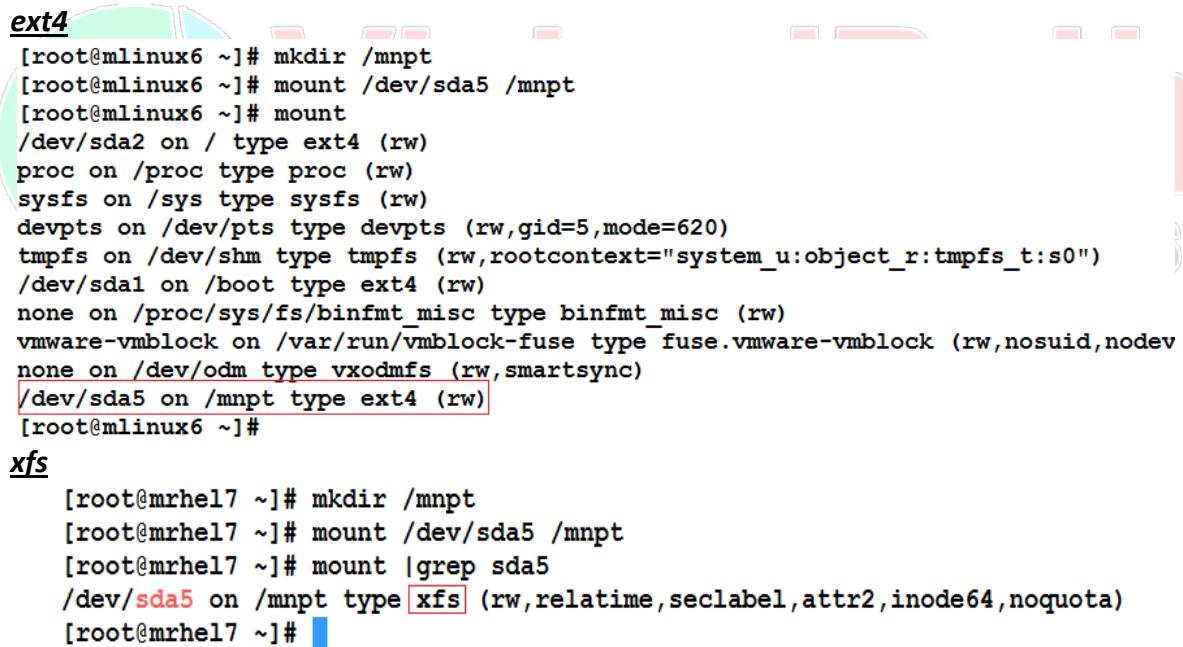
In a temporary mount point we will create a directory and mount it, but this mount point will last only till the system is up, once it is rebooted the mounting will be lost.

Syntax:

```
#mount <device name> <directory name (mount point)>
#mount /dev/sda5 /mnpt
[root@mrhel7 ~]# mkdir /mnpt
[root@mrhel7 ~]# mount /dev/sda5 /mnpt
```

### To View all the mounted partitions

#mount



```
ext4
[root@mlinux6 ~]# mkdir /mnpt
[root@mlinux6 ~]# mount /dev/sda5 /mnpt
[root@mlinux6 ~]# mount
/dev/sda2 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda1 on /boot type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
vmware-vmblock on /var/run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev
none on /dev/odm type vxcodmfs (rw,SMARTsync)
/dev/sda5 on /mnpt type ext4 (rw)
[root@mlinux6 ~]#
xfs
[root@mrhel7 ~]# mkdir /mnpt
[root@mrhel7 ~]# mount /dev/sda5 /mnpt
[root@mrhel7 ~]# mount |grep sda5
/dev/sda5 on /mnpt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
[root@mrhel7 ~]#
```

- Now we have successfully mounted the partition we can access it and can store the data
- To add the data access the mount point
- #cd /mnpt. Now, add the data and exit the directory

### Unmounting a partition

#umount <mount point directory>

#umount /mnpt

Verify it with **mount** command.

### Permanent Mounting

Permanent mounting procedure is exactly same like temp mounting, but here we will update the **/etc/fstab** file with the mounting details, so that it will be mounted even after the system is reboot.

#### Steps To make a permanent mount point:

- Make a directory or use an existing directory
- Add entry in **/etc/fstab** file
- Use **mount -a** command to check it is mounting. (**mount -a** will mount all the entry placed in **/etc/fstab**)

Here we will be using our existing **/kernel** directory as mount point which is created previously.

#vim **/etc/fstab**

```
#  
# /etc/fstab  
# Created by anaconda on Wed Sep  9 05:29:56 2015  
#  
# Accessible filesystems, by reference, are maintained under '/dev/disk'  
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info  
#  
UUID=f25262b9-f44f-42fb-8cba-663b1e77cd83 / xfs defaults 1 1  
UUID=bce2a1cf-2138-4570-8180-e13c710fbfb6 /boot xfs defaults 1 2  
UUID=d43b526a-93d0-4523-9499-b22e50c90329 swap swap defaults 0 0  
/dev/sda5 /mnpt ext4 defaults 0 0  
/dev/sda6 /mnpt2 xfs defaults 0 0
```



Note: For xfs, you can put xfs in place ext4

#mount -a

```
[root@mlinux7 ~]# mount -a  
[root@mlinux7 ~]# mount |grep sda*  
/dev/sda2 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)  
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)  
/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)  
gvfsd-fuse on /run/user/0/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,rel.  
/dev/sda5 on /mnpt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)  
/dev/sda6 on /mnpt2 type ext4 (rw,relatime,seclabel,data=ordered)  
[root@mlinux7 ~]#  
[root@mlinux7 ~]#
```

You can now access the directory and add, delete or modify the contents and can also unmount the file system at any point

### Mounting a partition permanently with its block id (UUID)

- To check the uuid of a partition use **blkid /dev/sda7** command.
- Copy the uuid
- Make an entry in **/etc/fstab** using UUID
- Verify it with **mount -a** option

```
[root@ cl5 ~]# blkid /dev/sda7
/dev/sda7: LABEL= UUID="f489d0b1-ffca-4e21-917a-7b82c0edd255" TYPE="ext4"
[root@ cl5 ~]#
```

### #vim /etc/fstab

tmpfs	/dev/shm	tmpfs	defaults	0 0
devpts	/dev/pts	devpts	gid=5,mode=620	0 0
sysfs	/sys	sysfs	defaults	0 0
proc	/proc	proc	defaults	0 0
UUID=f489d0b1-ffca-4e21-917a-7b82c0edd255		/mnpt2	ext4 defaults	0 0

Now mount it with **mount -a** command and verify it with **mount** command

Sometimes a directory reflects error while unmouting, the possible causes for it are

- You are in the same directory and trying to unmount it. Check with **pwd** command
- Some users are present in the directory and using the contents in it.
- Check with **fuser -cu /mnpt or /dev/sda5 and lsof /mnpt or /dev/sda6**

```
[root@mlinux7 ~]# fuser -cu /mnpt
/mnpt: 44673c(myuser) 44710c(myuser)
[root@mlinux7 ~]# lsof /mnpt
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bash 44673 myuser cwd DIR 8,5 6 128 /mnpt
vim 44710 myuser cwd DIR 8,5 6 128 /mnpt
```

- Kill the open connections using **fuser -ck /mnpt**, you could also try **umount -l** (lazy unmount)
- Now you can use **umount** command to unmount the file system.

### To view the usage information of mounted partition:

To view the usage information of mounted partition use the command **df -h**

**#df -h,**

```
[root@mlinux7 ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/sda2 20G 2.9G 18G 15% /
devtmpfs 899M 0 899M 0% /dev
tmpfs 913M 140K 913M 1% /dev/shm
tmpfs 913M 9.1M 904M 1% /run
tmpfs 913M 0 913M 0% /sys/fs/cgroup
/dev/sda1 497M 154M 344M 31% /boot
tmpfs 183M 32K 183M 1% /run/user/0
/dev/sda5 497M 26M 472M 6% /mnpt
/dev/sda6 190M 1.6M 175M 1% /mnpt2
tmpfs 183M 0 183M 0% /run/user/1000
[root@mlinux7 ~]#
```

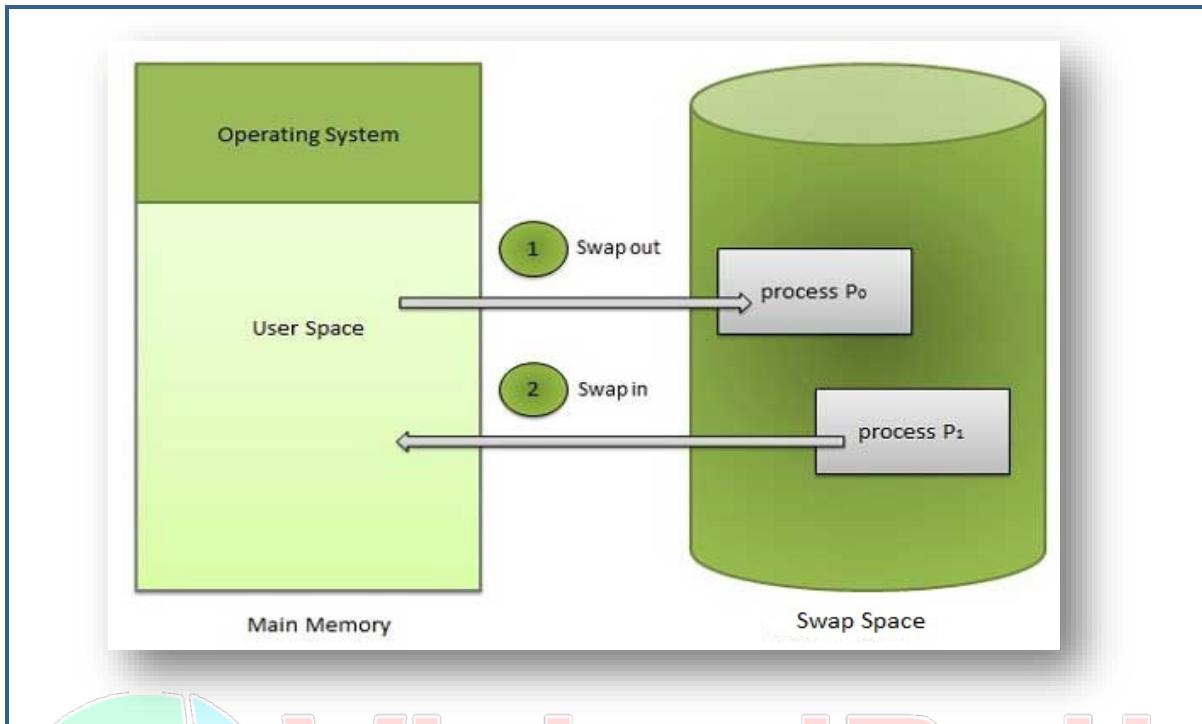
### To view the size of a file or directory

To view the size of the file or directory uses the command **du -h file or directory name**.

### To see more details about partition, FS and uuid of the partition

Also see **#lsblk -l , #lsblk -s, #lsblk -Fs**

## SWAP SPACES MANAGEMENT



Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM. Swap space is located on hard drives, which have a slower access time than physical memory.

### Recommended System Swap Space

Amount of RAM in the System	Recommended Amount of Swap Space
4GB of RAM or less	a minimum of 2GB of swap space
4GB to 16GB of RAM	a minimum of 4GB of swap space
16GB to 64GB of RAM	a minimum of 8GB of swap space
64GB to 256GB of RAM	a minimum of 16GB of swap space
256GB to 512GB of RAM	a minimum of 32GB of swap space
512GB TO 1TB of RAM	a minimum of 64 GB of swap space

#### The Basic Rule for the Size of SWAP:

Apart from the above recommendation a basic rule is applied to create the swap partitions

- if the size of the RAM is **less than or equal to 2GB**, then size of **SWAP=2 X RAM SIZE**
- If the size of the RAM is **more than 2GB**, then size of **SWAP= 2GB + size of the RAM**

Swap space is compulsory to be created at the time of installation. But, additional swap spaces can be created and deleted at any point of time, when it is required. Sometimes we need to increase the swap space, so we create additional swap spaces which will be added to the existing swap space to increase the size.

## Commands to be used in maintaining Swap spaces

- To see the memory size and the swap space size  
**#free -m, #free -h**
- To see the swap usage use  
**#swapon -s**
- To format the partition with swap file system use  
**#mkswap <partition name>**
- To activate the swap space use  
**#swapon <partition name>**
- To deactivate the swap space use  
**#swapoff <partition name>**

## Creating a Swap partition

- Create a normal partition using fdisk and change hex code to make it swap partition.
- The hex code for SWAP is **82**. (To change the use **t** in fdisk and list all the hex code use **I**)
- Update the partition table using **partprobe** command

```
[root@ linux /]# fdisk /dev/sda

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): n
First cylinder (4426-6527, default 4426):
Using default value 4426
Last cylinder, +cylinders or +size{K,M,G} (4426-6527, default 6527): +500M

Command (m for help): t
Partition number (1-6): 6
Hex code (type L to list codes): 82
Changed system type of partition 6 to 82 (Linux swap / Solaris)

Command (m for help): p

Disk /dev/sda: 53.7 GB, 53687091200 bytes
255 heads, 63 sectors/track, 6527 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0006d4e7

      Device Boot   Start     End   Blocks   Id  System
/dev/sda1            1    3825  30720000   8e  Linux LVM
/dev/sda2    *       3825    3851    204800   83  Linux
/dev/sda3       3851    4361   4096000   82  Linux swap / Solaris
/dev/sda4       4361    6527  17406303+   5  Extended
/dev/sda5       4361    4425     521957   83  Linux
/dev/sda6       4426    4490     522081   82  Linux swap / Solaris
```

## Format the partition with swap file system

```
#mkswap /dev/sda6
```

```
[root@ linux ]# mkswap /dev/sda6
Setting up swap space version 1, size = 522076 KiB
no label, UUID=d3d25afa-71d6-4339-a88a-8640f2680a74
[root@ linux ]#
```

## Turn on the newly created swap space and verify it.

- To turn on the swap space the syntax is

```
#swapon /dev/sda6
```

```
[root@ linux ]# swapon /dev/sda6
[root@ linux ]# swapon -s
Filename                                Type      Size   Used   Priority
/dev/sda3                               partition 4095992 0     -1
/dev/sda6                               partition 522072  0     -2
[root@ linux ]# free -m
              total     used     free   shared   buffers   cached
Mem:       2007      741     1266        0         4      272
-/+ buffers/cache:  464      1543
Swap:      4509      0      4509
```

## Making the Newly Created SWAP Partition to be activated after reboot

- In order to make the swap partition mount automatic after reboot, we need to make an entry in **/etc/fstab** file.

```
#vim /etc/fstab
```

```
# /etc/fstab
# Created by anaconda on Wed Nov 10 06:40:21 2010
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/vg_l... linux-rootlv /          ext4    defaults    1 1
UUID=ce33cb92-21b8-49c0-95fc-17f40437d44b /boot      ext4    defaults    1 2
/dev/mapper/vg_l... linux-homelv /home       ext4    defaults    1 2
/dev/mapper/vg_l... linux-usrlv /usr        ext4    defaults    1 2
/dev/mapper/vg_l... linux-varlv /var        ext4    defaults    1 2
UUID=60dcea45-f68b-473d-b953-0fbcc5b63d5fc swap      swap    defaults    0 0
tmpfs           /dev/shm      tmpfs   defaults    0 0
devpts          /dev/pts      devpts  gid=5,mode=620 0 0
sysfs           /sys         sysfs   defaults    0 0
proc             /proc        proc    defaults    0 0
/dev/mapper/ tpart /k        ext4    defaults    0 0
/dev/sda6        swap        swap    defaults    0 0
```

Note: In place of device name (sda6) blkid can also be used

- To Test auto activation of swap after putting entries in fstab use the following command  
**#swapon -a** (*It works the same way how mount -a would work*)

## Removing the SWAP Partition

- Deactivate the swap partition  
**#swapoff <device name>**
- Remove the entry from **/etc/fstab**.
- Delete the partition through **fdisk**

## Using File as a Swap Space:

At times it is seen that disks may run out of space and a partition may not be created to add a new swap space. In this situation a file can also be used as a swap space.

### Steps to Create File as a swap space.

**Step1:** Create a file with required size like 1G for example, using dd command

```
#dd if=/dev/urandom of=/swap-file bs=1M count=1024
```

```
[root@mlinux7 ~]# dd if=/dev/urandom of=/swap-file bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 9.04193 s, 119 MB/s
[root@mlinux7 ~]#
[root@mlinux7 ~]# du -h /swap-file
1.0G    /swap-file
```

**Step2:** Change the permission of the file to 600

```
[root@mlinux7 ~]# ls -l /swap-file
-rw-r--r--. 1 root root 1073741824 Jan 24 20:26 /swap-file
[root@mlinux7 ~]# chmod 600 /swap-file
[root@mlinux7 ~]# ls -l /swap-file
-rw-----. 1 root root 1073741824 Jan 24 20:26 /swap-file
```

**Step3:** Format the file with swap file system

```
#mkswap -f /swap-file
```

```
[root@mlinux7 ~]# mkswap -f /swap-file
mkswap: /swap-file: warning: wiping old swap signature.
Setting up swapspace version 1, size = 1048572 KiB
no label, UUID=1a56a0cf-7f74-40ef-805f-72cb7788abb3
```

**Step4:** Activate the swap file

```
#swapon /swap-file
```

```
[root@mlinux7 ~]# swapon /swap-file
[root@mlinux7 ~]# swapon -s
Filename                                     Type      Size   Used   Priority
/dev/sda3                                    partition 4194300 1664    -2
/swap-file                                    file     1048572 0        -3
```

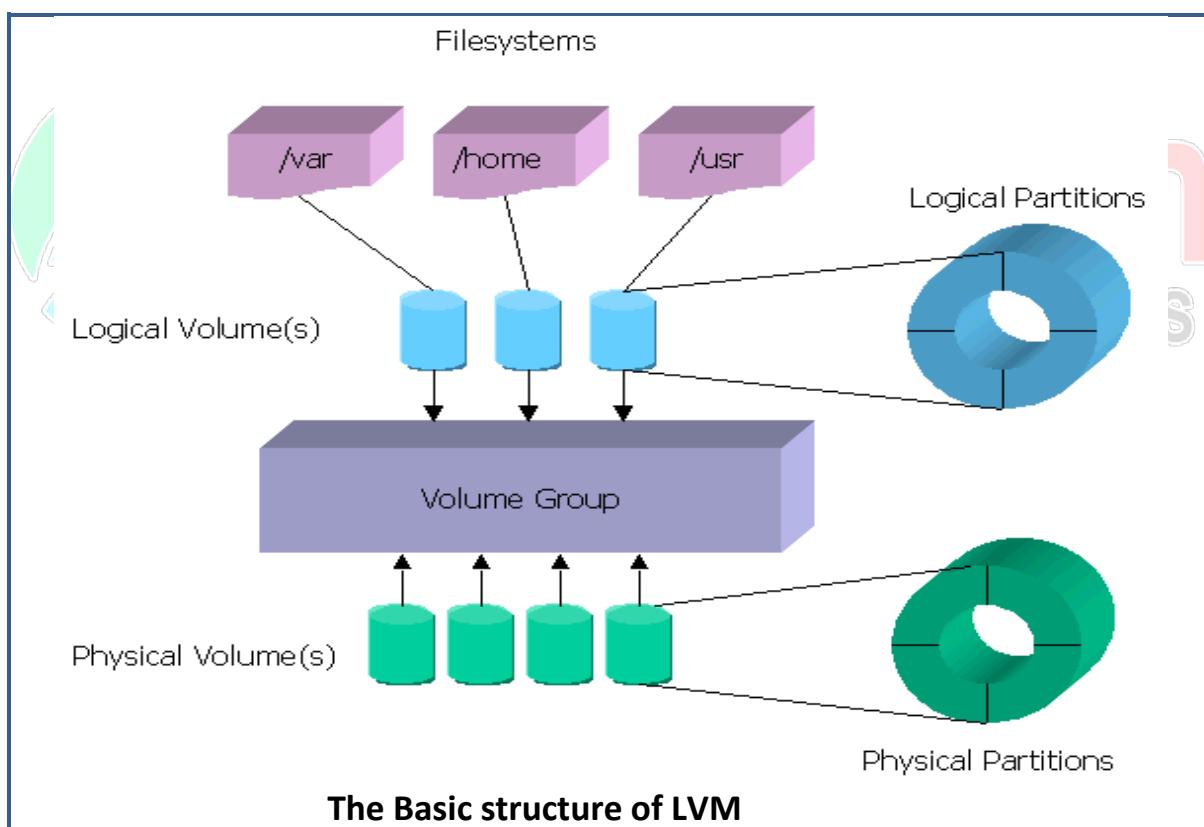
**Step5:** Make it permanent by adding entry in /etc/fstab

```
[root@mlinux7 ~]# vim /etc/fstab
[root@mlinux7 ~]# tail -3 /etc/fstab
/dev/sda5          /test        xfs        defaults      0 0
/dev/myvg/mylv    /my         xfs        defaults      0 0
UUID="1a56a0cf-7f74-40ef-805f-72cb7788abb3" swap      swap    defaults      0 0
```

## Logical Volume Management

The Linux Logical Volume Manager (LVM) is a mechanism to virtualize the disks. It can create "virtual" disk partitions out of one or more physical hard drives, allowing you to grow, shrink, or move those partitions from drive to drive as your needs change. It also allows you to create larger partitions than you could achieve with a single drive. Traditional uses of LVM have included databases and company file servers, but even home users may want large partitions for music or video collections, or for storing online backups. LVM can also be convenient ways to gain redundancy without sacrificing flexibility.

A typical example for the need of LVM can be, assuming that we are having a disk of size 2GB and we start adding the data in the form of a single file, eventually it grows to the size of 2GB. In this case the possibility is, you go for another disk which is larger than 2GB, let's say 4GB. But what if the file again grows more than 4GB? How far you will be migrating file from one disk to another so on and so forth? It requires a down time as well which is not possible in real time, so to avoid these circumstances we implement LVM and store data in LV's whose size can be easily increased whenever required without a downtime.



Above picture shows the structure of LVM. LVM consists of **Physical Volumes**, **Volume Group**, **Logical Volumes** and finally **file systems**. The Physical partitions are known as **Physical Extents (PE)**, and the logical partitions are known as **logical Extents (LE)**

## Components of LVM in Linux:

- **Physical Volumes (PV)**
- **Physical Extent (PE)**
- **Volume Group (VG)**
- **Logical Volume (LV)**
- **Logical Extent (LE)**

### Physical Volume (PV)

It is the standard partition that you add to the LVM. Normally, a physical volume is a standard primary or logical partition with the hex code **8e**.

### Physical Extent (PE)

It is a chunk of disk space. Every PV is divided into a number of equal sized PEs.

### Volume Group (VG)

It is composed of a group of PV's and LV's. It is the organizational group for LVM.

### Logical Volume (LV)

It is composed of a group of LEs. You can format and mount any file system on an LV. The size of these LV's can easily be increased or decreased as per the requirement.

### Logical Extent (LE)

It is also a chunk of disk space. Every LE is mapped to a specific PE.

LVM Command	Function
<b>pvs</b>	Displays all the physical volumes
<b>vgs</b>	Displays all volume groups in the system
<b>lvs</b>	Displays all the logical volumes in the system
<b>pvdisplay</b>	Displays detailed information on physical volumes
<b>vgdisplay</b>	Displays detailed information on volume groups
<b>lvdisplay</b>	Displays detailed information on logical volumes
<b>pvcreate</b>	Create a new physical volume
<b>vgcreate</b>	Create a new volume group.
<b>lvcreate</b>	Creates a new logical volume
<b>vgextend</b>	Add a new physical disk to a volume group.
<b>vgreduce</b>	Reduces a volume group by removing a PV from it.
<b>lvextend</b>	Extends the size of a logical volume
<b>lvreduce</b>	Reduces the size a logical volume
<b>lvresize</b>	Resizes a logical volume, i.e., increase as well as decrease the size
<b>pvmove</b>	Moves the contents of a PV from one PV to another
<b>lvremove</b>	Removes /Deletes a logical volume
<b>vgremove</b>	Removes /Deletes a volume group
<b>pvremove</b>	Removes/Deletes a PV

## LAB WORK:-

### Creating a Physical Volume (PV)

- Create a partition using fdisk, and change the hex code of it to **8e**.
- Save and exit the fdisk and update the partition table using **partx -a** command

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		1	3825	30720000	8e	Linux LVM
/dev/sda2	*	3825	3851	204800	83	Linux
/dev/sda3		3851	4361	4096000	82	Linux swap / Solaris
/dev/sda4		4361	6527	17406303+	5	Extended
/dev/sda5		4361	4425	521957	83	Linux
/dev/sda6		4426	4490	522081	82	Linux swap / Solaris
/dev/sda7		4491	4555	522081	83	Linux

```
Command (m for help): [t]
Partition number (1-7): 7
Hex code (type L to list codes): [8e]
Changed system type of partition 7 to 8e (Linux LVM)
```

- Create a PV on newly created partition i.e. **/dev/sda7**.
- Verify it by **pvs** or **pvdisplay** command
- **Syn:** #**pvcreate <partition name>**  
#**pvcreate /dev/sda7**

```
[root@ linux Desktop]# pvcreate /dev/sda7
Physical volume "/dev/sda7" successfully created
```

```
[root@ linux Desktop]# pvs
PV          VG      Fmt Attr PSize   PFree
/dev/sda1  vg    linux lvm2 a-  29.29g  1.95g
/dev/sda7           lvm2 a-  509.84m 509.84m
```

```
[root@ linux Desktop]# pvdisplay
"/dev/sda7" is a new physical volume of "509.84 MiB"
--- NEW Physical volume ---
PV Name          /dev/sda7
VG Name
PV Size         509.84 MiB
Allocatable     NO
PE Size          0
Total PE         0
Free PE          0
Allocated PE     0
PV UUID          RzuHEg-ks6y-cvem-C5F4-tfk8-veco-mJqs46
```

- The above command will list all the PVs in the system, if you want to see the details only for a particular PV, then use  
**#pvdisplay <partition name>** i.e. **#pvdisplay /dev/sda7**

### Creating a Volume Group (VG)

- After creating a PV, the next step is to create a **Volume Group** or **VG**
- To create a VG the syntax is  
**#vgcreate <name for the VG> <PV name>**  
**#vgcreate myvg /dev/sda7**

```
[root@mlinux7 ~]# vgcreate myvg /dev/sda7
Volume group "myvg" successfully created
```

- Verify it by using the following command  
**#vgs** or **#vgdisplay <vgname>**

```
[root@mlinux7 ~]# vgs
  VG #PV #LV #SN Attr   VSize   VFree
  myvg   1   0   0 wz--n- 508.00m 508.00m
[root@mlinux7 ~]# vgdisplay
--- Volume group ---
VG Name           myvg
System ID
Format           lvm2
Metadata Areas   1
Metadata Sequence No  1
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size          508.00 MiB
PE Size          4.00 MiB
Total PE         127
Alloc PE / Size  0 / 0
Free PE / Size   127 / 508.00 MiB
VG UUID          qfPegi-5PXa-xZF0-9Aqc-I3IO-YaXL-w3RXpj
```



- To check all the **VGs** detail you can also use the command  
**#vgdisplay**
- It will list out all the VGs in the system in detail.

## Logical Volume Creation

- Once we are ready with a **Volume Group** then it's the time to create a **Logical Volume LV**
- The syntax for creating an **LV** is
- #lvcreate -L <size of LV> -n <name for LV> <VG name>**
- #lvcreate -L 300M -n mylv myvg** (To create a LV of 300MB)

```
[root@mlinux7 ~]# lvcreate -L 300M -n mylv myvg
Logical volume "mylv" created.
[root@mlinux7 ~]#
[root@mlinux7 ~]#
```

- Verify the **LV** by using the following commands
- #lvs** or **#lvdisplay** to display all the **LVs** available in the system
- #lvdisplay <VG name>** to display the **LVs** of a particular **Volume Group**
- #lvdisplay myvg**

```
[root@mlinux7 ~]# lvs
  LV   VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  mylv myvg -wi-a---- 300.00m
[root@mlinux7 ~]# lvdisplay myvg
--- Logical volume ---
  LV Path          /dev/myvg/mylv
  LV Name         mylv
  VG Name         myvg
  LV UUID         MsjU7i-DRzu-UrGo-ESAO-dowh-2mJz-tz3Mgu
  LV Write Access read/write
  LV Creation host, time mlinux7.mb.com, 2017-02-14 16:07:54 +0530
  LV Status        available
  # open           0
  LV Size          300.00 MiB
  Current LE       75
  Segments          1
  Allocation       inherit
  Read ahead sectors    auto
  - currently set to 8192
  Block device     253:0
```

- Note:** The output for only **lvdisplay** command is very lengthy to show, it is recommended that you run the command on the system and check it out. The syntax is given above.

### Adding File system to the LV and Mounting it.

- As per now we have our VG created so is our LV. In order make it accessible we need to format it with a file system like ext4 or xfs
- The syntax for formatting an LV is exactly like formatting a normal partition, Instead of **/dev/partition name** we use the path of **LV** that will be something like **/dev/vg/lv**
- #mkfs.ext4 /dev/myvg/mylv**                          **or**                          **#mkfs.xfs /dev/myvg/mylv**

[root@mlinux7 ~]# mkfs.ext4 /dev/myvg/mylv mke2fs 1.42.9 (28-Dec-2013) Filesystem label= OS type: Linux Block size=1024 (log=0) Fragment size=1024 (log=0) Stride=0 blocks, Stripe width=0 blocks 76912 inodes, 307200 blocks 15360 blocks (5.00%) reserved for the super user First data block=1 Maximum filesystem blocks=33947648 38 block groups 8192 blocks per group, 8192 fragments per group 2024 inodes per group Superblock backups stored on blocks: 8193, 24577, 40961, 57345, 73729, 204801, 221185	[root@mlinux7 ~]# mkfs.xfs -f /dev/myvg/mylv meta-data=/dev/myvg/mylv              isize=256  agcount=4, agsize=19200 blks =              agcount=4, agsize=19200 blks =              sectsz=512  attr=2, projid32bit=1 =              crc=0      finobt=0 data                                  =              bsize=4096  blocks=76800, imaxpct=25 =              sunit=0      swidth=0 blks naming                              =version 2      bsize=4096  ascii-ci=0 ftype=0 log                                  =internal log  bsize=4096  blocks=853, version=2 =              sectsz=512  sunit=0 blks, lazy-count=1 realtime                           =none              extsz=4096  blocks=0, rtextents=0
---	---

### Mounting:

- Mounting an LV is exactly same like a normal partition, again the path for mounting will be **/dev/vg/lv**
- Create a directory over which the LV should be mounted.
- #mount </dev/vgname/lvname> /directory name**
- #mount /dev/myvg/mylv /mydir**
- Verify the mounting with **mount** command
- Make it a permanent mount by making an entry in **/etc/fstab**

```
[root@mlinux7 ~]# mkdir /mydir  
[root@mlinux7 ~]# mount /dev/myvg/mylv /mydir
```

```
#vim /etc/fstab  
  
#  
# /etc/fstab  
# Created by anaconda on Tue Feb 14 12:16:02 2017  
#  
# Accessible filesystems, by reference, are maintained under '/dev/disk'  
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info  
#  
UUID=f896b19f-142b-4796-919a-2f32e61e0cd1 /                          xfs  defaults      0  0  
UUID=42298fe6-71d8-4c41-a3dc-8a9dcffe4e40 /boot                  xfs  defaults      0  0  
UUID=5c384537-24fb-42c5-a763-0f02d6f33ae6 swap                  swap  defaults      0  0  
/dev/myvg/mylv                                          xfs  defaults      0  0
```

- Now you can access it and add the data as usual.

### Extending a Volume Group

- Extending a volume group is actually adding a new PV to the volume group.
- To extend a volume group we need to create a new partition using fdisk. Don't forget to change its **hex code** to **8e** and update the partition table using **partprobe** command
- Create a PV on the newly created partition using **pvcreate** command
- Add the partition to the **VG** using **vgextend** command, the syntax for it is
- #**vgextend <VG name> <PV name>**
- #vgextend myvg /dev/sda8**
- Verify it **pvs** command

```
[root@mlinux7 ~]# pvcreate /dev/sda8
Physical volume "/dev/sda8" successfully created
[root@mlinux7 ~]# pvs
PV          VG   Fmt Attr PSize  PFree
/dev/sda7  myvg lvm2 a--  508.00m 208.00m
/dev/sda8            lvm2 ---  500.00m 500.00m
[root@mlinux7 ~]#
[root@mlinux7 ~]# vgextend myvg /dev/sda8
Volume group "myvg" successfully extended
[root@mlinux7 ~]# pvs
PV          VG   Fmt Attr PSize  PFree
/dev/sda7  myvg lvm2 a--  508.00m 208.00m
/dev/sda8  myvg lvm2 a--  496.00m 496.00m
[root@mlinux7 ~]#
```

### Increasing the size of a logical volume

- Sometimes the file system size may be full, so we need to increase the size of the LV to continue adding the data in it.
- The size of LV can be increased online, no downtime is required.
- Check the current size of the LV by using **#df -h** command.
- Increase the size of the LV by using **lvextend** or **lvresize** command, the syntax for it is
- #**lvextend -L <+addition size> </dev/vg/lv name>** (syntax for **lvresize** is also same)
- #lvextend -L +200M /dev/myvg/mylv**
- Update the file system by using **resize2fs** command in case of ext filesystem and use **xfs\_growfs** in case of xfs filesystem
- #resize2fs /dev/vg/lv name, #xfs\_growfs /dev/vg/lv**
- #resize2fs /dev/myvg/mylv , #xfs\_growfs /dev/myvg/mylv**
- Verify the change by using **df -h** command

```
[root@mlinux7 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2       20G  2.9G  18G  15% /
devtmpfs        899M    0  899M  0% /dev
tmpfs          913M   84K  913M  1% /dev/shm
tmpfs          913M  9.0M  904M  1% /run
tmpfs          913M    0  913M  0% /sys/fs/cgroup
/dev/mapper/myvg-my lv 297M   16M  282M  6% /mydir
/dev/sda1       497M  154M  344M 31% /boot
```

- Increasing the size of the LV and updating the file system**

<pre>[root@mlinux7 ~]# lvextend -L +200M /dev/myvg/mylv Size of logical volume myvg/mylv changed from 300.00 MiB (75 extents) to 500.00 MiB (125 extents). Logical volume mylv successfully resized.  [root@mlinux7 ~]# xfs_growfs /dev/myvg/mylv meta-data=/dev/mapper/myvg-myvg  isize=256           =               sectsz=512           =               crc=0   data    =               bsize=4096           =               sunit=0  naming   =version 2      bsize=4096  log      =internal       bsize=4096           =               sectsz=512  realtime =none           extsz=4096  data blocks changed from 76800 to 128000 For xfs Filesystem</pre>	<pre>[root@mlinux6 ~]# resize2fs /dev/myvg/mylv resize2fs 1.41.12 (17-May-2010) Filesystem at /dev/myvg/mylv is mounted on /mydir; old_desc_blocks = 2, new_desc_blocks = 2 Performing an on-line resize of /dev/myvg/mylv to The filesystem on /dev/myvg/mylv is now 512000 blo For ext Filesystems</pre>
---	--

- Verify it by df -h**

<pre>[root@mlinux7 ~]# df -h Filesystem           Size  Used Avail Use% Mounted on /dev/sda2            20G  2.9G  18G  15% / devtmpfs             899M    0  899M  0% /dev tmpfs                913M   84K  913M  1% /dev/shm tmpfs                913M   9.1M  904M  1% /run tmpfs                913M    0  913M  0% /sys/fs/cgroup /dev/mapper/myvg-myvg 497M   16M  482M  4% /mydir /dev/sda1            497M  154M  344M  31% /boot tmpfs                183M   16K  183M  1% /run/user/42 tmpfs                183M    0  183M  0% /run/user/0</pre>	
---	--

### Reducing the size of the LV

- Reducing the size of an LV is a bit complicated task, there are few things which you need to keep in mind before reducing the size of an LV.
  - LV size cannot be reduced online, it requires a down time i.e. unmounting the file system.
  - Check the consistency of the File system.
  - Update the file system about the size. I.e. what its size will be after reduction.
  - Finally reduce the size. **Huh....! Lots of things to do!!!!**
- If any of the above things are missed then it will be a mess, you may corrupt the file system and LV.

#### **Let's start the steps carefully**

- Check the size of the lv using **df -h** command
- Unmount the LV using **umount** command
- Check the consistency of file system by using **e2fsck** command
- #e2fsck -f /dev/myvg/mylv.**
- Update the file system by using **resize2fs** command
- #resize2fs /dev/myvg/mylv 300M** (where **300M** is the approximate total size of LV after reduction)

- Now reduce the size by using # **lvreduce -L -200M /dev/myvg/mylv** command
- We know the size of LV is around 500MB, from previous picture in case of extending the size of LV.
- Or else you can run **df -h** and verify it again.
- Unmount the LV by using **umount** command

```
[root@mlinux6 ~]# umount /mydir
```

- Check the consistency of the file system.

```
[root@mlinux6 ~]# e2fsck -f /dev/myvg/mylv
e2fsck 1.41.12 (17-May-2010)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/myvg/mylv: 11/127512 files (0.0% non-contiguous),
```

- Update the file system about the size after reduction

```
[root@mlinux6 ~]# resize2fs /dev/myvg/mylv 300M
resize2fs 1.41.12 (17-May-2010)
Resizing the filesystem on /dev/myvg/mylv to 307200 (1k) blocks.
The filesystem on /dev/myvg/mylv is now 307200 blocks long.
```

- Finally reduce the size of the LV using **lvreduce** command. It will prompt you about the change type **y** to continue with reduction.

```
[root@mlinux6 ~]# lvreduce -L -200M /dev/myvg/mylv
WARNING: Reducing active logical volume to 300.00 MiB
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce mylv? [y/n]: y
Size of logical volume myvg/mylv changed from 500.00 MiB (125 extents) to 300.00 MiB (75 extents).
Logical volume mylv successfully resized
```

- Mount the LV and run the command **df-h**, to verify the change in the size of LV
- #mount -a** ( if an entry is passed in **/etc/fstab** use this command), else do manual mounting as shown below
- #df -h**

```
[root@mlinux6 ~]# mount /dev/myvg/mylv /mydir
[root@mlinux6 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2        24G   6.8G   16G  30% /
tmpfs           491M    72K  491M   1% /dev/shm
/dev/sda1       190M   28M  153M  16% /boot
/dev/mapper/myvg-mylv
                  283M  2.1M  266M   1% /mydir
```

Note: xfs filesystem does not support file system reduction

### Moving or Migrating the data from one PV to another.

- There might be a situation where the **PV** might be failing and it is required to be replaced, in such case, we need to migrate or move the data from such **PV** to the other and isolate the **PV**.
- **The Steps to migrate the PV are**
  - Access the mount point of failing **PV** and check the data in it,
  - Verify the size of the **PV** by **pvs** command or **pvdisplay** command.
  - Unmount the file system on that **PV (optional)**
  - Add new **PV**, which should be of the same size or higher than that of the replacing **PV** to the volume group.
  - Migrate the **PVs** contents to the new **PV** using following command
  - **#pvmmove <Old PV> <New PV>**
  - Mount back the **LV**, access the mount point and verify the data in it.
  - Remove the faulty **PV** from Volume Group.

**Okay! So let's do the practical following above steps.**

- Access the mount point of the failing PV and check the data in it,

```
[root@mlinux6 ~]# cd /mydir
[root@mlinux6 mydir]# ls
lost+found
[root@mlinux6 mydir]# touch mydir{1..10}
[root@mlinux6 mydir]# ls
lost+found  mydir1  mydir10  mydir2  mydir3  mydir4  mydir5
[root@mlinux6 mydir]#
```

- Verify the size of the PV by **pvs** command or **pvdisplay** command.

```
[root@mlinux6 ~]# pvs
  PV          VG  Fmt Attr PSize   PFree
  /dev/sda6    myvg lvm2 a-- 516.00m 216.00m
  /dev/sda7    myvg lvm2 a-- 516.00m 516.00m
[root@mlinux6 ~]# pvdisplay
--- Physical volume ---
PV Name           /dev/sda6
VG Name           myvg
PV Size           517.69 MiB / not usable 1.69 MiB
Allocatable       yes
PE Size           4.00 MiB
Total PE          129
Free PE           54
Allocated PE      75
PV UUID           46Gvd5-z04y-tneS-S4v1-1ySh-TG8J-PiqnXy
```

- **Unmount the file system on that PV (this is optional as migration can be done online as well)**
- **#umount /mydir**

- Add new PV which should be of the same size or higher than that of the replacing PV to the volume group.
- In our case the size of the failing PV is around **500MB**, so we need to add a PV whose size is at least **500MB** or more
- I have created another partition from fdisk i.e. **/dev/sda7** with the size around **500MB**

```
[root@mlinux6 ~]# pvs
  PV          VG  Fmt  Attr PSize   PFree
  /dev/sda6   myvg lvm2 a--  516.00m 216.00m
  /dev/sda7           lvm2 ---  517.69m 517.69m
[root@mlinux6 ~]# vgextend myvg /dev/sda7
  Volume group "myvg" successfully extended
[root@mlinux6 ~]# pvs
  PV          VG  Fmt  Attr PSize   PFree
  /dev/sda6   myvg lvm2 a--  516.00m 216.00m
  /dev/sda7   myvg lvm2 a--  516.00m 516.00m
[root@mlinux6 ~]#
```

- Migrate the PV's contents to the new PV using following command
- #pvmove <Old PV> <New PV>
- #pvmove /dev/sda6 /dev/sda7

```
[root@mlinux6 ~]# pvmove /dev/sda6 /dev/sda7
  /dev/sda6: Moved: 4.0%
  /dev/sda6: Moved: 100.0%
```

- Mount back the LV, access the mount point and verify the data in it.

```
[root@mlinux6 ~]# mount -a
[root@mlinux6 ~]# cd /mydir
[root@mlinux6 mydir]# ls
lost+found mydir1 mydir10 mydir2 mydir3 mydir4 mydir5 mydir6 mydir7 mydir8 mydir9
[root@mlinux6 mydir]#
```

- Remove the free PV from Volume Group.
- As the data is moved safely, now let's remove the faulty PV from the volume group.
- The syntax to remove a PV from a VG is
- #vgreduce <vg name> <PV name>
- #vgreduce myvg /dev/sda6

```
[root@mlinux6 ~]# vgreduce myvg /dev/sda6
  Removed "/dev/sda6" from volume group "myvg"
[root@mlinux6 ~]# pvs
  PV          VG  Fmt  Attr PSize   PFree
  /dev/sda7           lvm2 ---  517.69m 517.69m
  /dev/sda6   myvg lvm2 a--  516.00m 216.00m
[root@mlinux6 ~]#
```

## Deleting/Removing the LV:

- To Delete/Remove an LV, first unmount the file system.
- Remove the entry from **/etc/fstab**.
- Use the command **lvremove** i.e.
- **#lvremove <LV name>**
- **#lvremove /dev/myvg/mylv** ( it will prompt to you to continue, press **y** to continue)
- Verify it by using **lvdisplay** command

```
[root@mlinux6 ~]# umount /mydir
[root@mlinux6 ~]# lvremove /dev/myvg/mylv
Do you really want to remove active logical volume mylv? [y/n]: y
Logical volume "mylv" successfully removed
[root@mlinux6 ~]# lvdisplay myvg
[root@mlinux6 ~]#
```

- As we was having only one LV and that is now deleted, that's why it is not showing any LVs after executing **lvdisplay** command.

## Deleting a Volume Group

- To delete the volume group, make sure that if there is any LV in it, it should not be mounted. Because while removing a vg it will also remove LV's inside it. In our case we have no LV in our volume group, so we will not be concerned about it.
- To delete a VG, use the following command.
- **#vgremove <vgname>**
- **#vgremove myvg**

```
[root@mlinux6 ~]# vgremove myvg
Volume group "myvg" successfully removed
[root@mlinux6 ~]# vgs
No volume groups found
```

## Deleting a Physical Volume

- Deleting a PV is very simple. The only thing we should check that the PV we are going to delete should not belong to any volume group. We can only delete a PV which is free.
- The syntax to delete a PV is
- **#pvremove <PV name>**
- **#pvremove /dev/sda6**
- **#pvremove /dev/sda7 OR**
- **#pvremove /dev/sda{6,7}** (To remove multiple PVs in one command)

```
[root@mlinux6 ~]# pvremove /dev/sda{6,7}
Labels on physical volume "/dev/sda6" successfully wiped
Labels on physical volume "/dev/sda7" successfully wiped
[root@mlinux6 ~]# pvs
```

- If you want you can verify it by using **pvs** or **pvdisplay** commands

***Building anything requires lots of concentration, hard work, and patience, but to destroy it, it is just a matter of seconds. Isn't it....!***

## Creating a VG by customized PE size

- To create a VG with specifying an PE size,
- First create a partition and also create a pv

```
[root@ktlinux ~]# fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000003d37

      Device Boot      Start        End    Blocks   Id  System
/dev/sda1            1       1785    14336000   8e  Linux LVM
/dev/sda2     *       1785       1811     204800   83  Linux
/dev/sda3       1811       2072    2097152   82  Linux swap / Solaris
/dev/sda4       2072       2610    4325849    5  Extended
/dev/sda5       2072       2136    518412+   8e  Linux LVM

[root@ linux ~]# pvcreate /dev/sda5
Physical volume "/dev/sda5" successfully created
```

- To create a vg with custom PE size use  
`#vgcreate <name for the vg> -s <size of PE( 1-128)> <pv names>`  
`#vgcreate myvg -s 16 /dev/sda5`

```
[root@mlinux7 ~]# vgcreate myvg -s 16 /dev/sda5
Volume group "myvg" successfully created
```

- Verify the PE size using `vgdisplay` command

```
[root@mlinux7 ~]# vgdisplay myvg
--- Volume group ---
VG Name           myvg
System ID
Format           lvm2
Metadata Areas   1
Metadata Sequence No  1
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size          496.00 MiB
PE Size          16.00 MiB
Total PE         31
Alloc PE / Size  0 / 0
Free  PE / Size  31 / 496.00 MiB
VG UUID          YyxEjM-JaVu-cy0X-2WTb-d2Ur-BRLb-PZsjSs
```

## Creating LV by specifying no. of LE instead of giving size in MB or GB.

- To create an LV using LE, the things to keep in mind are
- **Size of LE = Size of PE**
- **For example if the size of PE is 16, then the size of LE will also be 16.**

### Steps to create an LV based on LE

- The syntax to create an LV with no. of LE is  

```
#lvcreate -l <no. of LE> -n <name for the LV> <volume group name>
#lvcreate -l 25 -n mylv myvg
```

```
[root@mlinux7 ~]# lvcreate -l 25 -n mylv myvg
Logical volume "mylv" created.
```

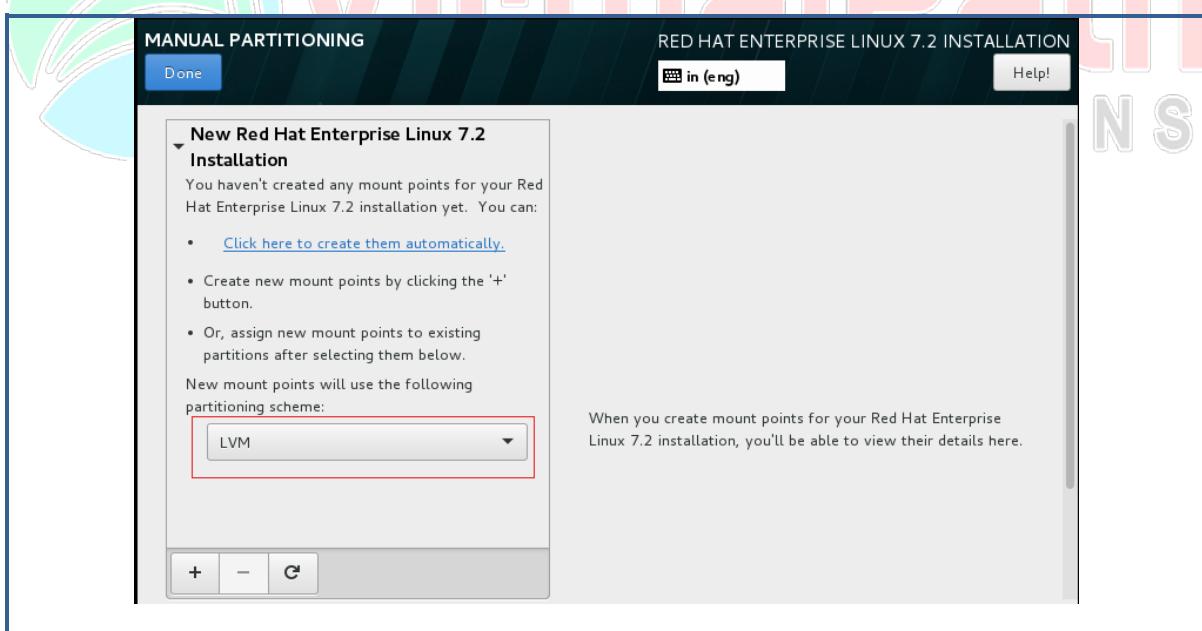
- Now check the size of the LV “mylv” using lvdisplay command  

```
#lvdisplay myvg
```

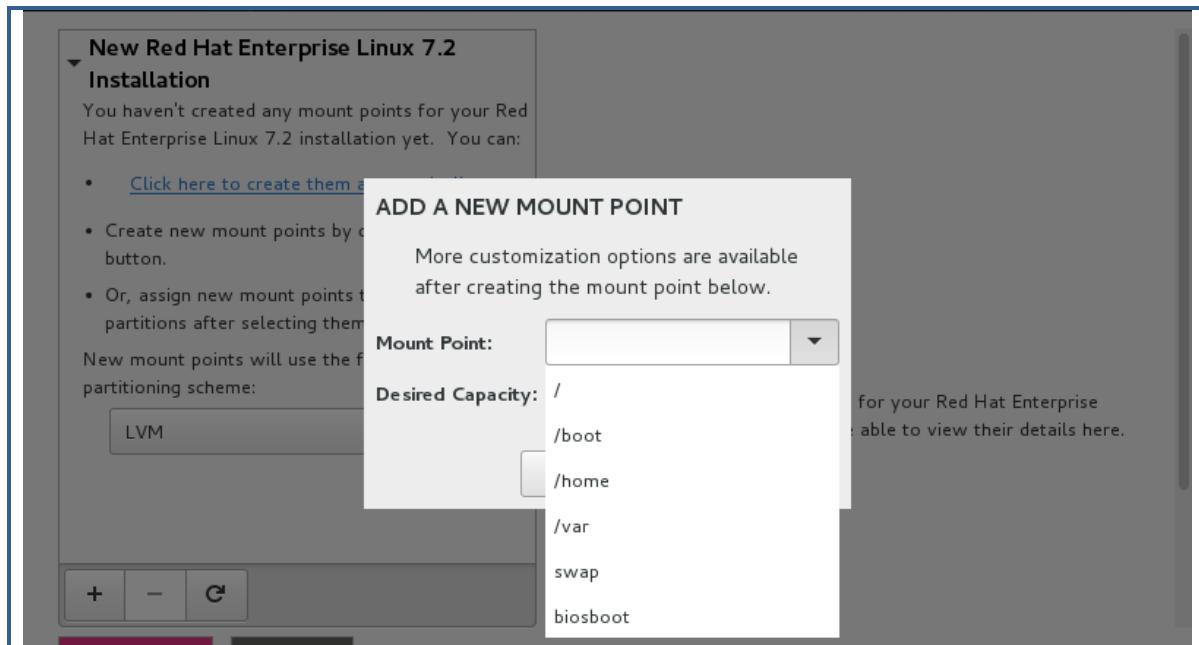
```
[root@mlinux7 ~]# lvdisplay myvg
--- Logical volume ---
LV Path          /dev/myvg/mylv
LV Name          mylv
VG Name          myvg
LV UUID          PYo57X-55qe-yRMu-dIso-rXX4-Mn0F-VZzYjA
LV Write Access  read/write
LV Creation host, time mlinux7.mb.com, 2017-02-15 14:45:38 +0530
LV Status        available
# open           0
LV Size          400.00 MiB
Current LE       25
Segments         1
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:0
```

## INSTALLING RHEL7/8 USING LVM PARTITIONING

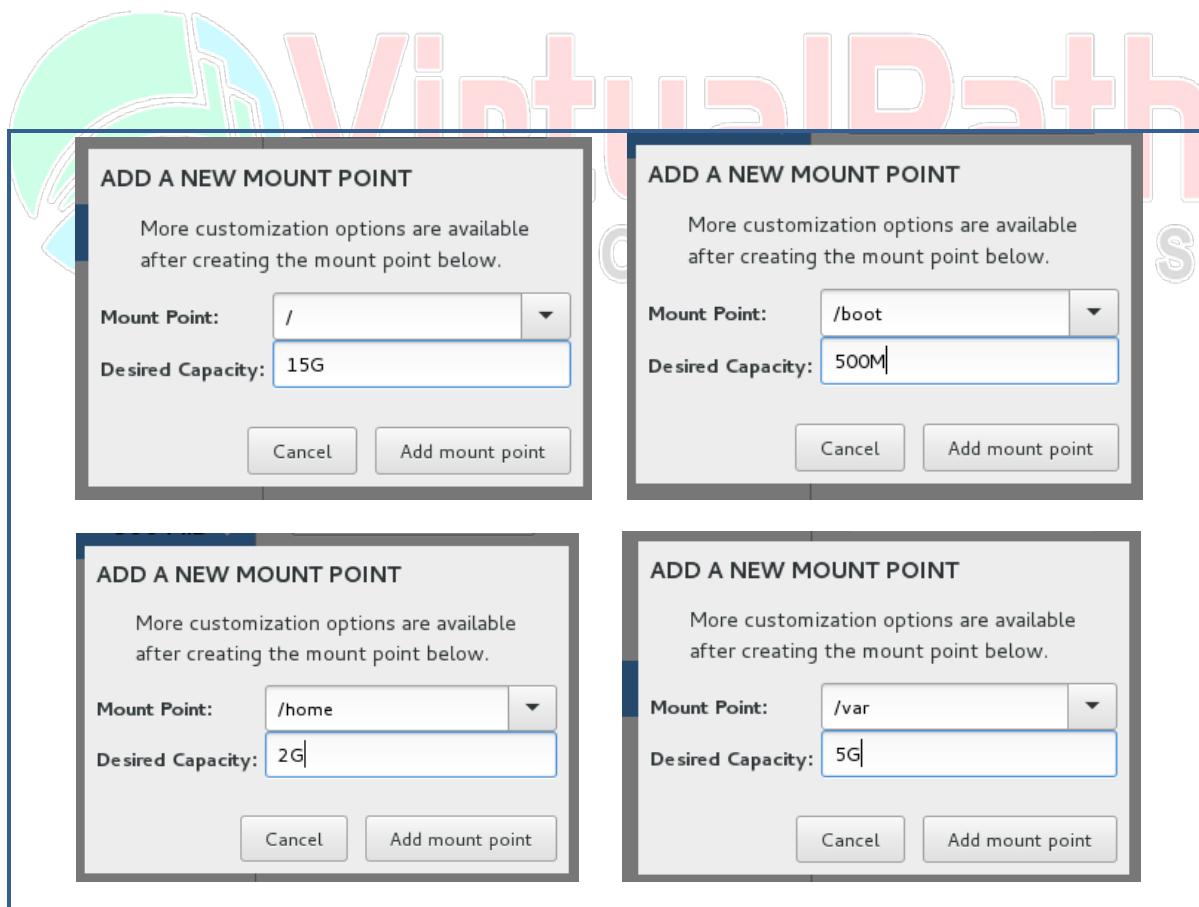
- The only difference in a normal installation and **LVM** installation is that instead of creating normal partition we will create a **VG** and then **LVs** for all partitions, except **/boot** and **swap**.
- The advantage of installing Linux using **LVM** is that, if any of system partition is running out of space and required more space, in case of normal partitioning it is not possible to increase the size of a partition once it is created. But, using LVM the space can be dynamically increased whenever it is required.
- Even if there is no space remaining in the disk some space can be borrowed from other **LVMs** and can easily be assigned to required system partition to fulfill its need.
- LVM** provides a greater scalability to the administrator and avoid uncertain down time to the server.
- LVM** ensures the possibility of increasing and decreasing the sizes whenever required and prevents unnecessary loss of time.
- Start the installation normally as done previously, but only at the time of partitioning follow the steps below.

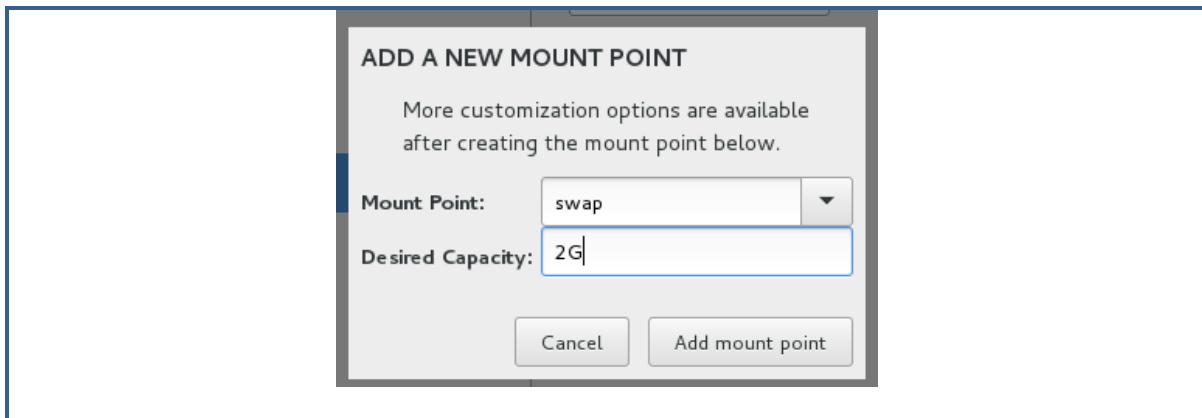


- Select the **LVM** scheme and click on **+** to proceed.



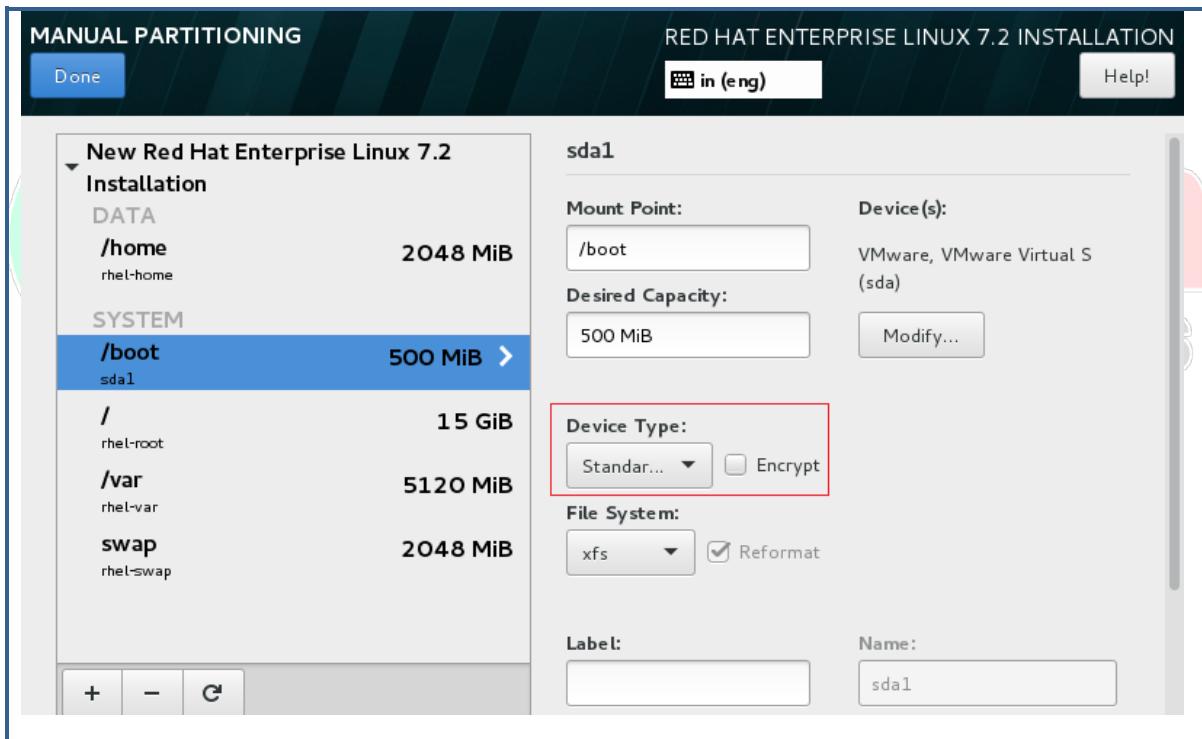
- Select an appropriate mount point and assign the size. Repeat it till all important mount points are created





- Give appropriate size and create /, /boot/, /home/, /var and swap

**Note:** All the sizes listed above are based on the availability of the space. It is no-where a recommended or minimum sizes. The sizes can be based on your requirements. But / should get more size as it will be having "/usr", "/opt" & "/tmp" as well, which in rhel6 used to be created separately.



- Verify the sizes and click on "**Done**" to continue with the installation. Complete the installation as usual as we have done previously at the beginning of the course.
- Note: /boot is automatically created as "**Standard Partition**" even though the scheme is selected as "**LVM**". Isn't it awesome?

**Practice the LVM Concept well; as it is the most important part in Linux and in any UNIX operating system as well.**

**That sums up the LVM concept in Linux**

## USER AND GROUP ADMINISTRATION

### PART- I USER ADMINISTRATION

In Linux/Unix user is one who uses the system. There can be at least one or more than one users in Linux at a time. Users on a system are identified by a username and a userid. The username is something that users would normally refer to, but as far as the operating system is concerned this is referred to using the user id (or uid). The username is typically a user friendly string, such as your name, whereas the user id is a number. The words username and userid are often (incorrectly) used interchangeably. The user id numbers should be unique (one number per user). If you had two usernames with the same user id, effectively there permissions would be the same and the files that they create would appear to have been created by the same user. This should not be allowed and the **useradd** command will not allow usernames to share the same userid.

#### Some Important Points related to Users:

- Users and groups are used to control access to files and resources
- Users login to the system by supplying their username and password
- Every file on the system is owned by a user and associated with a group
- Every process has an owner and group affiliation, and can only access the resources its owner or group can access.
- Every user of the system is assigned a unique user ID number ( the UID)
- Users name and UID are stored in **/etc/passwd**
- User's password is stored in **/etc/shadow** in encrypted form.
- Users are assigned a **home directory** and a program that is run when they login (**Usually a shell**)
- Users cannot read, write or execute each other's files without permission.

#### Types of users In Linux and their attributes:

TYPE	EXAMPLE	USER ID (UID)	GROUP ID (GID)	HOME DIRECTORY	SHELL
Super User	root	0	0	/root	/bin/bash
System User	ftp, ssh, apache nobody	1 to 999	1 to 999	/var/ftp, /var/www/html /var/named, etc.	/sbin/nologin
Normal User	Visitor, myuser,etc	1000 to 60000	1000 to 60000	/home/user name	/bin/bash

### In Linux there are three types of users.

#### **1. Super user or root user**

Super user or the root user is the most powerful user. He is the administrator user.

#### **2. System user**

System users are the users created by the software or applications. For example if we install Apache it will create a user apache. These kinds of users are known as system users.

#### **3. Normal user**

Normal users are the users created by root user. They are normal users like Rahul, Musab etc. Only the root user has the permission to create or remove a user.

### Whenever a user is created in Linux things created by default:-

- A home directory is created(/home/username)
- A mail box is created(/var/spool/mail/username)
- unique UID & GID are given to user

### Linux uses UPG (User Private Group) scheme

- It means that whenever a user is created it has its own private group
- For Example if a user is created with the name **musab**, then a primary group for that user will be **musab** only

### There are two important files a user administrator should be aware of

1. "/etc/passwd"
2. "/etc/shadow"

### Each of the above mentioned files have specific formats

#### **1. /etc/passwd**

```
[root@ linux ~]# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

The above fields are

- **root** =name
- **x**= link to password file i.e. /etc/shadow
- **0 or 1**= UID (user id)
- **0 or 1**=GID (group id)
- **root or bin** = comment (brief information about the user)
- **/root or /bin** = home directory of the user
- **/bin/bash or /sbin/nologin** = shell

## 2. /etc/shadow

```
root:$1fdgsdfsdksdkffefje:14757:0:99999:7:::
```

The fields are as follows,

1. **root** = User name
2. **:\$1fdgsdfsdksdkffefje** = Encrypted password
3. **14757** = Days since that password was last changed.
4. **0** = Days after which password must be changed.
5. **99999** = Days before password is to expire that user is warned.
6. **7** = Days after the password is expires that the user is disabled.
7. A reserved field.

## LAB WORK:-

### Creating a user

- The syntax for creating a user in Linux is
- # **useradd <username> <options>** or #**useradd <options> <username>**
- **options** are
- -u user id
- -G Secondary group id
- -g primary group id
- -d home directory
- -c comment
- -s shell

### Let's create a user with default attributes.

- When no option is used with **useradd** command the options like **UID**, **GID**, **home dir** and **shell** will be assigned default.
- #**useradd <username>**
- #**useradd myuser**

```
[root@mlinux7 ~]# useradd myuser
[root@mlinux7 ~]# tail /etc/passwd
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs/nobody
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:989:984::/run/gnome-inhibit
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd
tcpdump:x:72:72::/sbin/nologin
myuser:x:1000:1000::/home/myuser:/bin/bash
```

Observe that the uid, gid, home dir, and shell is assigned automatically.

### Let's create a user with customized attributes

- Create a user with following attributes
- Name = myuser2
- uid=1050
- comment =MGR
- shell = /bin/csh ( c shell)
- #useradd -u 1050 -c MGR -s /bin/csh myuser2 (rhel7)

```
[root@mlinux7 ~]# useradd -u 1050 -c MGR -s /bin/csh myuser2
[root@mlinux7 ~]# tail /etc/passwd
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:989:984::/run/gnome-initial-setup/:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/sbin/nologin
myuser:x:1000:1000::/home/myuser:/bin/bash
myuser2:x:1050:1050:MGR:/home/myuser2:/bin/csh
```

### Assigning password to the user:

- As a root user we can assign any password to any user
- The syntax for assigning a password is
- **#passwd** to assign password to current user ( the one with which you have logged in, if it is root then root's password will be changed)
- **#passwd <user name>** to assign a password to a specific user, only root can assign password to other user.

```
[root@mlinux7 ~]# passwd myuser
Changing password for user myuser.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

**Note:** If the “passwd” command is executed without any username, it would consider the Name of active or currently logged in user. Say, if you have logged in with root, it will try to change the root password

## Modifying the user's attribute

- After creating a user if we need to modify the attributes of user like changing uid, changing secondary group id or adding a comment, locking or unlocking the user account, can be done by following command
- Syntax. # usermod <options> <username>
- options are:
- all the options which are used with useradd command can be used and also the following,
- l to change login name
- L to LOCK account
- U to UNLOCK account
- ex. # usermod -l newname oldname (changing the name of the user)
- ex. # usermod -L username to lock the user account; #passwd -l username
- ex. # usermod -U username to unlock the user account; #passwd -u username
- Note: - when an account is locked it will show! (Exclamation mark) in /etc/shadow file.
- The account lock/unlock details can also be checked by "#passwd -S" command

### Locking and unlocking a user account:

- To lock a user a/c use the following
- #usermod -L < user name>; #passwd -l <user name>
- #usermod -L myusr; #passwd -l myuser
- Verify it in /etc/shadow file, it shows exclamation mark before user account or try using #passwd -S <user name>

```
[root@mlinux7 ~]# usermod -L myuser
[root@mlinux7 ~]# passwd -S myuser
myuser ! 2017-02-16 0 99999 7 -1 (Password locked.)
```

*OR*

```
[root@mlinux7 ~]# passwd -l myuser
Locking password for user myuser.
passwd: Success
[root@mlinux7 ~]# passwd -S myuser
myuser ! 2017-02-16 0 99999 7 -1 (Password locked.)
[root@mlinux7 ~]#
```

### "/etc/shadow" file status

```
[root@mlinux7 ~]# tail /etc/shadow |grep myuser
myuser:$!$6$70F3QNKR$dfCus1NqGy.eThCxfyYyicasCKGX6
:::
```

### Unlocking a user a/c:

- Unlock the above a/c
- #usermod -U < user name >; #passwd -u <user name>
- #usermod -U myuser ; #passwd -u myuser
- Verify it in /etc/shadow file, it shows exclamation mark before user account or try using #passwd -S username

```
[root@mlinux7 ~]# usermod -U myuser
[root@mlinux7 ~]# passwd -S myuser
myuser PS 2017-02-16 0 99999 7 -1 (Password set, SHA512 crypt.)
```

*OR*

```
[root@mlinux7 ~]# passwd -u myuser
Unlocking password for user myuser.
passwd: Success
[root@mlinux7 ~]# passwd -S myuser
myuser PS 2017-02-16 0 99999 7 -1 (Password set, SHA512 crypt.)
```

### “/etc/shadow” file status

```
[root@mlinux7 ~]# tail /etc/shadow |grep myuser
myuser:$6$70F3QNKR$dfCuslNqGy.eThCx fyYyicaSCkGX6
:::
```

- Observe in both pictures that once the account is unlocked the exclamation is gone.

### The password parameters

- For any user we can set the parameters for the password, like min and max password age, password expiration warnings and a/c expiration date etc.
- To view the advanced parameters of the user, use
- #chage -l < user name>
- #chage -l myuser

[root@mlinux7 ~]# chage -l myuser	
Last password change	: Feb 16, 2017
Password expires	: never
Password inactive	: never
Account expires	: never
Minimum number of days between password change	: 0
Maximum number of days between password change	: 99999
Number of days of warning before password expires	: 7

- **Last password change:** When the password was changed last time.
- **Password expires:** Password expiry date
- **Password inactive:** After password expiry grace period before the account gets locked.
- **Account expires:** Date on which the account expires.
- **Minimum number of days b/w password change:** Once the password is changed, it cannot be changed until a min period of specified date. [0] means never.
- **Max number of days b/w password change:** After changing the password how long it will be valid for.
- **Number of days of warning before password expires:** Start of warnings to change the password, no. of days before the password expires.

### Changing the password parameters:

- Changing of the password parameters can be done by two ways.
1. #chage <user name >
  2. #chage <option> <value> <username>
- Let's see the first method and then the other.
  - To set the password parameters of a user "myuser" to
    - Min password age : 1 days
    - Max password age: 7 days
    - Password expiration warnings: 2 days before password expires
    - Password inactive [-1]: 1 one day later the account will be locked, after password expiry.
    - A/C expiration date: 2017-03-31 (March 31<sup>st</sup> 2017)
  - #chage myuser

```
[root@mlinux7 ~]# chage myuser
Changing the aging information for myuser
Enter the new value, or press ENTER for the default

      Minimum Password Age [0]: 1
      Maximum Password Age [99999]: 7
      Last Password Change (YYYY-MM-DD) [2017-02-16]:
      Password Expiration Warning [7]: 2
      Password Inactive [-1]: 1
      Account Expiration Date (YYYY-MM-DD) [-1]: 2017-03-31

[root@mlinux7 ~]#
[root@mlinux7 ~]# chage -l myuser
Last password change : Feb 16, 2017
Password expires       : Feb 23, 2017
Password inactive      : Feb 24, 2017
Account expires        : Mar 31, 2017
Minimum number of days between password change : 1
Maximum number of days between password change : 7
Number of days of warning before password expires : 2
```

- The second method is for, if you want to change a particular field of password aging policy
- #chage <option> <value> <username>
- The options which can be used are as follows
  - -m for Min password age
  - -M for Max password age
  - -d for last time the password is changed. (*Note: if given d 0, it will force the user to change password at next login*)
  - -W Password expiration warnings
  - -I Password inactive [-1 means inactive].
  - -E A/C expiration date

- Let's see how to change only the account expiration date

```
[root@mlinux7 ~]# chage -E 2017-04-30 myuser
[root@mlinux7 ~]# chage -l myuser
Last password change : Feb 16, 2017
Password expires      : Feb 23, 2017
Password inactive     : Feb 24, 2017
Account expires       : Apr 30, 2017
Minimum number of days between password change : 1
Maximum number of days between password change : 7
Number of days of warning before password expires : 2
```

Likewise you can use any option listed above and change any particular field in password aging parameters.

#### Forcing a user to change the password at next login:

Sometimes it is required to force the users to change their password at next login. This can be done using following syntax

- #chage -d 0 <username>, (where 0 = zero days since last password change)
- #chage -d 0 myuser

```
[root@mlinux7 ~]# chage -d 0 myuser
[root@mlinux7 ~]# chage -l myuser
Last password change : password must be changed
Password expires      : password must be changed
Password inactive     : password must be changed
Account expires        : Apr 30, 2017
Minimum number of days between password change : 1
Maximum number of days between password change : 7
Number of days of warning before password expires : 2
```

*At next login*

login as: myuser  
myuser@192.168.100.20's password:  
You are required to change your password immediately (root enforced)  
Last failed login: Tue Feb 14 14:46:17 IST 2017 from 192.168.100.3 on ssh:notty  
There was 1 failed login attempt since the last successful login.  
WARNING: Your password has expired.  
You must change your password now and login again!  
Changing password for user myuser.  
Changing password for myuser.

#### Deleting a User:

- To delete a user the syntax used is
- #userdel <username> it will only delete the user but home directory will be there. To delete the user with its home directory and mailbox, use the following command.
- #userdel -r < user name >
- #userdel -r myuser2

```
[root@mlinux7 ~]# userdel -r myuser2
[root@mlinux7 ~]# cd /home
[root@mlinux7 home]# ls
myuser
[root@mlinux7 home]# cd /var/spool/mail/
[root@mlinux7 mail]# ls
myuser  rpc
```

*We're now done with user administration, let's see what's in part-II*

## PART-II GROUP ADMINISTRATION

### GROUPS

- Users are assigned to groups with unique group ID numbers (the GID)
- The group name and GID are stored in **/etc/group**
- Each user is given their own private group
- They can also be added to their groups to gain additional access
- All users in a group can share files that belong to the group

Each user is a member of at least one group, called a primary group. In addition, a user can be a member of an unlimited number of secondary groups. Group membership can be used to control the files that a user can read and edit. For example, if two users are working on the same project you might put them in the same group so they can edit a particular file that other users cannot access.

- A user's primary group is defined in the **/etc/passwd** file and Secondary groups are defined in the **/etc/group** file.
- The primary group is important because files created by this user will inherit that group affiliation.

#### Creating a Group with default options :

- To create a group the syntax is
- **#groupadd <name for the group>**
- **#groupadd mygroup**

```
[root@mlinux7 ~]# groupadd mygrp
[root@mlinux7 ~]# tail /etc/group
postdrop:x:90:
postfix:x:89:
ntp:x:38:
sshd:x:74:
tcpdump:x:72:
stapusr:x:156:
stapsys:x:157:
stapdev:x:158:
myuser:x:1000:
mygrp:x:1001:
```

### Creating a group with user specified group id (GID)

- **#groupadd <option> <name for the group>**
- **#groupadd -g 1050 mygrp2**
- Verify it in /etc/group

```
[root@mlinux7 ~]# groupadd -g 1050 mygrp2
[root@mlinux7 ~]# tail /etc/group
postfix:x:89:
ntp:x:38:
sshd:x:74:
tcpdump:x:72:
stapusr:x:156:
stapsys:x:157:
stapdev:x:158:
myuser:x:1000:
mygrp:x:1001:
mygrp2:x:1050:
```

### Modifying the properties of the group

- To modify the group properties the syntax is
- **#groupmod <option> <arguments> <group name>**
- The options are
- -g to change the group id
- -o to override the previous assigned id, if it matches with the new one.
- -n to change the group name

### Changing the GID of the group

- **#groupmod -g 1100 mygrp**
- Verify it in /etc/group

```
[root@mlinux7 ~]# groupmod -g 1100 mygrp
[root@mlinux7 ~]# tail /etc/group
postfix:x:89:
ntp:x:38:
sshd:x:74:
tcpdump:x:72:
stapusr:x:156:
stapsys:x:157:
stapdev:x:158:
myuser:x:1000:
mygrp:x:1100:
mygrp2:x:1050:
```

### Changing the name of the group

- The syntax for changing the group name is
- **#groupmod -n <new name > < existing name >**
- **#groupmod -n mygrp mygroup**

```
[root@mlinux7 ~]# groupmod -n mygroup mygrp
[root@mlinux7 ~]# grep mygroup /etc/group
mygroup:x:1100:
```

## Adding and Removing Members to a Group

- Adding single or multiple users to the group with various attributes
- #gpasswd < option> <arguments> <group name>
- Options:
  - -M For Adding Multiple users to a group
  - -a for Adding a single user to a group
  - -A for Adding a group Administrator
  - -d removing a user from a group
- #gpasswd -M <user>,<user>,<user> <group>
- #gpasswd -M u1,u2,u3 mygroup

```
[root@mlinux7 ~]# gpasswd -M u1,u2,u3 mygroup
[root@mlinux7 ~]# tail /etc/group
tcpdump:x:72:
stapusr:x:156:
stapsys:x:157:
stapdev:x:158:
myuser:x:1000:
mygrp2:x:1050:
mygroup:x:1100:u1,u2,u3
```

### Adding a single user using gpasswd

- #gpasswd -a u4 mygroup (verify it in /etc/group)

```
[root@mlinux7 ~]# useradd u4
[root@mlinux7 ~]# gpasswd -a u4 mygroup
Adding user u4 to group mygroup
[root@mlinux7 ~]# grep mygroup /etc/group
mygroup:x:1100:u1,u2,u3,u4
[root@mlinux7 ~]#
```

### Making a user as an administrator of the group

- #gpasswd -A u1 mygroup (verify it in /etc/gshadow)

```
[root@mlinux7 ~]# grep mygroup /etc/gshadow
mygroup:!:u1,u2,u3,u4
[root@mlinux7 ~]# gpasswd -A u1 mygroup
[root@mlinux7 ~]# grep mygroup /etc/gshadow
mygroup:!:u1:u1,u2,u3,u4
[root@mlinux7 ~]#
```

### Removing a user from the group

- #gpasswd -d u2 mygroup

```
[root@mlinux7 ~]# grep mygroup /etc/group
mygroup:x:1100:u1,u2,u3,u4
[root@mlinux7 ~]# gpasswd -d u2 mygroup
Removing user u2 from group mygroup
[root@mlinux7 ~]# grep mygroup /etc/group
mygroup:x:1100:u1,u3,u4
```

### Removing a group

- #groupdel <group name>; #groupdel mygroup

## CONTROLLING ACCESS TO FILES

In this chapter we will be dealing with two things.

1. Special Permissions or Advanced Permission
2. Access Control List (ACL)

Let's first begin with Special Permissions

### **1. Special Permissions or Advanced Permission**

- There are three special permissions that can be assigned to a file or directory apart from basic file permissions(rwx), they are
- **SUID – SET USER ID**
- **SGID – SET GROUP ID**
- **STICKY BIT**

Permission	Symbolic Form	Numeric Form	Syntax
SETUID	s or S	4	#chmod u+s or #chmod 4766
SETGID	s or S	2	#chmod g+s or #chmod 2766
STICKY BIT	t or T	1	#chmod o+t or #chmod 1766

**Note:** Where **s**= **setuid + execute** permission and **S**= **setuid only**. Same is for **SGID** and also for sticky bit.

#### **SUID – SET USER ID**

Change user ID on execution. If SETUID bit is set, when the file will be executed by a user, the process will have the same rights as the owner of the file being executed. Many of the system commands are the best example for SUID, basically the owner of the commands will be **root**, but still a normal user can execute it.

#### **Example**

- By default passwd command is having uid, so all users can run that command but if uid is removed and a normal user wants to user execute it, then they would not be able to use it to update /etc/shadow with new passwd.

```
[root@client ~]# which passwd
/usr/bin/passwd
[root@client ~]# ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832 Jan 29 2014 /usr/bin/passwd
```

**Note:** observe that in the permissions “**-rwsr-xr-x**” it contains an “**s**”, which means SUID is placed.

- Let's remove uid on passwd command and logged in as normal user and check the results

```
[root@client ~]# chmod u-s /usr/bin/passwd
[root@client ~]# ls -l /usr/bin/passwd
-rwxr-xr-x. 1 root root 27832 Jan 29 2014 /usr/bin/passwd
[root@client ~]# su - myuser
Last login: Sun Jan 13 08:16:01 EST 2019 on pts/0
[myuser@client ~]$ passwd
Changing password for user myuser.
Changing password for myuser.
(current) UNIX password:
New password:
Retype new password:
passwd: Authentication token manipulation error
```

### SGID – SET GROUP ID

Set group ID, used on executable files to allow the file to be run as if logged into the group (like SUID but uses file group permissions)

SGID can also be used on a directory so that every file created in that directory will have the directory group owner rather than the group owner of the user creating the file.

#### Example

- When a directory is created and its group is set to some group. Now if SGID is applied to it, and the group member creates files and directory inside it, then it will get the same group rather than getting user's primary group
- Let's see it practically.

```
[root@mlinux7 ~]# mkdir mydir
[root@mlinux7 ~]# chgrp mygroup mydir
[root@mlinux7 ~]# ls -ld mydir
drwxr-xr-x. 2 root mygroup 6 Feb 16 16:25 mydir
[root@mlinux7 ~]# chmod 777 mydir
[root@mlinux7 ~]# ls -ld mydir
drwxrwxrwx. 2 root mygroup 6 Feb 16 16:25 mydir
```

- Login as other user, access the directory, create some files and check the group it is getting. ( It will be getting the logged in user's group)

```
[u1@mlinux7 ~]$ whoami
u1
[u1@mlinux7 ~]$ cd /mydir
[u1@mlinux7 mydir]$ touch f1
[u1@mlinux7 mydir]$ ll
total 0
-rw-rw-r--. 1 u1 u1 0 Feb 16 16:38 f1
[u1@mlinux7 mydir]$
```

- Now, as a root, assign SGID on the directory
- #chmod g+s /mydir

```
[root@mlinux7 ~]# chmod g+s /mydir
[root@mlinux7 ~]# ls -ld /mydir
drwxrwsrwx. 2 root mygroup 15 Feb 16 16:38 /mydir
```

- Try creating other files with other user(s) in the same directory, instead of getting it's own group it would now be inheriting directory's group

```
[u1@mlinux7 mydir]$ whoami
u1
[u1@mlinux7 mydir]$ touch f2 f3
[u1@mlinux7 mydir]$ ll
total 0
-rw-rw-r--. 1 u1 u1 0 Feb 16 16:38 f1
-rw-rw-r--. 1 u1 mygroup 0 Feb 16 16:51 f2
-rw-rw-r--. 1 u1 mygroup 0 Feb 16 16:51 f3
```

## STICKY BIT

If sticky bit is applied on a file or directory, then only root and owner of that file or directory can delete it. Even if others are having full permissions they cannot delete or edit the contents of the directory

- Let see it practically.
- Apply sticky bit to the directory

```
[root@mlinux7 /]# ls -ld /mydir  
drwxrwsrwx. 2 root mygroup 33 Feb 16 16:51 /mydir  
[root@mlinux7 /]# chmod o+t /mydir  
[root@mlinux7 /]# ls -ld /mydir  
drwxrwsrwt. 2 root mygroup 33 Feb 16 16:51 /mydir
```

- Access the directory with other user and try deleting the files

```
[u2@mlinux7 ~]$  
[u2@mlinux7 ~]$ whoami  
u2  
[u2@mlinux7 ~]$ cd /mydir  
[u2@mlinux7 mydir]$ ll  
total 0  
-rw-rw-r--. 1 u1 u1 0 Feb 16 16:38 f1  
-rw-rw-r--. 1 u1 mygroup 0 Feb 16 16:51 f2  
-rw-rw-r--. 1 u1 mygroup 0 Feb 16 16:51 f3  
[u2@mlinux7 mydir]$ rm -f f1  
rm: cannot remove 'f1': Operation not permitted  
[u2@mlinux7 mydir]$ rm -f f2  
rm: cannot remove 'f2': Operation not permitted  
[u2@mlinux7 mydir]$ rm -f f3  
rm: cannot remove 'f3': Operation not permitted
```

Note: If the owner or root tries to modify or delete the contents it would be allowed

## 2. Access Control List (ACL)

- Define more fine-grained discretionary access rights for files and directories.
- Often, you want to share files among certain groups and specific users. It is a good practice to designate a directory for that purpose. You want to allow those groups and users to read, and write files in that directory, as well as create new files into the directory. Such special permissions can be given using ACL.

**To check the acl permission syntax is:**

- `#getfacl <option> <dir/file name>`
- **Options:**
- `-d` Displays the default ACL
- `-R` Recurses into subdirectories

`#getfacl /mydir`

```
[root@mlinux6 ~]# ls -ld /mydir
drwxr-xr-x. 3 root root 4096 Feb 16 21:38 /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
```

- Now let's assign full permission to the directory and then apply acl on it, so that we can analyze how acl will work.

```
[root@mlinux6 ~]# chmod 777 /mydir
[root@mlinux6 ~]# ls -ld /mydir
drwxrwxrwx. 3 root root 4096 Feb 16 21:38 /mydir
[root@mlinux6 ~]#
```

- Okay, now we are ready to apply acl, but first lets understand the command and option in details.
- The syntax to apply acl is
- `#setfacl <option> < argument > < file or directory name >`
- **The options are,**
- `-m` Modifies an ACL
- `-x` Removes an user/group from ACL
- `-R` Recurses into subdirectories
- `-b` completely banishing/removing the ACL from a file/directory

**The possible arguments are**

- `u: user`
- `g: group`

To assign **read and execute** permission to a particular user the syntax could be

- `#setfacl -m u: <username>: <permissions> <file or dir name>`
- `#setfacl -m u1:rx /mydir`
- Verify it by using `getfacl` command

```
[root@mlinux6 ~]# setfacl -m u1:rx /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
user:u1:r-x
group::rwx
mask::rwx
other::rwx
```

- Now login as u1 user and try to create a file inside /mydir, as we have not assigned write permission to u1 user, though it is having full permissions, still it will not allow u1 user to create a file inside it.

```
[root@mlinux6 ~]# su - u1
[u1@mlinux6 ~]$ ls -ld /mydir
drwxrwxrwx+ 3 root root 4096 Feb 16 21:38 /mydir
[u1@mlinux6 ~]$ cd /mydir
[u1@mlinux6 mydir]$ touch f1
touch: cannot touch `f1': Permission denied
[u1@mlinux6 mydir]$
```

Observe that when you check for the permissions it is showing a + sign after normal permission, that indicate that ACL is applied on this directory.

**To assign read write and execute permission to a particular group**

- `#setfacl -m g:<group name>:<permissions> <file or directory name>`
- `#setfacl -m g:g1:rwx /mydir`

```
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
user:u1:r-x
group::rwx
group:g1:rwx
mask::rwx
other::rwx
```

Now you know how to apply acl on any file or directory, let me just give one more examples which you can broaden your understandings.

Assigning read and execute permission for a user and a group at same time.

- #setfacl -m u:u1:rx,g:g1:rx /mydir

```
[root@mlinux6 ~]# setfacl -m u:u1:rx,g:g1:rx /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
user:u1:r-x
group::rwx
group:g1:r-x
mask::rwx
other::rwx
```

Likewise you can explore applying acl to any user, group, or others in many ways.

**Removing acl for a particular user**

- #setfacl -x u:<username> <dir name>
- #setfacl -x u1 /mydir; #setfacl -x u:u1 /mydir

```
[root@mlinux6 ~]# setfacl -x u1 /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
group:g1:r-x
mask::rwx
other::rwx
```

**Removing acl for a particular group**

- #setfacl -x g:<group name> <directory name>
- #setfacl -x g:g1 /mydir

```
[root@mlinux6 ~]# setfacl -x g:g1 /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
mask::rwx
other::rwx
```

### Removing all ACL permissions from a file or directory

- #setfacl -b <dir name>
- #setfacl -b /ktdir

As we have removed acl for a group and a user, let's apply back some acl on **ktdir** and remove it using above command

```
[root@mlinux6 ~]# setfacl -m u:u1:rwx,u:u2:rw,u:u3:r,g:g1:rwx,g:g2:rwx,g:g3:--- /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
user:u1:rwx
user:u2:rw-
user:u3:r--
group::rwx
group:g1:rwx
group:g2:r-x
group:g3:---
mask::rwx
other::rwx

[root@mlinux6 ~]# setfacl -b /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
other::rwx
```

### To Apply /Remove ACL recursively on a directory and its contents

- #setfacl -Rm u:u1:rwx,g:g1:rw /mydir
- #setfacl -Rb /mydir (To recursively banishing ACL from directory and its contents)

```
[root@mlinux6 /]# setfacl -Rm u:u1:rwx,g:g1:rw /mydir
[root@mlinux6 /]# ls -l /mydir
total 28
-rw-rwrxr--+ 1 root root      0 Feb 16 22:22 f1
-rw-rwrxr--+ 1 root root      0 Feb 16 22:22 f2
-rw-rwrxr--+ 1 root root      0 Feb 16 22:22 f3
drwxrwxrwx---+ 2 root root 16384 Feb 16 21:38 lost+found
```

- To remove a user or group from acl
- #setfacl -Rx g:g1 /mydir (use #setfacl -Rx u1 /mydir for a user)

```
[root@mlinux6 ~]# setfacl -Rx u1 /mydir
[root@mlinux6 ~]# getfacl -R /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
group:g1:rwx-
mask::rwx
other::rwx

# file: mydir/lost+found
# owner: root
# group: root
user::rwx
group::---
group:g1:rwx-
mask::rwx-
other::---
```

#### To remove ACL completely from a directory recursively

- #setfacl -Rb /mydir

```
[root@mlinux6 ~]# setfacl -Rb /mydir
[root@mlinux6 ~]# ls -l /mydir
total 16
-rw-r--r--. 1 root root      0 Feb 16 22:22 f1
-rw-r--r--. 1 root root      0 Feb 16 22:22 f2
-rw-r--r--. 1 root root      0 Feb 16 22:22 f3
drwx-----. 2 root root 16384 Feb 16 21:38 lost+found
[root@mlinux6 ~]# getfacl -R /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
other::rwx
```

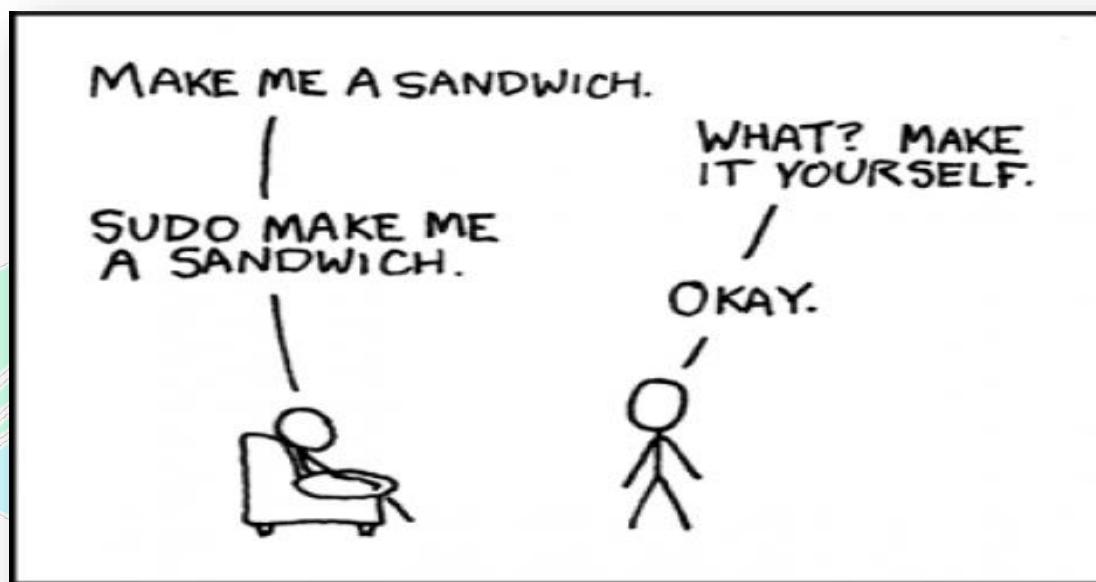


*There are still much more experiments can be done, go ahead and read man pages for more details*

## ENHANCED USER SECURITY WITH SUDO

### SUDO

- Sudo stands for either "substitute user do" or "super user do" (depending upon how you want to look at it). What sudo does is incredibly important and crucial to many Linux distributions. Effectively, sudo allows a user to run a program as another user (most often the root user). There are many that think sudo is the best way to achieve "best practice security" on Linux
- Users can login using their username and password and can issue administrative commands placing sudo in front of the commands, e.g. sudo rpm -Uvh \*.rpm , to run the command which installs and updates programs in Linux (rpm).



- The file **/etc/sudoers** file has the rules that users have to follow when using sudo command. That means that whatever commands access is provided to any user in **/etc/sudoers** file, that user can only run those commands.
- Do not edit the **/etc/sudoers** directly; instead use "**visudo**" command to edit the sudoers file. There are two reasons for that- it prevents two users from editing the file at the same time, and it also provides limited syntax checking. Even if you are the only root user, you need the syntax checking, so use "visudo".

### Advantages of using SUDO

**Two of the best advantages about using sudo are:**

- **Limited user privileges**

As we have studied above that we can restrict users to use certain commands as a privileged user as per the role of the user.

**E.g.:** Networking commands for Network user and Admin commands for Admin users etc.

- **Logs of the actions done by users**

All commands executed by sudo users will be stored in **/var/log/secure** file, but still if you want you can make your own log file by passing an entry in **/etc/sudoers** file at the bottom as “**Defaults logfile=/var/log/sudo.log**” or whatever name you want, to save the logs of what commands is executed by which sudo user.

### The /etc/sudoers file

- As we learnt above that it is the configuration file for sudo users, which is used to assign specific commands to the specific users or groups.
- Always use **visudo** command to edit this file. it prevents two users from editing the file at the same time, and it also provides limited syntax checking .
- When you run **visudo** command the output will be as follows

```
## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##       user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
```

- As you can see there is basically one line
- **root ALL=(ALL) ALL**
- This lines means that the user root can execute from ALL terminals, acting as ALL (any) users, and run ALL (any) command.
- So the first part is the **user**, the second is the **terminal** from where the user can use sudo, the third is **as which user he may act**, and the last one, is which **commands** he may run.
- The advantage of **visudo** command , while editing if there are any syntax error it will be reflected as follows

```
[root@ cl5 ~]#
[root@ cl5 ~]# visudo
>>> /etc/sudoers: [syntax error near line 93] <<<
visudo: Warning: unused User_Alias   ADMIN
What now? ■
```

## LAB WORK:-

Allow a user "myuser" all privileges like root

- To assign root privileges to user add a line by using sudoers file as shown below.  
**#visudo** (save the sudoers file as we save a vim file using "**wq!**")

```
##      user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root  ALL=(ALL)      ALL
myuser  ALL=(ALL)      ALL
```

- Now logged in as myuser and run admin commands like **fdisk -l** etc
- First try to run fdisk command normally and see what happens.

```
[root@ cl5 ~]# su - myuser
[myuser@ cl5 ~]$ fdisk -l
[myuser@ cl5 ~]$ fdisk /dev/sda

Unable to open /dev/sda
[myuser@ cl5 ~]$
```

**It will not allow a normal user to run privileged user's command**

- Now run the same command using **sudo** before command  
**#sudo fdisk -l** (or) **#sudo fdisk /dev/sda**

```
[myuser@ cl5 ~]$ sudo fdisk -l
[sudo] password for myuser:

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0002e9e6

Device Boot      Start        End      Blocks   Id
/dev/sda1            1       1785    14336000   8e
/dev/sda2      *        1785       1811      204800   83
/dev/sda3            1811       1941     1048576   82
```

**Note:** Only for the first time of the session it will prompt for user's password to continue, but for rest of the process it will continue normally as shown below

```
[myuser@ cl5 ~]$ sudo fdisk /dev/sda
```

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to switch off the mode (command 'c') and change display units to sectors (command 'u').

Command (m for help):

### Allow a group called mygroup, all root privileges.

- Let's first check the members of mygroup and then apply root privileges.

```
#tail /etc/gshadow
```

```
[root@ cl5 ~]# tail /etc/gshadow
fuse:::
stapdev:::
stapusr:::
gdm:::
student::::
myuser:::
mygroup:::musab,rahul,sai
musab:::
rahul:::
sai:::
[root@ cl5 ~]#
```

- Okay as we know the users in mygroup, let's assign it root privileges.

```
#visudo and look for the below line.
```

```
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING,
## Allows people in group wheel to run all commands
# %wheel      ALL=(ALL)      ALL
%mygroup      ALL=(ALL)      ALL
```

- Now, login as one of the user of mygroup try root commands

```
[root@ cl5 ~]# su - musab
[musab@ cl5 ~]$ sudo parted -l
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
[sudo] password for musab:
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system    Flags
 1      1049kB  14.7GB  14.7GB  primary               lvm
 2      14.7GB   14.9GB  210MB   primary    ext4          boot
 3      14.9GB   16.0GB  1074MB  primary   linux-swap(v1)
```

**Allow a user “myuser2” to run all commands without prompting for his password any time.**

- To allow run all commands, the syntax we have already seen, but allow him run command's without prompting password a small change is to be made,

```
## Allows people in group wheel to run all commands
# %wheel      ALL=(ALL)      ALL
# %mygroup    ALL=(ALL)      ALL
## Same thing without a password
# %wheel      ALL=(ALL)      NOPASSWD: ALL
myuser2     ALL=(ALL)      NOPASSWD: ALL
```

- Now login as that user and check whether password is prompted or not

```
[root@ cl5 ~]# su - myuser2
[myuser2@ cl5 ~]$ sudo parted -l
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system    Flags
 1      1049kB  14.7GB  14.7GB  primary   ext4          lvm
 2      14.7GB  14.9GB  210MB   primary   ext4          boot
 3      14.9GB  16.0GB  1074MB  primary   linux-swap(v1)
```

Note: - The same can be done for groups also, try it!

**Restrict a user “myuser” to run only two root commands.**

- This task is very simple; just modify the previous permissions assign to myuser.
- Let's give myuser to run only **#fdisk** and **#parted** command access.
- First check the complete path of those command by using following command

**#which fdisk**

**#which parted**

```
[root@ cl5 ~]# which fdisk
/sbin/fdisk
[root@ cl5 ~]# which parted
/sbin/parted
[root@ cl5 ~]#
```

- Let's assign both above paths in sudoers file

**#visudo**

```
## Syntax:
##
##      user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)      ALL
myuser  ALL=(ALL)      /sbin/fdisk, /sbin/parted
```

- Login as myuser and try assigned commands and other commands as well

```
[root@ cl5 ~]# su - myuser
[myuser@ cl5 ~]$ sudo fdisk -l
[sudo] password for myuser:

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

[myuser@ cl5 ~]$ sudo parted -l
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
[myuser@ cl5 ~]$ sudo useradd maarij
Sorry, user myuser is not allowed to execute '/usr/sbin/useradd maarij' as root
on cl5.my.com.
[myuser@ktcl5 ~]$
```

**Note:** - Try the same for groups also. It is exactly same

**Allow a group “mygroup” to run only network related commands as sudo user**

- To allow a group run only network commands, first uncomment the following line

```
## Networking
# Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient, /usr/bin/net,
sbin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig, /sbin/mii-tool

## Networking
Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient, /usr/bin/net,
bin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig, /sbin/mii-tool
```

Observe that we have just remove ‘#’ before the line to make the line readable. And also observe that it contains all networking commands.

- Just replace “**ALL**” with “**NETWORKING**” from the last field of mygroup line.

```
## Allows people in group wheel to run all commands
# %wheel        ALL=(ALL)          ALL
%mygroup        ALL=(ALL)          NETWORKING
```

**NOTE:** - **NETWORKING** is the name of the command alias, which was uncommented line.

- Now login as one of the member of mygroup and try some commands assigned it.

```
[root@ cl5 ~]# su - rahul
[rahul@ cl5 ~]$ sudo fdisk -l
[sudo] password for rahul:
Sorry, user rahul is not allowed to execute '/sbin/fdisk -l' as root on cl5.my.com.
[rahul@ cl5 ~]$ sudo route
[sudo] password for rahul:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.10.0   *               255.255.255.0   U     0      0        0 eth0
link-local      *               255.255.0.0    U     1002   0        0 eth0
```

### Create a customize commands alias and assign it to mygroup with network command.

- Okay, first we need to create an alias say "CUSTOM" with some commands and assign it to mygroup in addition to NETWORK commands.
- Let's firs get the path of the command need to be in CUSTOM alias

```
[root@ cl5 ~]# which service
/sbin/service
[root@ cl5 ~]# which rpm
/bin/rpm
[root@ cl5 ~]# which yum
/usr/bin/yum
[root@ cl5 ~]#
```

- Okay, now let's create an alias for these commands and assign it to mygroup
- **#visudo**

```
## Networking
Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping,
/bin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig,
## Custom
Cmnd_Alias CUSTOM = /sbin/service, /bin/rpm, /usr/bin/yum
```

- What are you waiting for! Assign it to ktgroup and save the file.

```
## Allows people in group wheel to run all commands
# %wheel      ALL=(ALL)        ALL
%mygroup      ALL=(ALL)        NETWORKING, CUSTOM
```

- Login as one of the users in mygroup and try newly added commands.

```
[root@ tcl5 ~]# su - rahul
[rahul@ cl5 ~]$ sudo rpm -q vsftpd
vsftpd-2.2.2-6.el6.i686
[rahul@ cl5 ~]$ sudo yum list installed |grep -i vsftpd
This system is not registered with RHN.
RHN support will be disabled.
vsftpd.i686                               2.2.2-6.el6
                                         enterpriseLinux-201009221732.i386/6.0
```

### Allowing user all commands except few commands

- Most of the time it might require to block few commands from users, even though all commands have been granted. It is an important aspect as per the security view.
- Let's assume we want to block "**visudo**" and "**su**" commands
- Find out the path of those two command needs to be block

```
[root@mlinux7 ~]# which su
/usr/bin/su
[root@mlinux7 ~]# which visudo
/usr/sbin/visudo
[root@mlinux7 ~]#
```

- Now, allow all the commands to myuser and block the above two by using an “!” exclamation mark

```
## Allow root to run any commands anywhere
root    ALL=(ALL)          ALL
myuser  ALL=(ALL)          ALL, !/usr/bin/su, !/usr/sbin/visudo
```

- Login as myuser, try other commands and then go for blocked one and check whether they are allowed or not

```
[root@mlinux7 ~]# su - myuser
Last login: Fri Feb 24 23:05:16 IST 2017 on pts/0
[myuser@mlinux7 ~]$ sudo route
[sudo] password for myuser:
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
192.168.100.0   0.0.0.0        255.255.255.0  U      100    0        0 eno16777736
192.168.122.0   0.0.0.0        255.255.255.0  U      0       0        0 virbr0
[myuser@mlinux7 ~]$ sudo whoami
root
[myuser@mlinux7 ~]$ sudo who am i
root      pts/1        2017-02-24 23:41  (192.168.100.3)
```

#### Now let's try the blocked ones:

```
[myuser@mlinux7 ~]$ sudo su -
Sorry, user myuser is not allowed to execute '/bin/su -' as root on mlinux7.mb.com.
[myuser@mlinux7 ~]$ sudo visudo
Sorry, user myuser is not allowed to execute '/sbin/visudo' as root on mlinux7.mb.com.
[myuser@mlinux7 ~]$
```

**Note:** As a user to know how many commands are allowed via sudo, login as a user and run #sudo -l command

**Note: Checkout sudoers file for more option and try it out on your own!!!!**

## Network Configuration & Troubleshooting

### **Networking:**

It is a connection between two or more machines to communicate with each other.

**The basic requirements for Networking are:**

1. **NIC (Network Interface Controller or Card)**
2. **Media**
3. **Topology**
4. **Protocol**
5. **IP Addresses**

#### **1. NIC (Network Interface Controller or Card)**

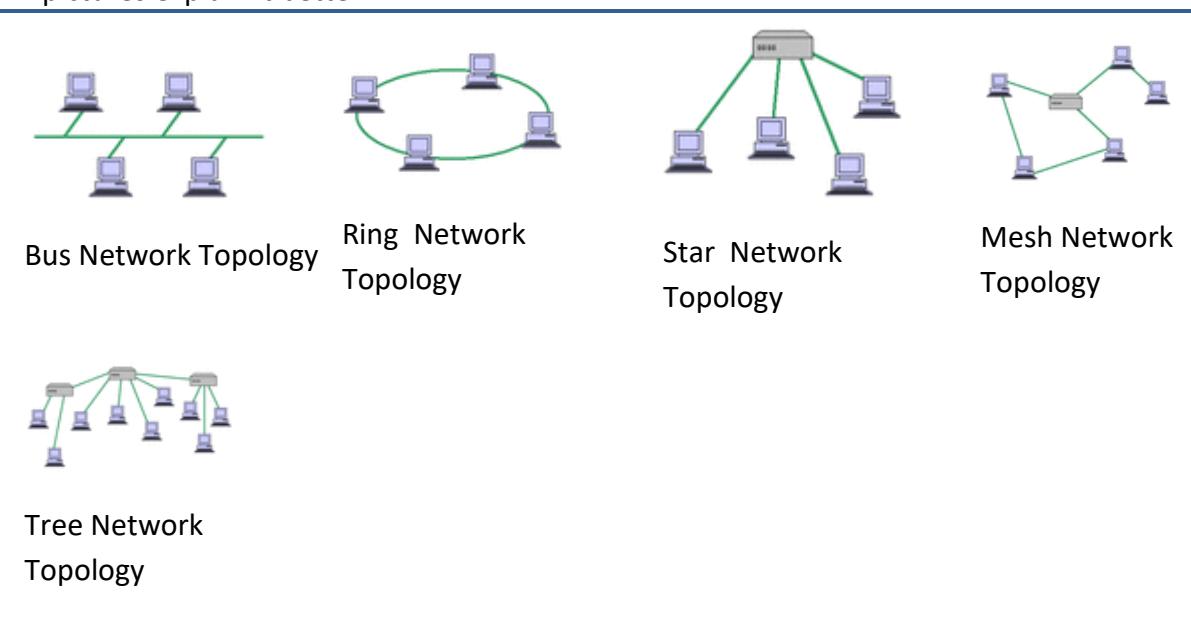
A **network interface controller** (also known as a **network interface card**, **network adapter**, **LAN adapter** and by similar terms) is a computer hardware component that connects a computer to a computer network. Each NIC will be having a unique MAC addresses (**Media Access Control address**) to avoid conflicts between same NIC adapters. In Linux these NIC adapter is represented by the word “**eth**” or “**ens**”. Example if there are two Ethernet adapters in the system then it will be denoted as eth0, eth1, ens1, ens2 etc.

#### **2. Media**

Media is the medium via which two different computer's NIC card will be connected. The best example for media is Cable. Example **RJ 45, CAT 5** etc.

#### **3. Topology**

Topology is the scheme or design in which the computers in the network will be connected to each other. Example for topology is Bus, Ring, star, mesh, tree topologies. The following pictures explain it better.



#### 4. Protocol

A **network protocol** defines rules and conventions for communication between network devices. Protocols for computer networking all generally use packet switching techniques to send and receive messages in the form of *packets*.

Network protocols include mechanisms for devices to identify and make connections with each other, as well as formatting rules that specify how data is packaged into messages sent and received. Some protocols also support message acknowledgement and data compression designed for reliable and/or high-performance network communication. Hundreds of different computer network protocols have been developed each designed for specific purposes and environments.

Examples for Protocols are **TCP/IP (Transmission Control Protocol)**, **UDP (User Datagram Protocol)**, **HTTP**. The most widely and regularly used protocols for transferring data are TCP and UDP. Let's analyze some basic differences between **TCP/IP** and **UDP**.

TCP/IP	UDP
Transmission Control Protocol	User Datagram Protocol
It is connection Oriented	Connectionless
Reliable	Non-Reliable
TCP Acknowledgement will be sent/received	No Acknowledgement for UDP
Slow Communication	Faster Communication
Protocol Number for TCP is 6	Protocol Number for UDP is 17
HTTP, FTP, SMTP uses TCP	DNS, DHCP uses UDP

#### 5. IP ADDRESS

An IP address can be thought of as being similar to a phone number. Just as every person who communicates with a telephone is using a phone with a unique phone number, every computer that is on the Internet has a unique IP address. Not only on internet but within an organization every computer is assigned an IP address so that they can communicate with each other. Basically IP addressing is very deep concept. To understand the concept of IP address we need to understand some important aspect of IP Address which is

- IP Address Classes
- Subnet mask
- Gateway

The above concepts in IP Addressing are very important to understand networking clearly.

- [IP Address Classes](#)

The IP addresses are further broken down into classes. These classes are A, B, C, D, E and their possible ranges can be seen in Figure below.

Class	Start	End	Default subnet mask	CIDR
Class A	0.0.0.0	127.255.255.255	255.0.0.0	/8
Class B	128.0.0.0	191.255.255.255	255.255.0.0	/16
Class C	192.0.0.0	223.255.255.255	255.255.255.0	/24
Class D (multicast)	224.0.0.0	239.255.255.255		
Class E (reserved)	240.0.0.0	255.255.255.255		

\*CIDR - Classless Inter-Domain Routing

\* 127.0.0.0 to 127.255.255.255 is reserved for loopback address

### [Loopback](#)

A special IP number (127.0.0.1), that is designated for the software loopback interface of a machine. 127.0.0.0 Through 127.255.255.255 is also reserved for loopback and is used for internal testing on local machines.

### [Multicast](#)

Multicasting allows a single message to be sent to a group of recipients. **Emailing**, **teleconferencing**, are examples of multicasting. It uses the network infrastructure and standards to send messages.

- [Subnet Mask](#)

A subnet mask allows users to identify which part of an IP address is reserved for the network and which part is available for host use. By looking at the IP address alone, especially now with classless inter-domain routing, users cannot tell which part of the address is which. Adding the subnet mask or netmask gives users all the information needed to calculate network and host portions of the address with ease. In summary, knowing the subnet mask can allow users to easily calculate whether IP addresses are on the same subnet or not.

A commonly used netmask is a 24-bit netmask as seen below.

<b>Netmask:</b>	<b>255.</b>	<b>255.</b>	<b>255.</b>	<b>0</b>
<b>Binary:</b>	<b>11111111</b>	<b>11111111</b>	<b>11111111</b>	<b>00000000</b>
<b>Netmask length</b>	<b>8</b>	<b>16</b>	<b>24</b>	<b>--</b>

- Gateway

A gateway is a network point that provides entrance into another network. On the Internet, a node or stopping point can be either a gateway node or a host (end-point) node. Both the computers of Internet users and the computers that serve pages to users are host nodes. The computers that control traffic within your company's network or at your local Internet service provider (ISP) are gateway nodes.

For example let's say our network is 192.168. something and we want to send a file to other computer on 10.10.network, so we need a gateway to communicate between two computers of different networks.

### Some Important configuration files/directories of network configurations

#/etc/sysconfig/network-scripts is the directory which keeps the configuration of network devices connected to the system.

<pre>[root@ linux network-scripts]# ls ifcfg-eth0  ifdown-isdn  ifup-aliases ifcfg-lo    ifdown-post   ifup-bnep ifdown      ifdown-ppp   ifup-eth ifdown-bnep ifdown-routes ifup-ipp ifdown-eth  ifdown-sit   ifup-ipv6 ifdown-ipp  ifdown-tunnel ifup-isdn ifdown-ipv6 ifup        ifup-ppip</pre>	<pre>[root@mlinux7 network-scripts]# ls ifcfg-<b>eno1</b>6777736  ifdown-ipp ifcfg-lo          ifdown-ipv6 ifdown           ifdown-isdn ifdown-bnep       ifdown-post ifdown-eth        ifdown-ppp ifdown-ib         ifdown-routes</pre>
<b>RHEL6</b>	<b>RHEL7/8</b>

#/etc/sysconfig/network (in rhel6) & #/etc/hostname (in rhel7) is the files which keeps the information about the hostname assigned to the system. If you want to change the hostname permanently, you need to change the hostname in this file.

<b>RHEL6</b>
<pre>[root@mlinux6 ~]# cat /etc/sysconfig/network NETWORKING=yes HOSTNAME=mlinux6.mb.com</pre>

<b>RHEL7/8</b>
<pre>[root@mlinux7 ~]# cat /etc/hostname mlinux7.mb.com</pre>

#/etc/hosts a file which is responsible for resolving hostname into IP locally, in other word it acts as local DNS if DNS server is not accessible. There is no change In this file in either versions

<pre>[root@mlinux7 ~]# cat /etc/hosts 127.0.0.1  localhost localhost.localdomain localhost4 localhost4.localdomain4 ::1        localhost localhost.localdomain localhost6 localhost6.localdomain6 192.168.100.10  mlinux6.mb.com  mlinux6</pre>
---

#/etc/resolv.conf is a file which keeps the address of DNS server to which the clients will be accessing to resolve IP to hostname and hostname to IP.

<pre>[root@ linux ~]# cat /etc/resolv.conf # Generated by NetworkManager search vpts.com nameserver 192.168.10.98</pre>
---

## LAB WORK:

- To check the ip address assign to all the interfaces

#ifconfig

```
[root@mlinux7 ~]# ifconfig
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.20 netmask 255.255.255.0 broadcast 192.168.100.255
        inet6 fe80::20c:29ff:fea:9af0 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:1a:9a:f0 txqueuelen 1000 (Ethernet)
                RX packets 382 bytes 35593 (34.7 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 246 bytes 42590 (41.5 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 0 (Local Loopback)
            RX packets 532 bytes 43824 (42.7 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 532 bytes 43824 (42.7 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- To check the ip of a particular interface

#ifconfig < adapter name >

#ifconfig eth0; #ifconfig ensxx

```
[root@mlinux7 ~]# ifconfig eno16777736
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.20 netmask 255.255.255.0 broadcast 192.168.100.255
        inet6 fe80::20c:29ff:fea:9af0 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:1a:9a:f0 txqueuelen 1000 (Ethernet)
                RX packets 420 bytes 38917 (38.0 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 267 bytes 46296 (45.2 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- To check the hostname of the system.

#hostname

```
[root@mlinux6 ~]# hostname
mlinux6.mb.com
```

- To check ip of the host

#hostname -i

```
[root@ linux ~]# hostname -i
192.168.10.98 127.0.0.1
```

- To check whether DNS is resolving or not

```
#host < ip address >
```

```
#host 192.168.10.95
```

```
[root@ linux ~]# host 192.168.10.98
98.10.168.192.in-addr.arpa domain name pointer linux.mb.com.
```

```
#host <hostname>
```

```
#host mylinux.kt.com
```

```
[root@ linux ~]# host linux.mb.com.
linux.mb.com has address 192.168.10.98
```

- Same with “nslookup” command

```
#nslookup < ip address >
```

```
#nslookup < hostname >
```

```
[root@ linux ~]# nslookup 192.168.10.98
Server:          192.168.10.98
Address:         192.168.10.98#53

98.10.168.192.in-addr.arpa      name = mb.com.
98.10.168.192.in-addr.arpa      name = linux.mb.com.
```

```
[root@ linux ~]# nslookup linux.mb.com
Server:          192.168.10.98
Address:         192.168.10.98#53

Name:    linux.mb.com
Address: 192.168.10.98
```



- The most common command used to check DNS function is “dig”

```
#dig <hostname>
```

```
[root@ linux ~]# dig linux.mb.com

; <>> DiG 9.7.0-P2-RedHat-9.7.0-5.P2.el6 <>> linux.mb.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11898
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
; linux.mb.com.           IN      A

;; ANSWER SECTION:
linux(mb).com.      10800   IN      A      192.168.10.98

;; AUTHORITY SECTION:
linux(mb).com.      10800   IN      NS     linux(mb).com.
```

### With ip address

```
#dig -x <ip address>
#dig -x 192.168.10.98
```

```
[root@ linux ~]# dig -x 192.168.10.98

; <>> DiG 9.7.0-P2-RedHat-9.7.0-5.P2.el6 <>> -x 192.168.10.98
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4532
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;98.10.168.192.in-addr.arpa. IN PTR

;; ANSWER SECTION:
98.10.168.192.in-addr.arpa. 10800 IN PTR linux.mb.com.
98.10.168.192.in-addr.arpa. 10800 IN PTR mb.com.

;; AUTHORITY SECTION:
10.168.192.in-addr.arpa. 10800 IN NS linux.mb.com.
```

- Checking network connectivity using ping command

```
#ping < ip address >
#ping 192.168.10.95
```

```
[root@ linux ~]# ping 192.168.10.95
PING 192.168.10.95 (192.168.10.95) 56(84) bytes of data.
64 bytes from 192.168.10.95: icmp_seq=1 ttl=64 time=0.115 ms
64 bytes from 192.168.10.95: icmp_seq=2 ttl=64 time=0.140 ms
64 bytes from 192.168.10.95: icmp_seq=3 ttl=64 time=0.099 ms
64 bytes from 192.168.10.95: icmp_seq=4 ttl=64 time=0.111 ms
64 bytes from 192.168.10.95: icmp_seq=5 ttl=64 time=0.110 ms
^C
--- 192.168.10.95 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4431ms
rtt min/avg/max/mdev = 0.099/0.115/0.140/0.013 ms
```

Note: use **ctrl + c** to stop pinging.

- To limit the pinging for specific number of counts

```
#ping -c <counts> <ip address>
#ping -c 2 192.168.10.95
```

```
[root@ linux ~]# ping -c 2 192.168.10.95
PING 192.168.10.95 (192.168.10.95) 56(84) bytes of data.
64 bytes from 192.168.10.95: icmp_seq=1 ttl=64 time=0.100 ms
64 bytes from 192.168.10.95: icmp_seq=2 ttl=64 time=0.106 ms

--- 192.168.10.95 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.100/0.103/0.106/0.003 ms
```

## Changing the hostname

- Check the current hostname with **hostname** command
- The syntax for changing the hostname is  
**#hostname <new name>**  
**#hostname mlinux6.mb.com**

```
[root@mlinux6 ~]# hostname
mlinux2.mb.com
[root@mlinux6 ~]# hostname mlinux6.mb.com
[root@mlinux6 ~]# hostname
mlinux6.mb.com
```

**Note:** The above change is temporary and will be last only till you are logged in, if you want to change it permanently edit the **/etc/sysconfig/network (rhel6)** and **/etc/hostname (rhel7)** file and then logout and login to confirm the change.

**#vim /etc/sysconfig/network; #vim /etc/hostname** delete the previous hostname and add the new name.

```
mlinux6.mb.com
```

- Now logoff and logon and check the hostname

```
[root@mlinux6 ~]# hostname
mlinux6.mb.com
```

**Note:** Once you logout and login again the change will be permanent, observe the highlighted region above.

- hostnamectl command**

In rhel7 a new command has been introduced to see the hostname/system details as well as changing hostname permanently.

```
[root@node1 ~]# hostnamectl
  Static hostname: node1.private
    Icon name: computer-vm
    Chassis: vm
    Machine ID: 36462571f9e14f54b7cf5065ee8cd22b
      Boot ID: ca0f6031684a47e59abe13ab49790b23
  Virtualization: vmware
Operating System: Red Hat Enterprise Linux Server 7.6 (Maipo)
  CPE OS Name: cpe:/o:redhat:enterprise_linux:7.6:GA:server
        Kernel: Linux 3.10.0-957.el7.x86_64
  Architecture: x86-64
```

- Changing hostname permanently with **hostnamectl** command

**#hostnamectl set-hostname <new name>**

```
[root@node1 ~]# hostnamectl set-hostname mlinux7.vpts.com
[root@node1 ~]# hostname
mlinux7.vpts.com
[root@node1 ~]# cat /etc/hostname
mlinux7.vpts.com
```

## Assigning /Changing the IP Address

Steps for changing the IP Address

To change the IP Address use the following command line

**To see the all the available connection**

**#nmcli con show**

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc 802-3-ethernet ens3
```

**To see the details of any connection**

**#nmcli con show <connection name>**

```
[root@mlinux2 ~]# nmcli con show ens3
connection.id:                           ens3
connection.uuid:                         e4a31311-1ac5-4614-83ee-03629af62ddc
connection.interface-name:                ens3
connection.type:                          802-3-ethernet
connection.autoconnect:                  yes
connection.autoconnect-priority:        0
connection.timestamp:                   1466229442

ipv4.addresses:                          192.168.106.82/24
ipv4.gateway:                           192.168.106.1
```

**To add a new connection**

**#nmcli con add con-name <name of connection> ifname <interface name> type <type of connection> autoconnect <yes/no> ip4/ip6 <ip add> gw4/gw6 <gateway>**

```
[root@mlinux2 ~]# nmcli con add con-name mycon ifname ens3 type ethernet autoconnect no
ip4 192.168.106.182/24 gw4 192.168.106.1
```

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
mycon    268e6fb2-662f-4482-af68-161a1a4ad775 802-3-ethernet  --
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc 802-3-ethernet ens3
```

**To activate a connection**

**#nmcli con up <con name>**

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
mycon    268e6fb2-662f-4482-af68-161a1a4ad775 802-3-ethernet  --
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc 802-3-ethernet ens3
[root@mlinux2 ~]# nmcli con up mycon
```

```
Pinging 192.168.106.182 with 32 bytes of data:
Request timed out.
Reply from 192.168.106.182: bytes=32 time=9ms TTL=63
Reply from 192.168.106.182: bytes=32 time=10ms TTL=63
Reply from 192.168.106.182: bytes=32 time=11ms TTL=63
Reply from 192.168.106.182: bytes=32 time=1ms TTL=63
```

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
mycon    268e6fb2-662f-4482-af68-161a1a4ad775 802-3-ethernet ens3
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc 802-3-ethernet --
```

### To modify an existing connection

```
#nmcli con mod <con-name> attributes <value>
```

```
[root@mlinux2 ~]# nmcli con mod mycon ipv4.addresses 192.168.106.188/24
[root@mlinux2 ~]# nmcli con show mycon |grep ipv4.addresses
ipv4.addresses: 192.168.106.188/24
```

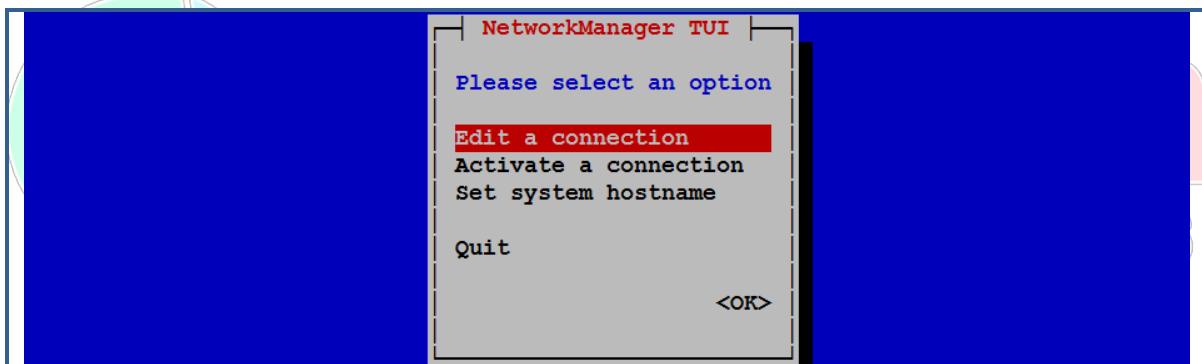
### To delete a connection

```
#nmcli con del <con-name>
```

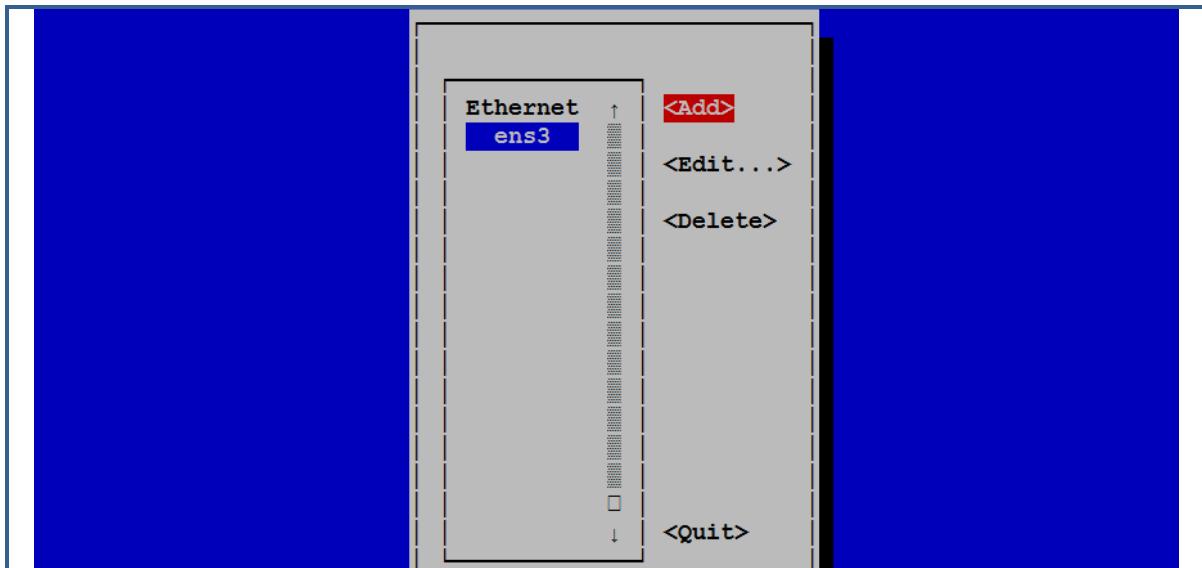
```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
mycon    268e6fb2-662f-4482-af68-161a1a4ad775  802-3-ethernet  --
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc  802-3-ethernet  ens3
[root@mlinux2 ~]# nmcli con del mycon
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc  802-3-ethernet  ens3
[root@mlinux2 ~]#
```

### Using nmtui for a text based tool

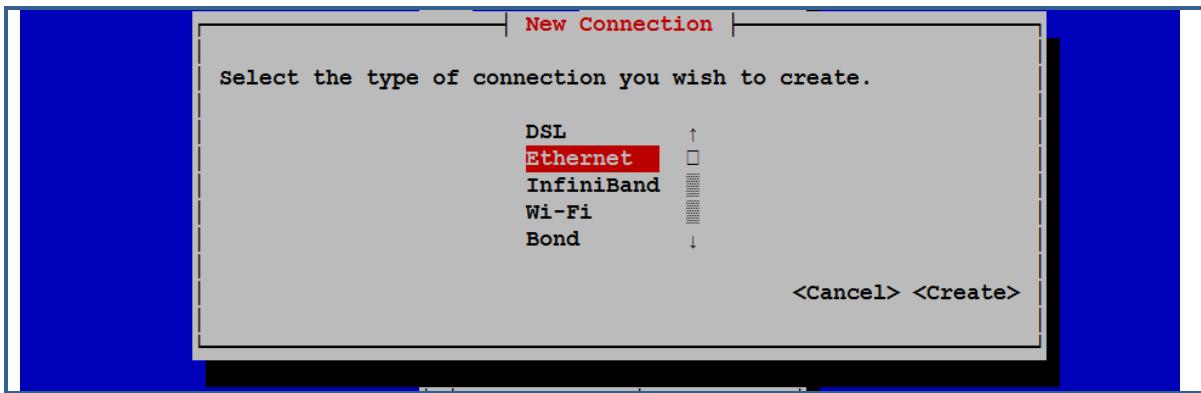
```
#nmtui
```



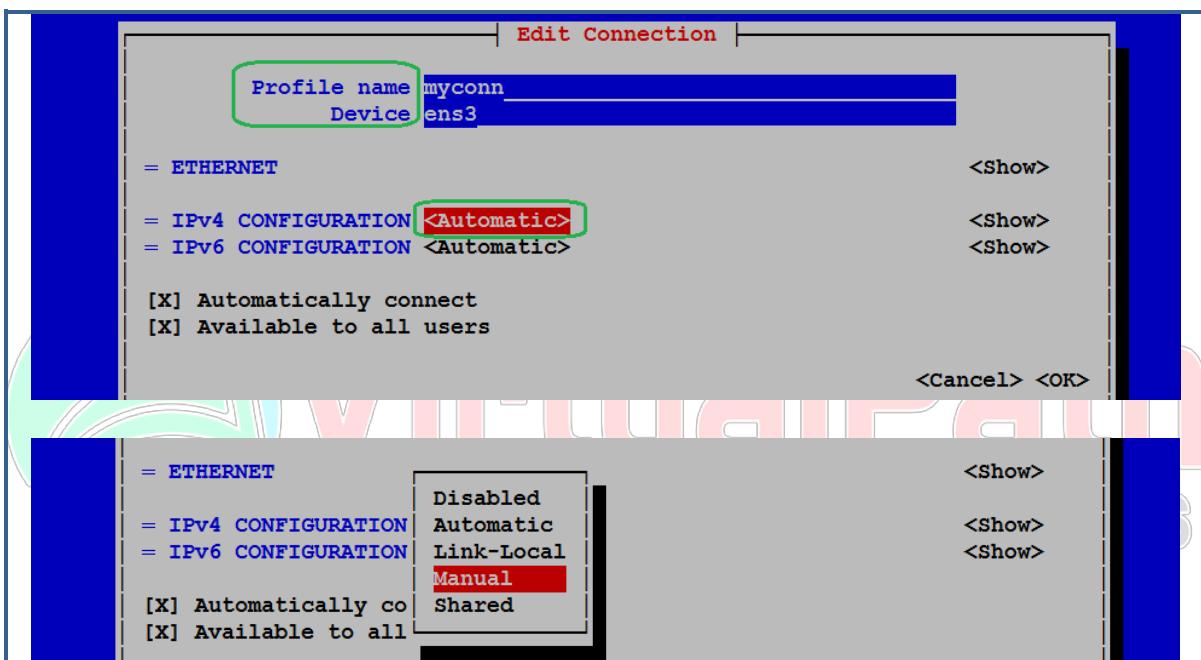
Move the cursor to *Edit a connection* and press Enter



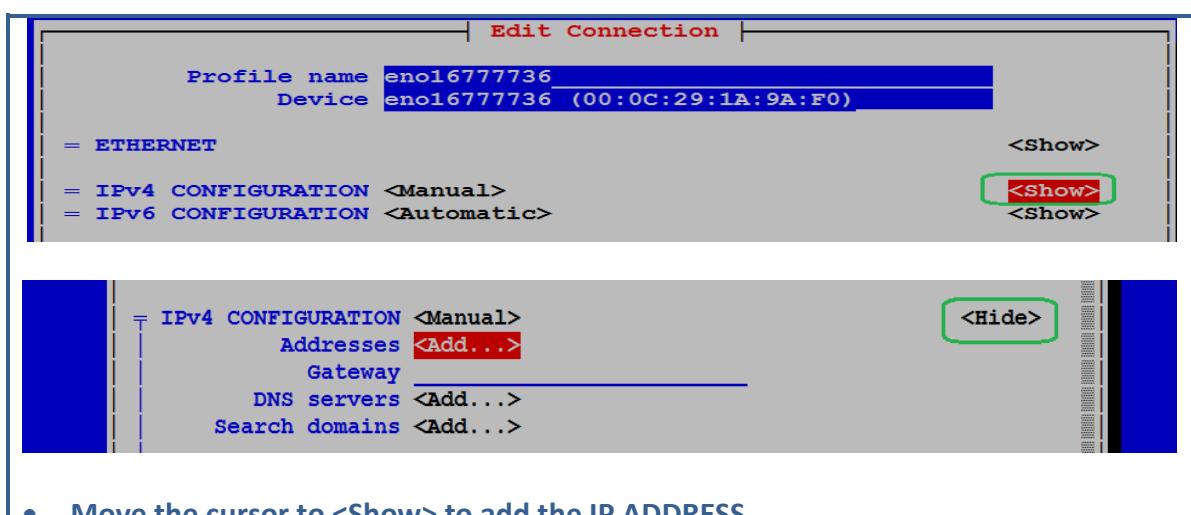
Move the cursor to '*Add*' to add a new connection and press Enter



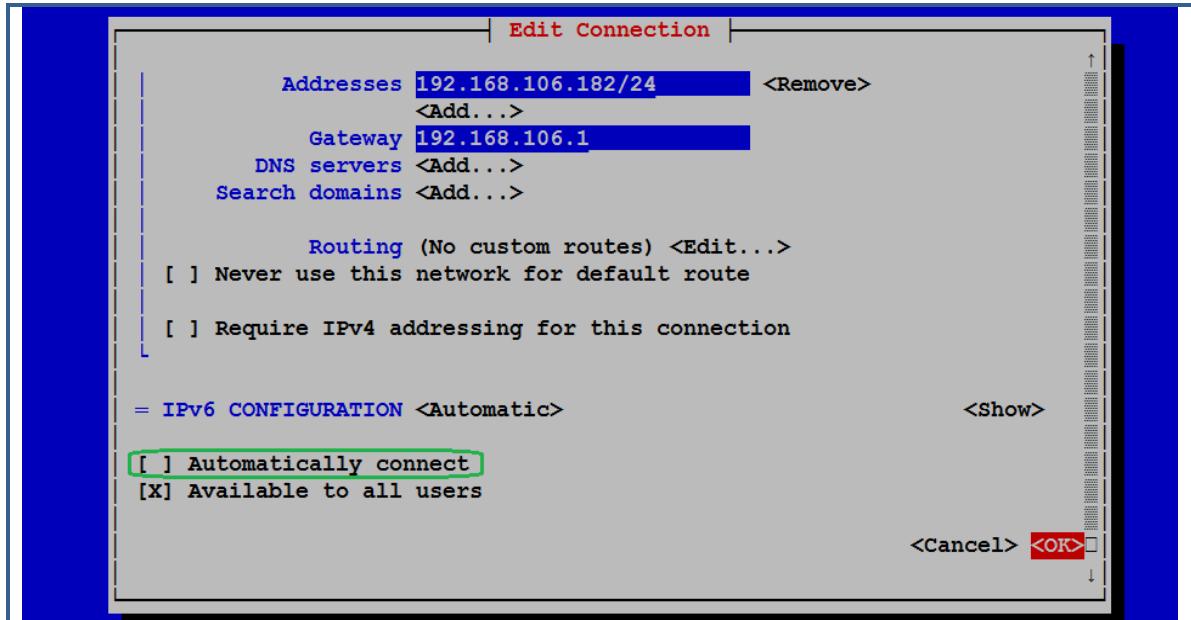
Now select the type of connection, ex., Ethernet and press Enter



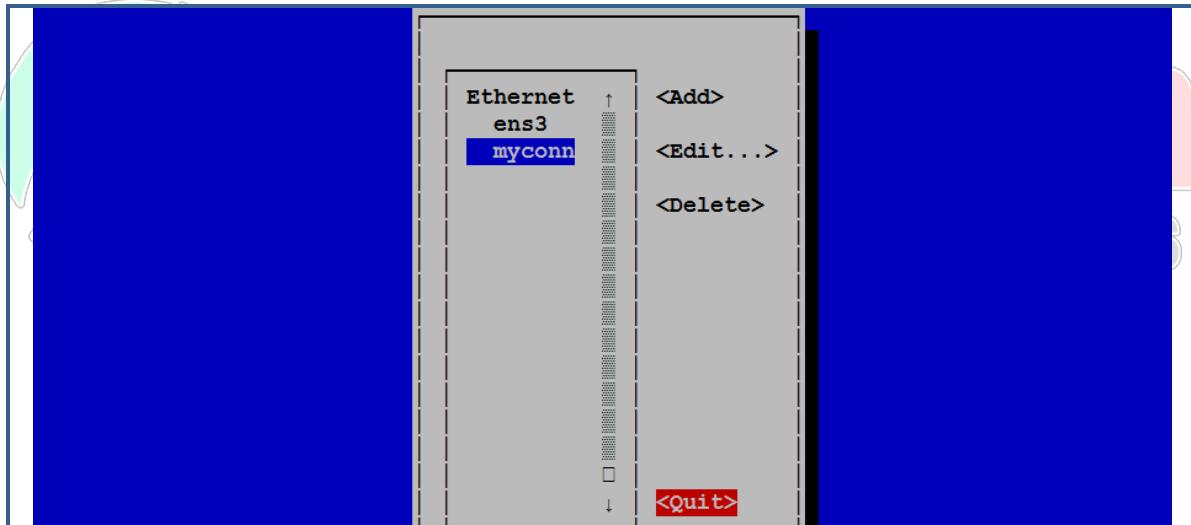
Assign the connection name, Device name and change the ipv4 from automatic to manual to continue



- Move the cursor to <Show> to add the IP ADDRESS



Move the cursor to “add” to assign the ip address move cursor to “ok” Enter.

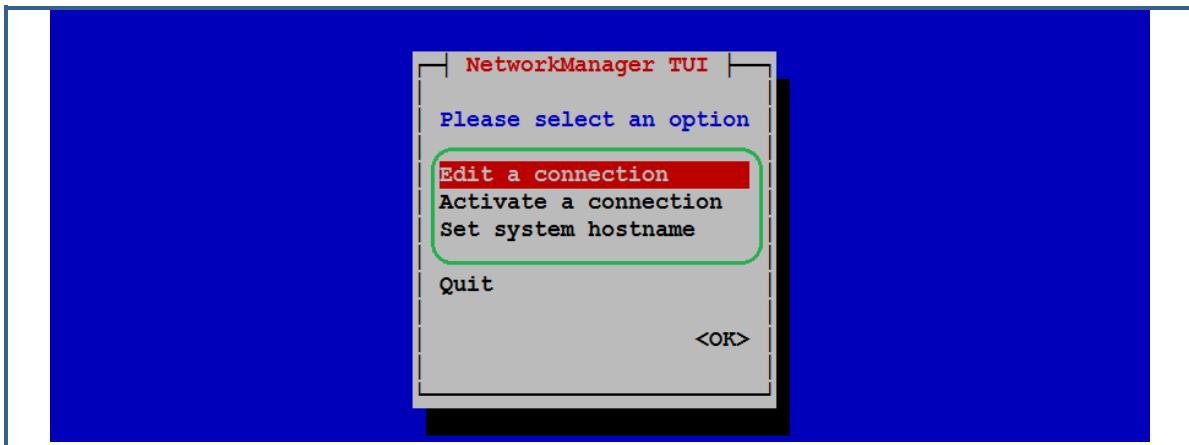


move the cursor to “Quit” button and press Enter to quit the utility

Check the connection by nmcli con show command

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
myconn    62351fe9-9b6b-4596-b030-d4fe208c095f  802-3-ethernet --
ens3      e4a31311-1ac5-4614-83ee-03629af62ddc  802-3-ethernet ens3
```

In the same fashion, the hostname, connections can be created and modify



- To Know more about the NIC card/adapter use  
**#ethtool <adapter name>**

```
[root@client ~]# ethtool ens192
Settings for ens192:
    Supported ports: [ TP ]
    Supported link modes:  1000baseT/Full
                           10000baseT/Full
    Supported pause frame use: No
    Supports auto-negotiation: No
    Supported FEC modes: Not reported
    Advertised link modes: Not reported
    Advertised pause frame use: No
    Advertised auto-negotiation: No
    Advertised FEC modes: Not reported
    Speed: 10000Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 0
    Transceiver: internal
    Auto-negotiation: off
    MDI-X: Unknown
    Supports Wake-on: uag
    Wake-on: d
    Link detected: yes
```



- To see the same in summarized way

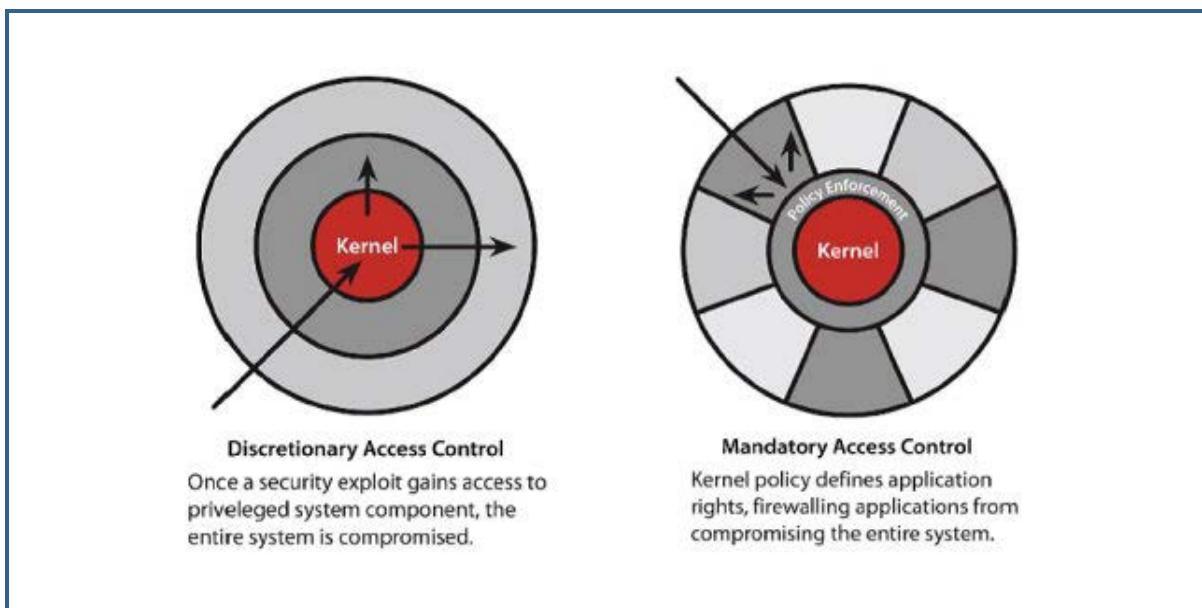
**# mii-tool <adapter name>**

```
[root@localhost ~]# mii-tool enp0s3
enp0s3: no autonegotiation, 1000baseT-FD flow-control, link ok
```

## MANAGING SELINUX (BASICS OF SELINUX)

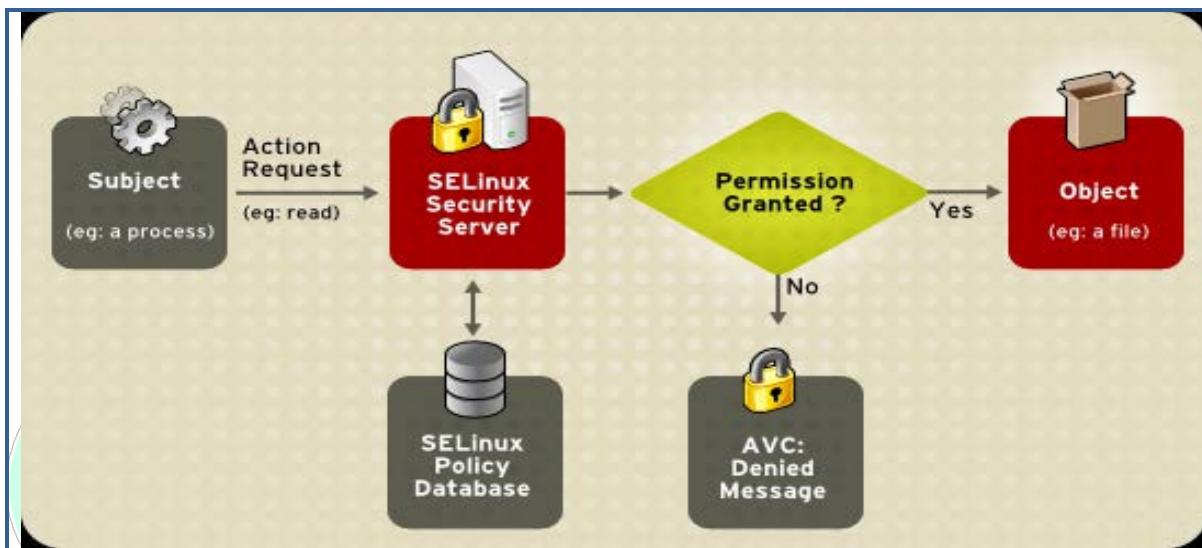
### Basic SELinux Security Concepts

- SELinux is a security enhancement to Linux that allows users and administrators more control over which users and applications can access which resources, such as files. Standard Linux access controls, such as file modes (-rwxr-xr-x) are modifiable by the user and applications that the user runs, whereas SELinux access controls are determined by a policy loaded on the system and not changeable by careless users or misbehaving applications.
- SELinux also adds finer granularity to access controls. Instead of only being able to specify who can read, write or execute a file, for example, SELinux lets you specify who can unlink, append only, and move a file and so on. SELinux allows you to specify access to many resources other than files as well, such as network resources and inter-process communication (IPC).
- SELinux provides a flexible **Mandatory Access Control (MAC)** system built into the Linux kernel. Under standard Linux **Discretionary Access Control (DAC)**, an application or process running as a user (UID or SUID) has the user's permissions to objects such as files, sockets, and other processes. Running a MAC kernel protects the system from malicious or flawed applications that can damage or destroy the system. The following picture explains more detailed about both Access controls.



- The SELinux Decision Making Process**

When a subject, (for example, an application), attempts to access an object (for example, a file), the policy enforcement server in the kernel checks an *access vector cache* (AVC), where subject and object permissions are cached. If a decision cannot be made based on data in the AVC, the request continues to the security server, which looks up the *security context* of the application and the file in a matrix. Permission is then granted or denied, with an avc: denied message detailed in /var/log/messages if permission is denied. The security context of subjects and objects is applied from the installed policy, which also provides the information to populate the security server's matrix.



- Important SELinux configuration Files**

/etc/selinux/config is the main configuration file of SELinux.

/etc/sysconfig/selinux contains a symbolic link to the actual configuration file, /etc/selinux/config.

**Note:** If you want to turn on or off the SELinux security you need to make changes in the main configuration file i.e. /etc/selinux/config file. Well we'll see it later in this chapter.

```
[root@ linux ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

## Modes of SELinux

- There are three modes in which SELinux can be at a time, they are
- **Enforcing, Permissive and Disabled**
  - **Enforcing**  
Enable and enforce the SELinux security policy on the system, denying access and logging actions
  - **Permissive**  
Permissive mode is similar to Debugging Mode. In Permissive Mode, SELinux policies and rules are applied to subjects and objects, but actions (for example, Access Control denials) are not affected. The biggest advantage of Permissive Mode is that log files and error messages are generated based on the SELinux policy implemented.
  - **Disabled**  
SELinux is turned off and no warn and log messages will be generated and stored.
- **Booleans**  
Booleans are variables that can either be set as true or false. Booleans enhance the effect of SELinux policies by letting the system administrator fine tune a policy. A policy may protect a certain daemon or service by applying various access control rules. In real world scenarios, a system administrator would not like to implement all the access controls specified in the policy.

## SELinux Policy

- The SELinux Policy is the set of rules that guide the SELinux security engine. It defines *types* for file objects and *domains* for processes. It uses roles to limit the domains that can be entered, and has user identities to specify the roles that can be attained. In essence, types and domains are equivalent, the difference being that types apply to objects while domains apply to processes.

## SELinux Context

- Processes and files are labeled with a SELinux context that contains additional information, such as a SELinux user, role, type, and, optionally, a level.

## LAB WORK:-

### To check the SELinux Mode

#getenforce

```
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]#
```

#sestatus

[root@mlinux6 ~]# sestatus	RHEL6
SELinux status:	enabled
SELinuxfs mount:	/selinux
Current mode:	enforcing
Mode from config file:	enforcing
Policy version:	24
Policy from config file:	targeted

[root@mlinux7 ~]# sestatus	RHEL7/8
SELinux status:	enabled
SELinuxfs mount:	/sys/fs/selinux
SELinux root directory:	/etc/selinux
Loaded policy name:	targeted
Current mode:	enforcing
Mode from config file:	enforcing
Policy MLS status:	enabled
Policy deny_unknown status:	allowed
Max kernel policy version:	28

*Note: Observe that there is a small change between 6, 7&8, which the mount point*

### Display the SELinux context of a file or directory.

- To display the context of a file the syntax is  
#ls -Z <filename>

```
[root@ linux ~]# ls
anaconda-ks.cfg  Documents  install.log      ktfile  Pictures  Templates
Desktop          Downloads  install.log.syslog  Music   Public    Videos
[root@ktlinux ~]# ls -Z ktfile
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 ktfile
[root@ktlinux ~]#
```

- To display the context of a directory the syntax is

#ls -ldZ <directory name>

```
[root@ linux ~]# ls -ldZ Documents
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Documents
[root@ktlinux ~]#
```

### Changing the SELinux Context of a file or directory

- To change the context of the file the steps are
  - Check the existing context of the file by  
#ls -ldZ <filename>

```
[root@ linux ~]# ls -ldZ myfile
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 myfile
```

Observe that the type is **admin\_home\_t**, let's change it to **public\_content\_t**, so that it will be available for all users.

- To change the context of a file or directory the syntax is  
**#chcon -t <argument> <file/dir name>**  
**#chcon -t public\_content\_t myfile**

```
[root@ linux ~]# chcon -t public_content_t myfile
[root@ linux ~]# ls -ldZ myfile
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 myfile
[root@ linux ~]#
```

- To change the context for a directory and its contents
- Check the context of both directory and its contents

```
[root@ linux ~]# ls -ldZ mydir
drwxr-xr-x. root root system_u:object_r:admin_home_t:s0 mydir
[root@ linux ~]# ls -lZmydir
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file1
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file2
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file3
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file4
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file5
[root@ linux ~]#
```

- To change the context for a directory and its contents, the syntax is  
**#chcon -R -t <argument> <dir name>**  
**#chcon -R -t public\_content\_t mydir**

```
[root@ linux ~]# chcon -R -t public_content_t mydir
[root@ linux ~]# ls -ldZ mydir
drwxr-xr-x. root root system_u:object_r:public_content_t:s0 mydir
[root@ linux ~]# ls -lZ mydir
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file1
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file2
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file3
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file4
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file5
[root@ linux ~]#
```

### Restoring back the modified SELinux context to its default value

- To restore the modified/changed SELinux context of a file to its default one, the syntax is  
**#restorecon -v <filename>**  
**#restorecon -v myfile**

```
[root@ linux ~]# ls -ldZ myfile
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 myfile
[root@ linux ~]# restorecon -v myfile
restorecon reset /root/ktfile context unconfined_u:object_r:public_content_t:s0->system_u:object_r:admin_home_t:s0
[root@ linux ~]# ls -ldZ myfile
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 myfile
[root@ linux ~]#
```

- To restore back the same of a directory with its contents, the syntax is

```
#restorecon -Rv <dir name >
```

```
#restorecon -Rv mydir
```

```
[root@mlinux6 ~]# ls -lZ mydir
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file1
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file2
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file3
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file4
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file5
[root@mlinux6 ~]# restorecon -R mydir
[root@mlinux6 ~]# ls -lZ mydir
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file1
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file2
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file3
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file4
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file5
```

**Note:** For restoring the context of only the dir except its contents do not add "R" in the command.

### Changing the default context for the file:

In this case we want to change the default context of the file, so that even the context is restored any time it should be changed to the context we want.

- First of all check the present context on the file

```
[root@mlinux3 ~]# ls -lZ /myfile
-rw-r--r--. 1 root root unconfined_u:object_r:etc_runtime_t:s0 0 Apr 19 22:50 /myfile
```

- Let's change the default context to something required for example **samba\_share\_t**

```
# semanage fcontext -a -t samba_share_t /myfile
```

```
[root@mlinux3 ~]# semanage fcontext -a -t samba_share_t /myfile
```

- The **-a** option adds a new record, and the **-t** option defines a type (**samba\_share\_t**).  
**Note:** running this command does not directly change the type - myfile is still labeled with the **etc\_runtime\_t** type:
- The **semanage fcontext -a -t samba\_share\_t /etc/file1** command adds the following entry to **/etc/selinux/targeted/contexts/files/file\_contexts.local**:

```
[root@mlinux3 ~]# cat /etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.

myfile    system_u:object_r:samba_share_t:s0
 myfile    system_u:object_r:samba_share_t:s0
```

- Now restore the context and verify is it changing by default to **samba\_share\_t**

```
[root@mlinux3 ~]# restorecon -v /myfile
Relabeled /myfile from unconfined_u:object_r:etc_runtime_t:s0 to unconfined_u:object_r:samba_share_t:s0
[root@mlinux3 ~]# ls -lZ /myfile
-rw-r--r--. 1 root root unconfined_u:object_r:samba_share_t:s0 0 Apr 19 22:50 /myfile
[root@mlinux3 ~]#
```

## Changing the default context for the directory and its content:

In this case we want to change the default context of the dir, so that even the context is restored any time it should be changed to the context we want.

- First of all check the present context on the file

```
[root@mlinux3 ~]# ls -ldZ /mydir
drwxr-xr-x. 2 root root unconfined_u:object_r:default_t:s0 45 Apr 19 23:14 /mydir
[root@mlinux3 ~]# ls -lZ /mydir
total 0
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 0 Apr 19 23:14 file1
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 0 Apr 19 23:14 file2
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 0 Apr 19 23:14 file3
[root@mlinux3 ~]#
```

- Let's change the default context to something required for example **httpd\_sys\_content\_t**.

```
# semanage fcontext -a -t httpd_sys_content_t "/mydir(/.*)?"
```

```
[root@mlinux3 ~]# semanage fcontext -a -t samba_share_t /myfile
```

- The **-a** option adds a new record, and the **-t** option defines a type (**httpd\_sys\_content\_t**). The **" /mydir(/.\*)?"** regular expression causes the **semanage** command to apply changes to the **/mydir/** directory, as well as the files in it.
- Note: running this command does not directly change the type **/mydir/** and files in it are still labeled with the **default\_t** type:
- The **semanage fcontext -a -t httpd\_sys\_content\_t "/mydir(/.\*)?"** command adds the following entry to **/etc/selinux/targeted-contexts/files/file\_contexts.local**:

```
[root@mlinux3 ~]# cat /etc/selinux/targeted-contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.

myfile    system_u:object_r:samba_share_t:s0
 myfile    system_u:object_r:samba_share_t:s0
 /mydir(/.*)?    system_u:object_r:httpd_sys_content_t:s0
[root@mlinux3 ~]#
```

- Now restore the context and verify is it changing by default to **httpd\_sys\_content**

```
[root@mlinux3 ~]# restorecon -R /mydir
[root@mlinux3 ~]# ls -ldZ /mydir
drwxr-xr-x. 2 root root unconfined_u:object_r:httpd_sys_content_t:s0 45 Apr 19 23:14 /mydir
[root@mlinux3 ~]# ls -lZ /mydir
total 0
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 0 Apr 19 23:14 file1
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 0 Apr 19 23:14 file2
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 0 Apr 19 23:14 file3
[root@mlinux3 ~]#
```

### Checking the Booleans and modifying it.

- To see the Booleans of a particular service, the syntax is  

```
#getsebool -a |grep <service name >
#getsebool -a |grep ftp
```

**Note1:** if **grep** is not used it will list Booleans for all the services in the system and output will be very lengthy.

**Note2:** Booleans can only be checked and changed when **SELinux** is in enforcing or Permissive modes; if the SELinux is in disabled mode Booleans cannot be modified.

```
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]# getsebool -a |grep ftp
allow_ftpd_anon_write --> off
allow_ftpd_full_access --> off
allow_ftpd_use_cifs --> off
allow_ftpd_use_nfs --> off
ftp_home_dir --> off
ftpd_connect_db --> off
httpd_enable_ftp_server --> off
sftpd_anon_write --> off
sftpd_enable_homedirs --> off
sftpd_full_access --> off
sftpd_write_ssh_home --> off
tftp_anon_write --> off
[root@ linux ~]#
```

- To change any Boolean just copy the Boolean and give the option (the only possible option for a Boolean to enable and disable is **on/off**). The syntax for changing Boolean value is  
**#setsebool < Boolean > < option (on/off) >**  
**#setsebool allow\_ftpd\_anon\_write on**; Verify the change with **getsebool** command.

```
[root@ tlinux ~]# setsebool allow_ftpd_anon_write on
[root@ tlinux ~]# getsebool -a |grep ftp
allow_ftpd_anon_write --> on
allow_ftpd_full_access --> off
allow_ftpd_use_cifs --> off
allow_ftpd_use_nfs --> off
ftp_home_dir --> off
ftpd_connect_db --> off
httpd_enable_ftp_server --> off
sftpd_anon_write --> off
sftpd_enable_homedirs --> off
sftpd_full_access --> off
sftpd_write_ssh_home --> off
tftp_anon_write --> off
[root@ tlinux ~]#
```

- Making boolean permanent**

**#setsebool -P <Boolean> on/off to make change permanent**

```
[root@node1 ~]# setsebool -P ftpd_anon_write on
[root@node1 ~]# getsebool -a |grep ftp
ftpd_anon_write --> on
ftpd_connect_all_unreserved --> off
ftpd_connect_db --> off
ftpd_full_access --> on
ftpd_use_cifs --> off
ftpd_use_fusefs --> off
```

## Changing the Modes of SELinux

- To change the mode of SELinux the syntax is  
**#setenforce <option>**  
Options used are **0** or **1** (Where **0** means **Permissive** and **1** means **Enforcing**)
- To change the SELinux Mode to permissive  
**#setenforce 0**
- Verify it by **getenforce** or **sestatus** command.

```
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]# setenforce 0
[root@ linux ~]# getenforce
Permissive
[root@ linux ~]# sestatus
SELinux status:          enabled
SELinuxfs mount:         /selinux
Current mode:            permissive
Mode from config file:   enforcing
Policy version:          24
Policy from config file: targeted
[root@ linux ~]#
```

- To change the SELinux Mode back to Enforcing mode  
**#setenforce 1**
- Verify the change

```
[root@ linux ~]# getenforce
Permissive
[root@ linux ~]# setenforce 1
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]# sestatus
SELinux status:          enabled
SELinuxfs mount:         /selinux
Current mode:            enforcing
Mode from config file:   enforcing
Policy version:          24
Policy from config file: targeted
[root@ linux ~]#
```

### Disabling and Enabling the SELinux Security

- To disable the SELinux protection or to change it to **disabled** Mode
- Edit the **/etc/selinux/config** file and change **SELINUX=disabled**
- Whenever changing the mode of **SELinux** from **Enforcing/Permissive** to **Disabled** or **Disabled** to **Permissive/Enforcing**, you need to restart the system so that the changes can take effect.
- First check the current status of **SELinux** and the configuration file.

```
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.

SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- Now, edit the configuration file, restart the computer and check the status.

```
#vim /etc/selinux/config
#reboot (to reboot the system)
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.

SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

```
[root@ linux ~]# getenforce
Disabled
[root@ linux ~]# sestatus
SELinux status:                 disabled
[root@ linux ~]#
```

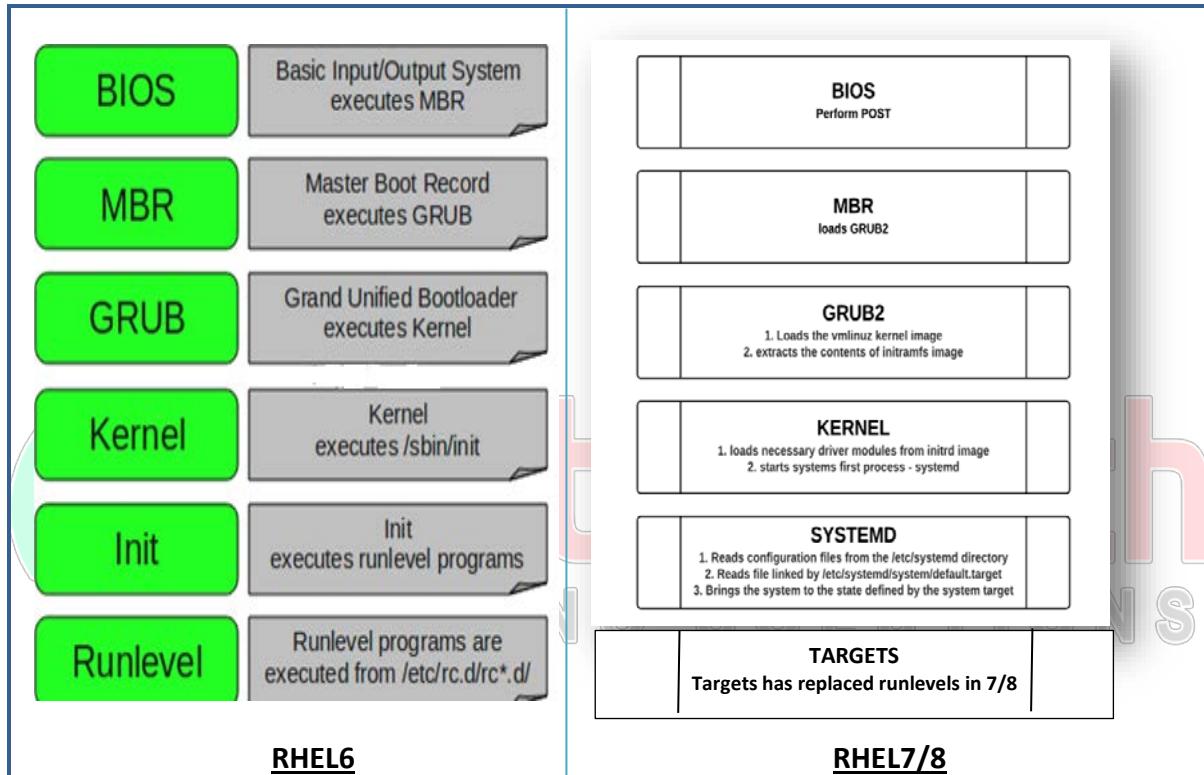
To Enable it back the procedure is exactly same as above, instead of **SELINUX=disabled** change it to **SELINUX=enforcing** or **permissive**. Don't forget to restart the system, unless the system is rebooted the changes will not take effect.

## BOOTING PROCEDURE

Press the power button on your system, and after few moments you see the Linux login prompt.

Have you ever wondered what happens behind the scenes from the time you press the power button until the Linux login prompt appears?

**The following are the 6 high level stages of a typical RHEL7/8 boot process.**



### 1. BIOS/UEFI

- **BIOS** stands for Basic Input/Output System, **UEFI** stands for Unified Extensible Firmware Interface
- Performs some system integrity checks
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 or F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS/UEFI loads and executes the MBR/GPT boot loader.

## 2. MBR/GPT

- MBR stands for Master Boot Record, GPT stands for GUID Partition Table
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB/GRUB2 (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

## 3. GRUB2

- The default bootloader used on RHEL 7&8 is GRUB 2. GRUB stands for GRand Unified Bootloader. GRUB 2 replaces the older GRUB bootloader also called as legacy GRUB.
- The GRUB 2 configuration file is located at **/boot/grub2/grub.cfg** and link at **/etc/grub2.cfg**(Do not edit this file directly).
- GRUB 2 menu-configuration settings are taken from **/etc/default/grub** when generating grub.cfg.

```
[root@localhost ~]# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
[root@localhost ~]#
```

- If changes are made to any of these parameters, you need to run **grub2-mkconfig** to re-generate the **/boot/grub2/grub.cfg** file
- **#grub2-mkconfig -o /boot/grub2/grub.cfg**

### initramfs

- The job of the initial RAM file system is to preload the block device modules, such as for IDE, SCSI, or RAID, so that the root file system, on which those modules normally reside, can then be accessed and mounted.
- The initramfs is bound to the kernel and the kernel mounts this initramfs as part of a two-stage boot process.
- The dracut utility creates initramfs whenever a new kernel is installed.
- Use the lsinitrd command to view the contents of the image created by dracut

## 4. Kernel

- Mounts the root file system as specified in the “root=” in grub2.conf
- Kernel executes the systemd program

Since system is the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.

## 5. Systemd

1. systemd is the ancestor of all processes on a system.
2. systemd reads the file linked by /etc/systemd/system/default.target (for example, /usr/lib/systemd/system/multi-user.target) to determine the default system target (equivalent to run level). The system target file defines the services that systemd starts.
3. systemd brings the system to the state defined by the system target, performing system initialization tasks such as:
  1. Setting the host name
  2. Initializing the network
  3. Initializing SELinux based on its configuration
  4. Printing a welcome banner
  5. Initializing the system hardware based on kernel boot arguments
  6. Mounting the file systems, including virtual file systems such as the /proc file system
  7. Cleaning up directories in /var
  8. Starting swap

## 6. Runlevel/Targets

- Prior to RHEL 7, runlevels were used to identify a set of services that would start or stop when that runlevel was requested. Instead of runlevels, systemd uses the concept of *targets* to group together sets of services that are started or stopped. A target can also include other targets (for example, the multi-user target includes an nfs target).
- Depending on your default init level setting, the system will execute the programs from one of the following directories.

Traditional runlevel	New Target name	Symbolically linked to...
Runlevel 0	runlevel0.target	poweroff.target
Runlevel 1	runlevel1.target	rescue.target
Runlevel 2	runlevel2.target	multi-user.target
Runlevel 3	runlevel3.target	multi-user.target
Runlevel 4	runlevel4.target	multi-user.target
Runlevel 5	runlevel5.target	graphical.target
Runlevel 6	runlevel6.target	reboot.target

To know more about targets, pl read /etc/inittab file

### LAB WORK:-

#### To check the default run level in linux

- To see the default run level in linux the command is  
**#who -r**

```
[root@localhost ~]# who -r
  run-level 5 2016-08-02 09:06
[root@localhost ~]#
```

### Changing the default run level/target

- To check the default run level

```
#systemctl get-default
```

```
[root@localhost ~]# systemctl get-default  
graphical.target
```

- To change the default target/runlevel

```
#systemctl set-default multiuser.target
```

```
[root@localhost ~]# systemctl set-default multi-user.target  
rm '/etc/systemd/system/default.target'  
ln -s '/usr/lib/systemd/system/multi-user.target' '/etc/systemd/system/default.target'  
[root@localhost ~]#
```

Now reboot the system and check in which runlevel it is.

```
#reboot
```

```
Red Hat Enterprise Linux Server 7.1 (Maipo)  
Kernel 3.10.0-229.el7.x86_64 on an x86_64  
  
localhost login: root  
Password:  
Last login: Wed Aug 3 13:55:42 on :1  
[root@localhost ~]# who -r  
run-level 3 2016-08-03 14:40  
[root@localhost ~]# _
```

- To start the graphical interface when you are in runlevel 3, use the following command

```
#startx
```

### To see the details regarding the kernel installed

- To see the version of the kernel use

```
#uname -r
```

```
[root@localhost ~]# uname -r  
3.10.0-229.el7.x86_64  
[root@localhost ~]#
```

- To see the same thing with more details use

```
#uname -a
```

```
Linux localhost.localdomain 3.10.0-229.el7.x86_64 #1 SMP Thu Jan 29 18:37:38 EST 2015 x86_64 x86_64  
x86_64 GNU/Linux  
[root@localhost ~]#
```

Note: The same information can be seen in /boot/grub2/grub2.cfg

```
fi  
linux16 /vmlinuz-3.10.0-229.el7.x86_64 root=UUID=d06af8d4-c931-4412-aa1-674309315701  
nol�=auto rhgb quiet LANG=en_IN.UTF-8  
initrd16 /initramfs-3.10.0-229.el7.x86_64.img
```

### To change the default values in grub2 configuration file

- Changing the default timeout value in grub2.cfg
- Check the present timeout value with following command

```
[root@mlinux1 ~]# grep -i timeout /boot/grub2/grub.cfg
if [ x$feature_timeout_style = xy ] ; then
    set timeout_style=menu
    set timeout=5
# Fallback normal timeout code in case the timeout_style feature is
    set timeout=5
[root@mlinux1 ~]#
```

- Make the changes in /etc/default/grub file as per the requirement

```
[root@mlinux1 ~]# vim /etc/default/grub
GRUB_TIMEOUT=9
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

- Update the change in grub2 config file by using following command
- #grub2-mkconfig -o /boot/grub2/grub.cfg

```
[root@mlinux1 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-327.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-71773719b0e6437abb6a06811abed8c4
Found initrd image: /boot/initramfs-0-rescue-71773719b0e6437abb6a06811abed8c4
g
done
```

- Check back in grub2 config file whether the changes are applied

```
[root@mlinux1 ~]# grep -i timeout /boot/grub2/grub.cfg
if [ x$feature_timeout_style = xy ] ; then
    set timeout_style=menu
    set timeout=9
# Fallback normal timeout code in case the timeout_style feature is
    set timeout=9
[root@mlinux1 ~]#
```

### To check the architecture of the O/S

- To check the architecture of the O/S the command is

```
#arch
#uname -m
```

```
[root@ktadm Desktop]# arch
x86_64
[root@ktadm Desktop]# uname -m
x86_64
[root@ktadm Desktop]#
```

### To check the version of the O/S in the system

- To check the O/S version you have to navigate to the following file  

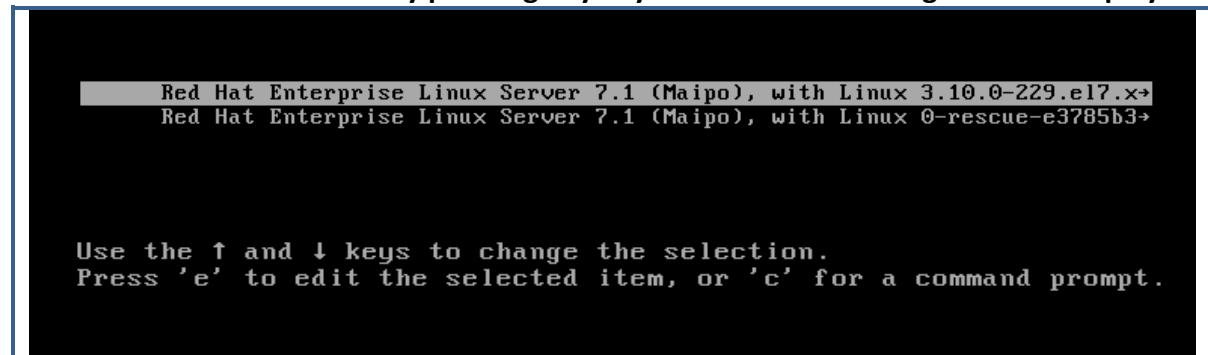
```
# cat /etc/redhat-release or # cat /etc/os-release
```

```
[root@localhost ~]# cat /etc/redhat-release
Red Hat Enterprise Linux release 8.1 (Ootpa)
```

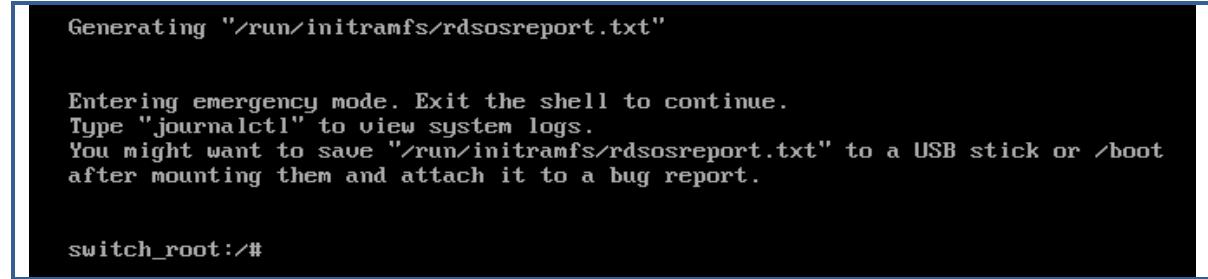
### Recovering the lost password in RHEL 7/8

To recover the password the steps are

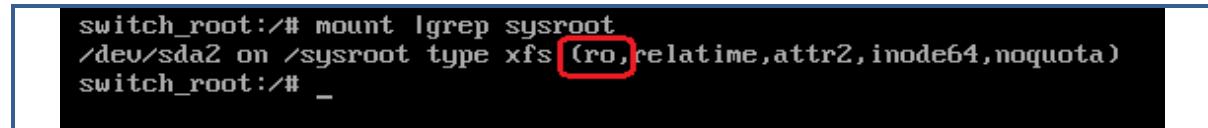
- Disturb the normal boot by pressing any key when RHEL 7 booting screen is displayed



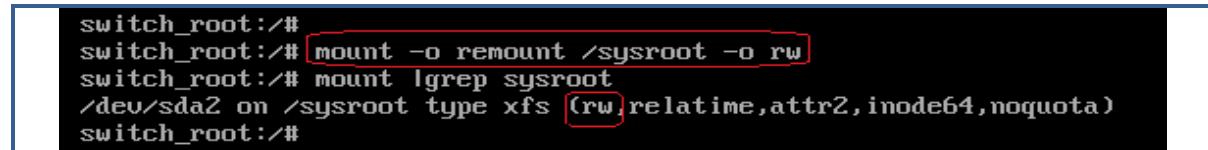
- Type "e" to edit the kernel argument
- Go to the end of kernel line "linux16" using ctrl+e, type rd.break. To continue booting use ctrl+x



- It will boot into emergency mode



- Check in which mode /sysroot is mounted, it would be in read-only mode



- Change the /sysroot to rw, by using above command

```
switch_root:/#  
switch_root:/# chroot /sysroot  
sh-4.2# ls  
bin boot dev etc home ktdir lib lib64 media mnt opt proc root run sbin srv sys tmp usr var  
sh-4.2#
```

- Now access the /sysroot using #chroot command

```
sh-4.2# passwd  
Changing password for user root.  
New password:  
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic  
Retype new password:  
passwd: all authentication tokens updated successfully.  
sh-4.2#
```

- To change the password use command #passwd and change the passwd

```
sh-4.2#  
sh-4.2# touch /.autorelabel  
sh-4.2# _
```

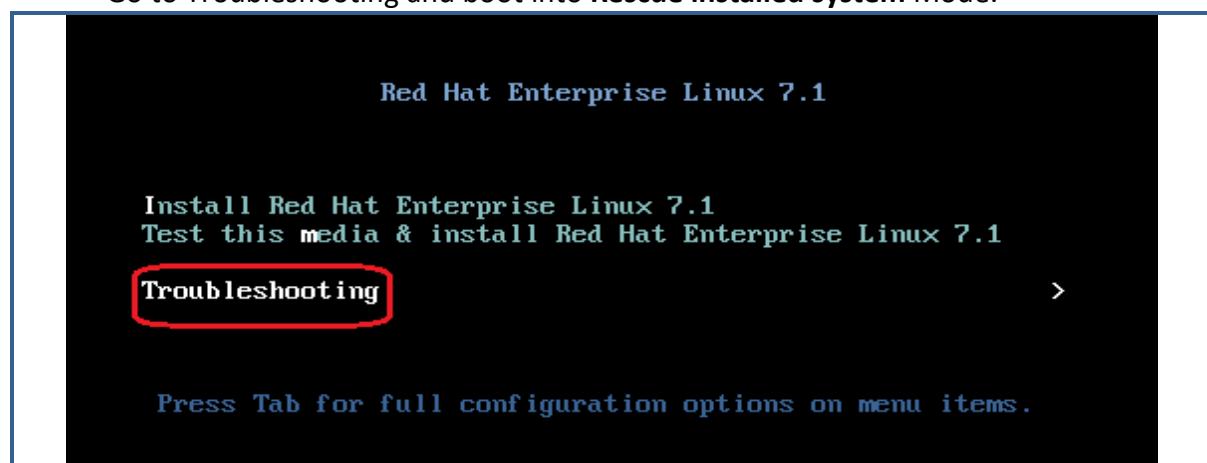
- To auto relabel selinux policies, create a blank hidden file “/.autorelabel”

```
sh-4.2# exit  
exit  
switch_root:/# reboot_
```

- Okay, Now we have successfully changed the password, type exit to leave the /sysroot and reboot the system. After reboot try the new password for root user.

## Repairing the corrupted boot loader and recovering it

- There might be a situation where your boot loader i.e., GRub might got corrupted and you want to recover it or in other word repair it. Basically the repairing of GRub means installing a new grub on the existing one from RHEL 7/8 installation media/DVD.
- To recover the grub the steps are:
  - Boot the system with RHEL 7/8 DVD
  - Go to Troubleshooting and boot into Rescue installed system Mode.



**Troubleshooting**

Install Red Hat Enterprise Linux 7.1 in basic graphics mode  
**Rescue a Red Hat Enterprise Linux system**  
 Run a memory test

Boot from local drive

Return to main menu

&lt;

Press Tab for full configuration options on menu items.

The rescue environment will now attempt to find your Linux installation and mount it under the directory : /mnt/sysimage. You can then make any changes required to your system. Choose '1' to proceed with this step.  
 You can choose to mount your file systems read-only instead of read-write by choosing '2'.  
 If for some reason this process does not work choose '3' to skip directly to a shell.

**1) Continue**

2) Read-only mount

3) Skip to shell

4) Quit (Reboot)

Please make a selection from the above: **1**   
 [anaconda 1:main\* 2:shell 3:log 4:storage-1o> Switch tab: Alt+Tab | Help: F1

- Move the cursor to “continue” tab, and hit enter

=====  
 ======  
 ======  
 ======  
 ======  
 ======  
 ======

Rescue Mount

Your system has been mounted under **/mnt/sysimage**.

If you would like to make your system the root environment, run the command:

**chroot /mnt/sysimage**

Your system is mounted under the /mnt/sysimage directory.  
 Please press <return> to get a shell.

[anaconda 1:main\* 2:shell 3:log 4:storage-1o> Switch tab: Alt+Tab | Help: F1

- Observe from above pic, that now your system has been mounted on /mnt/sysimage. It means where our system root is residing
- Press Enter to continue.

Starting installer, one moment...  
 anaconda 19.31.123-1 for Red Hat Enterprise Linux 7.1 started.  
 \* installation log files are stored in /tmp during the installation  
 \* shell is available on TTY2  
 \* if the graphical installation interface fails to start, try again with the inst.text bootoption to start text installation  
 \* when reporting a bug add logs from /tmp as separate text/plain attachments

Your system is mounted under the /mnt/sysimage directory.  
 When finished please exit from the shell and your system will reboot.

sh-4.2#

```
sh-4.2# chroot /mnt/sysimage/  
bash-4.2# _
```

- Change the DVD root to system root in order to access OS with root credentials by using following command

```
#chroot /mnt/sysimage
```

```
bash-4.2# grub2-install /dev/sda  
Installing for i386-pc platform.  
Installation finished. No error reported.  
bash-4.2# _
```

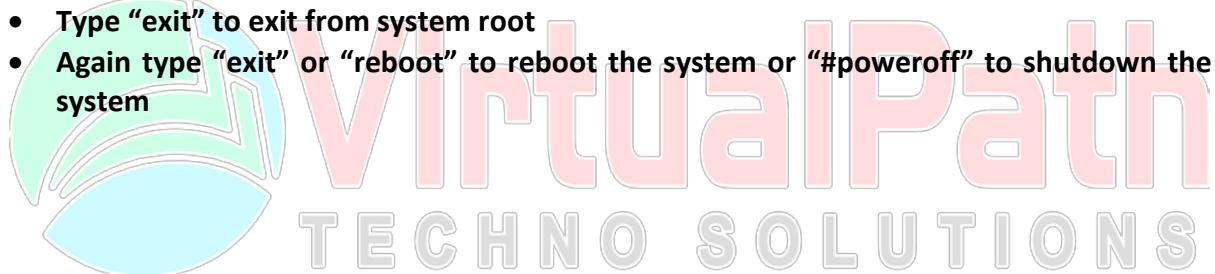
- Install the grub on the /boot device i.e. /dev/sda by using following command

```
#grub2-install <device name>  
#grub2-install /dev/sda
```

- If it shows no error reported, that means we have successfully recovered the grub.

```
sh-4.1# exit  
exit  
bash-4.1# reboot_
```

- Type “exit” to exit from system root
- Again type “exit” or “reboot” to reboot the system or “#poweroff” to shutdown the system



*All these steps are very critical and highly useful, please practice the stuff very well.*

## MANAGING INSTALLED SERVICES

- Services are programs (called daemons) that once started run continuously in the background and are ready for input or monitor changes in your computer and respond to them. For example the Apache server has a daemon called **httpd** (the d is for daemon) that listens on port 80 on your computer and when it receives a request for a page it sends the appropriate data back to the client machine.
- Many services are required to run all the time however many can be safely turned off for both security reasons as running unnecessary services opens more doors into your computer, but also for performance reasons. It may not make much difference but your computer should boot slightly faster with less services it has to start on boot.
- One of the techniques in every Linux administrator's toolbox to improve security of a box is to turn off unneeded services.

### INTRODUCTION TO *systemd* and *systemctl* command in RHEL7/8

Systemd is a system and service manager for Linux operating systems. It is designed to be backwards compatible with SysV init scripts, and provides a number of features such as parallel startup of system services at boot time, on-demand activation of daemons, support for system state snapshots, or dependency-based service control logic. In Red Hat Enterprise Linux 7, systemd replaces Upstart as the default init system.

Systemd is a replacement to the older traditional "System V init" system . systemd stands for system daemon. systemd was designed to allow for better handling of dependencies and have the ability to handle more work in parallel at system startup. systemd supports snapshotting of your system and the restoring of your systems state, keeps track of processes stored in what is known as a "cgroup" as opposed to the conventional "PID" method. systemd is now shipping by default with many popular Linux distributions such as Fedora, Mandriva, Mageia, Arch Linux, CentOS 7, RHEL 7.0 (Red Hat Enterprise Linux) and Oracle Linux 7.0. systemd refers to runlevels as targets.

Runlevel	Systemd Description
0	poweroff.target
1	rescue.target
2	multi-user.target
3	multi-user.target
4	multi-user.target
5	graphical.target
6	reboot.target

## Service vs systemctl commands

service	systemctl	Description
<b>service name start</b>	systemctl start name.service	Starts a service.
<b>service name stop</b>	systemctl stop name.service	Stops a service.
<b>service name restart</b>	systemctl restart name.service	Restarts a service.
<b>service name reload</b>	systemctl reload name.service	Reloads configuration.
<b>service name status</b>	systemctl status name.service	Check the status of the service
<b>service --status-all</b>	systemctl list-units --type service --all	Displays the status of all services.
	systemctl is-active name.service	Checks if a service is running.

## Chkconfig vs systemctl commands

chkconfig	systemctl	Description
<b>chkconfig --list name</b>	systemctl status name	Checks if a service is enabled
<b>chkconfig service on</b>	systemctl enable service	Enables a service
<b>chkconfig service off</b>	systemctl disable service	Disables a service
	Systemctl is-enable service	Checks if a service is enabled

## LAB WORK:-

### To check the status of ftp service “vsftpd”

- To check the status of the above service

```
#systemctl status vsftpd
```

```
[root@mlinux71 ~]# systemctl status vsftpd
vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled)
   Active: active (running) since Fri 2016-09-30 07:03:05 IST; 5h 6min ago
     Process: 1312 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited, status=0/SUCCESS)
    Main PID: 1351 (vsftpd)
      CGroup: /system.slice/vsftpd.service
              └─1351 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf

Sep 30 07:03:05 mlinux71.kt.com systemd[1]: Starting Vsftpd ftp daemon...
Sep 30 07:03:05 mlinux71.kt.com systemd[1]: Started Vsftpd ftp daemon.
[root@mlinux71 ~]#
```

### To stop the ftp services

- To start the ftp service, the command is

```
#systemctl stop vsftpd
```

```
[root@mlinux71 ~]# systemctl stop vsftpd
[root@mlinux71 ~]# systemctl status vsftpd
vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled)
   Active: inactive (dead) since Fri 2016-09-30 12:14:55 IST; 2s ago
     Process: 1312 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited, status=0/SUCCESS)
    Main PID: 1351 (code=killed, signal=TERM)

Sep 30 07:03:05 mlinux71.kt.com systemd[1]: Starting Vsftpd ftp daemon...
Sep 30 07:03:05 mlinux71.kt.com systemd[1]: Started Vsftpd ftp daemon.
Sep 30 12:14:55 mlinux71.kt.com systemd[1]: Stopping Vsftpd ftp daemon...
Sep 30 12:14:55 mlinux71.kt.com systemd[1]: Stopped Vsftpd ftp daemon.
[root@mlinux71 ~]#
```

### Reload the ftp services, may be required after doing some change in config file.

- To reload the service, the command is

```
#service crond reload
```

```
[root@mlinux71 ~]# systemctl reload crond
[root@mlinux71 ~]#
```

### To restart the ftp or any service required when reload does not work

- To restart the ftp services, the command will be

```
#systemctl restart vsftpd
```

```
[root@mlinux71 ~]# systemctl restart vsftpd.service
[root@mlinux71 ~]#
```

### Check the status of the service availability.

- To check the status of the service availability, use

```
#systemctl status vsftpd
```

```
[root@mlinux71 ~]# systemctl status vsftpd.service
vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; disabled)
   Active: active (running) since Fri 2016-09-30 12:25:53 IST; 3min 38s ago
     Main PID: 6986 (vsftpd)
       CGroup: /system.slice/vsftpd.service
               └─6986 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf

Sep 30 12:25:53 mlinux71.kt.com systemd[1]: Starting Vsftpd ftp daemon...
Sep 30 12:25:53 mlinux71.kt.com systemd[1]: Started Vsftpd ftp daemon.
[root@mlinux71 ~]#
```

### Make the service enable at boot for vsftpd .

- To make the service enable at boot vsftpd service,

```
#systemctl enable vsftpd.service
```

```
[root@mlinux71 ~]# systemctl enable vsftpd.service
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd,
[root@mlinux71 ~]# systemctl is-enabled vsftpd.service
enabled
[root@mlinux71 ~]#
```

### Make the service availability disabled for vsftpd

- To make the service availability disabled the command is  
**#systemctl disable vsftpd.service**

```
[root@mlinux71 ~]# systemctl disable vsftpd.service
rm '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
[root@mlinux71 ~]# systemctl is-enabled vsftpd.service
disabled
[root@mlinux71 ~]#
```

### To start and enable the service at the same time

- To make the service availability disabled the command is  
**#systemctl enable vsftpd --now**

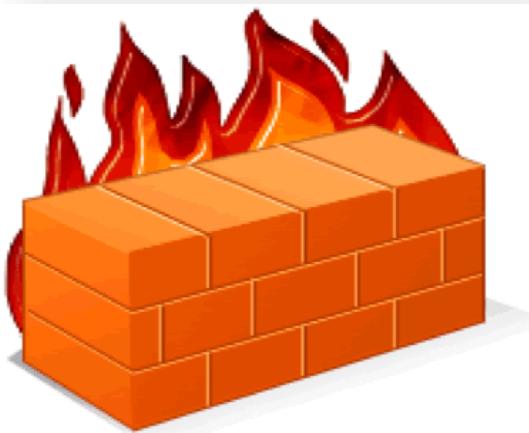
```
[root@mlinux1 ~]# systemctl enable vsftpd --now
Created symlink from /etc/systemd/system/multi-user.target.wants/vsftpd.service
service.
[root@mlinux1 ~]# systemctl status vsftpd
● vsftpd.service - Vsftpd ftp daemon
  Loaded: loaded (/usr/lib/systemd/system/vsftpd.service); enabled; vendor prese
  Active: active (running) since Sat 2020-03-28 11:35:58 IST; 27s ago
    Process: 2385 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited,
   Main PID: 2386 (vsftpd)
     Tasks: 1
```

### To stop and disable the service at the same time

- To make the service availability disabled the command is  
**#systemctl disable vsftpd --now**

```
[root@mlinux1 ~]# systemctl status vsftpd
● vsftpd.service - Vsftpd ftp daemon
  Loaded: loaded (/usr/lib/systemd/system/vsftpd.service); disabled;
  Active: inactive (dead)
```

## INTRODUCTION TO FIREWALL



- A firewall is a device that allows multiple networks to communicate with one another according to a defined security policy. They are used when there is a need for networks of varying levels of trust to communicate with one another. For example, a firewall typically exists between a corporate network and a public network like the Internet. It can also be used inside a private network to limit access to different parts of the network. Wherever there are different levels of trust among the different parts of a network, a firewall can and should be used.
- A firewall is a device that allows multiple networks to communicate with one another according to a defined security policy. They are used when there is a need for networks of varying levels of trust to communicate with one another. For example, a firewall typically exists between a corporate network and a public network like the Internet. It can also be used inside a private network to limit access to different parts of the network. Wherever there are different levels of trust among the different parts of a network, a firewall can and should be used.

**LAB WORK:**

To check the status of firewalld status

#firewall-cmd --state

```
[root@mlinux71 ~]# firewall-cmd --state  
running  
[root@mlinux71 ~]#
```

To check the list of trusted services and ports in firewalld

#firewall-cmd --list-all

```
[root@mlinux71 ~]# firewall-cmd --list-all  
public (default, active)  
  interfaces: bond0 ens3 ens8  
  sources:  
  services: dhcpcv6-client ssh  
  ports:  
  masquerade: no  
  forward-ports:  
  icmp-blocks:  
  rich rules:
```

To add a service into trusted list permanently

#firewall-cmd --add-service=<service name> --permanent (ex ftp service)

```
[root@mlinux71 ~]# firewall-cmd --add-service=ftp --permanent  
success  
[root@mlinux71 ~]#
```

The service will not be updated until reloading of firewall

To reload firewalld service

#firewall-cmd --reload

```
[root@mlinux71 ~]#  
[root@mlinux71 ~]# firewall-cmd --reload  
success  
[root@mlinux71 ~]# firewall-cmd --list-all  
public (default, active)  
  interfaces: bond0 ens3 ens8  
  sources:  
  services: dhcpcv6-client ftp ssh  
  ports:  
  masquerade: no  
  forward-ports:  
  icmp-blocks:  
  rich rules:
```

To remove the service from firewall trusted list permanently

```
#firewall-cmd --remove-service=ftp --permanent
#firewall-cmd --reload
```

```
[root@mlinux71 ~]# firewall-cmd --remove-service=ftp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

To add a port into firewall trusted lists permanently

```
#firewall-cmd --add-port=21/tcp --permanent
#firewall-cmd --reload
```

```
[root@mlinux71 ~]# firewall-cmd --add-port=21/tcp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpv6-client ssh
  ports: 21/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```



To remove a port from firewall trusted lists permanently

```
#firewall-cmd --remove-port=21/tcp --permanent
#firewall-cmd --reload
```

```
[root@mlinux71 ~]# firewall-cmd --remove-port=21/tcp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

To start the firewall permanently

```
#systemctl start firewalld.service
#systemctl enable firewalld.service
```

```
[root@mlinux71 ~]# systemctl is-active firewalld.service
unknown
[root@mlinux71 ~]# systemctl start firewalld.service
[root@mlinux71 ~]# systemctl is-active firewalld.service
active
[root@mlinux71 ~]# systemctl enable firewalld.service
ln -s '/usr/lib/systemd/system/firewalld.service' '/etc/systemd/
fedoraproject.FirewallD1.service'
ln -s '/usr/lib/systemd/system/firewalld.service' '/etc/systemd/
get.wants/firewalld.service'
[root@mlinux71 ~]# systemctl is-enabled firewalld.service
enabled
[root@mlinux71 ~]#
```

To stop the firewall permanently

```
#systemctl stop firewalld.service
#systemctl disable firewalld.service
```

```
[root@mlinux71 ~]# systemctl is-active firewalld.service
active
[root@mlinux71 ~]# systemctl stop firewalld.service
[root@mlinux71 ~]# systemctl is-active firewalld.service
inactive
[root@mlinux71 ~]#
[root@mlinux71 ~]# systemctl is-enabled firewalld.service
enabled
[root@mlinux71 ~]# systemctl disable firewalld.service
rm '/etc/systemd/system/basic.target.wants/firewalld.service'
rm '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
[root@mlinux71 ~]# systemctl is-enabled firewalld.service
disabled
[root@mlinux71 ~]#
```

To add or remove multiple services/ports in firewall use following command

```
#firewall-cmd --add-service=1st service --add-service=2nd service –permanent
#firewall-cmd --reload
```

```
#firewall-cmd --add-port=portid/protocol --add-port=portid/protocol –permanent
#firewall-cmd --reload
```

## INTRODUCTION TO COCKPIT IN RHEL8

- The Cockpit is a free and open source web-based server management tool. By default, Cockpit comes preinstalled on an RHEL 8 server. But, it is not activated. A sysadmin must enable it. One can see the server in a web browser and perform system tasks with a GUI/mouse. It is easy to start containers, administer storage or users, configure networks, and inspect log files on RHEL 8. The Cockpit web interface is user-friendly for new to Linux users and seasoned sysadmins too.
- Cockpit is a useful Web based GUI tool through which sysadmins can monitor and manage their Linux servers, it can also be used to manage networking and storage on servers, containers, virtual machines and inspections of system and application's logs.

### **Activating cockpit services:**

As mentioned above cockpit tool comes pre-installed in RHEL8 and hence we don't have to install any package for it. The only thing we need to do is to activate the services related to cockpit and allow it through firewall.

It can be observed that while we login into any rhel8 machine, by default it gives a message on the screen related to cockpit, as shown below.

```
login as: root
root@192.168.10.30's password:
Activate the web console with: systemctl enable --now cockpit.socket

This system is not registered to Red Hat Insights. See https://cloud.redhat.com/
To register this system, run: insights-client --register
```

In order to use cockpit start the service of cockpit and check whether in firewall the services is allowed or not.

**#systemctl enable --now cockpit.socket**

```
[root@mlinux3 ~]# systemctl enable cockpit.socket --now
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/systemd/s
[root@mlinux3 ~]# systemctl status cockpit.socket
● cockpit.socket - Cockpit Web Service Socket
   Loaded: loaded (/usr/lib/systemd/system/cockpit.socket; enabled; vendor preset: disabled)
   Active: active (listening) since Sun 2020-04-19 21:32:52 IST; 6s ago
     Docs: man:cockpit-ws(8)
     Listen: [::]:9090 (Stream)
   Process: 3221 ExecStartPost=/bin/ln -snf active.motd /run/cockpit/motd (code=exited, statu
   Process: 3214 ExecStartPost=/usr/share/cockpit/motd/update-motd localhost (code=exited, s
     Tasks: 0 (limit: 11501)
    Memory: 312.0K
      CGroup: /system.slice/cockpit.socket

Apr 19 21:32:52 mlinux3.vpts.com systemd[1]: Starting Cockpit Web Service Socket.
Apr 19 21:32:52 mlinux3.vpts.com systemd[1]: Listening on Cockpit Web Service Socket.
```

Check whether **cockpit** is allowed in firewall or not. If not, allow cockpit in firewall  
**#firewall-cmd --list-all**

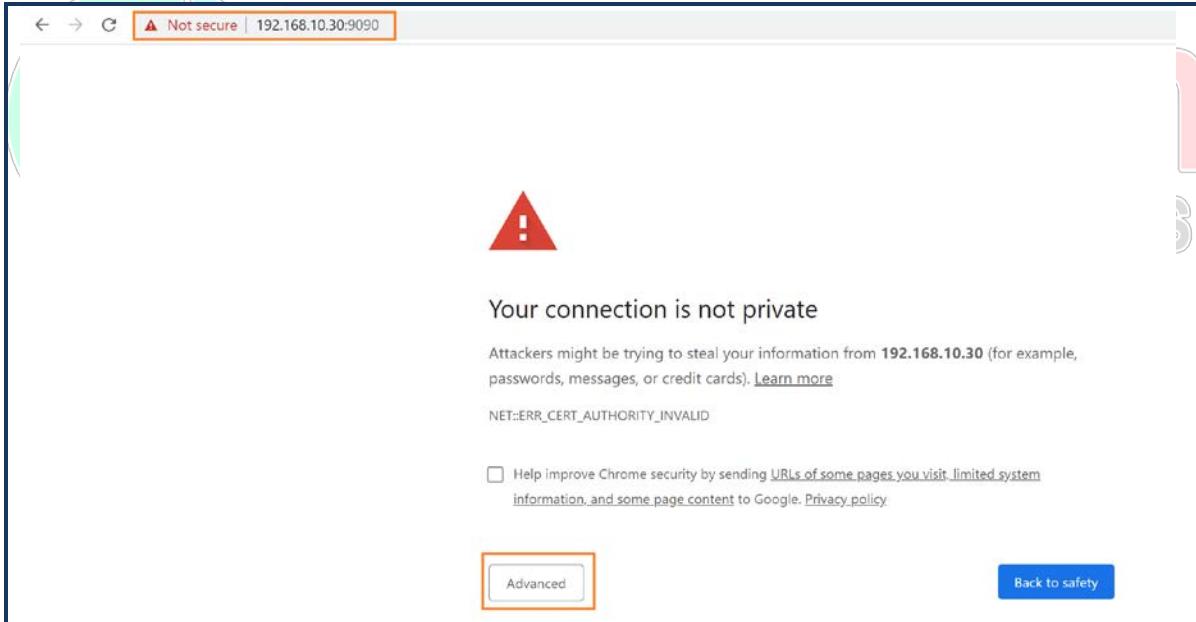
```
[root@mlinux3 ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit dhcpcv6-client dns ftp http mounted
  ports: 8080/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

By default **cockpit** is pre-allowed in firewall. If it is not pre-allowed, then it can be added later by using following commands

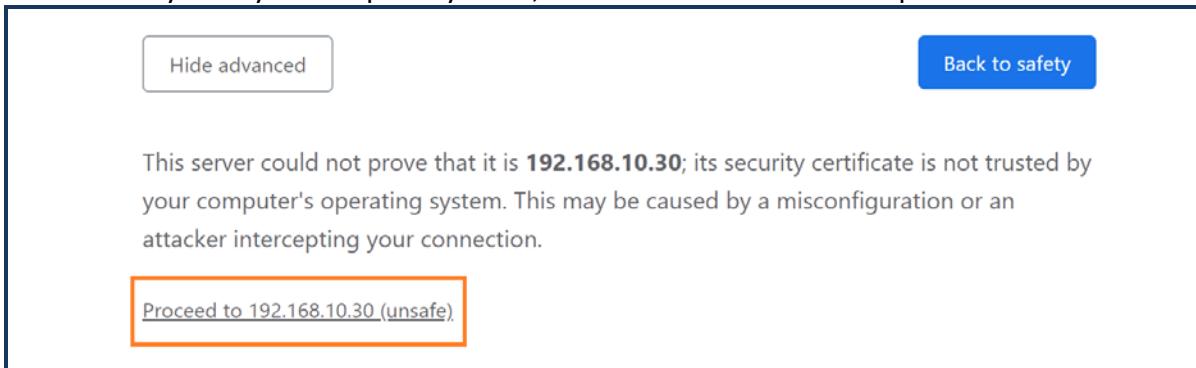
**#firewall-cmd --add-service=cockpit –permanent**  
**#firewall-cmd –reload**

Now then the cockpit services are active and allowed in firewall, we can now access it through a browser using the following URL

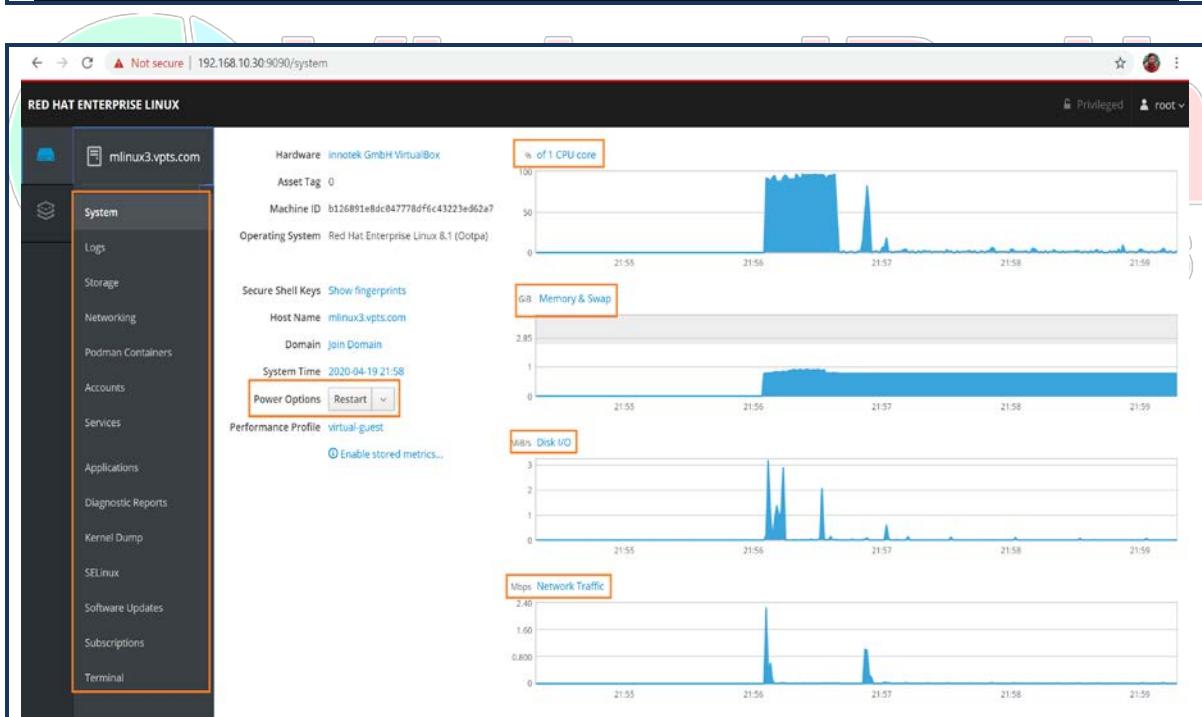
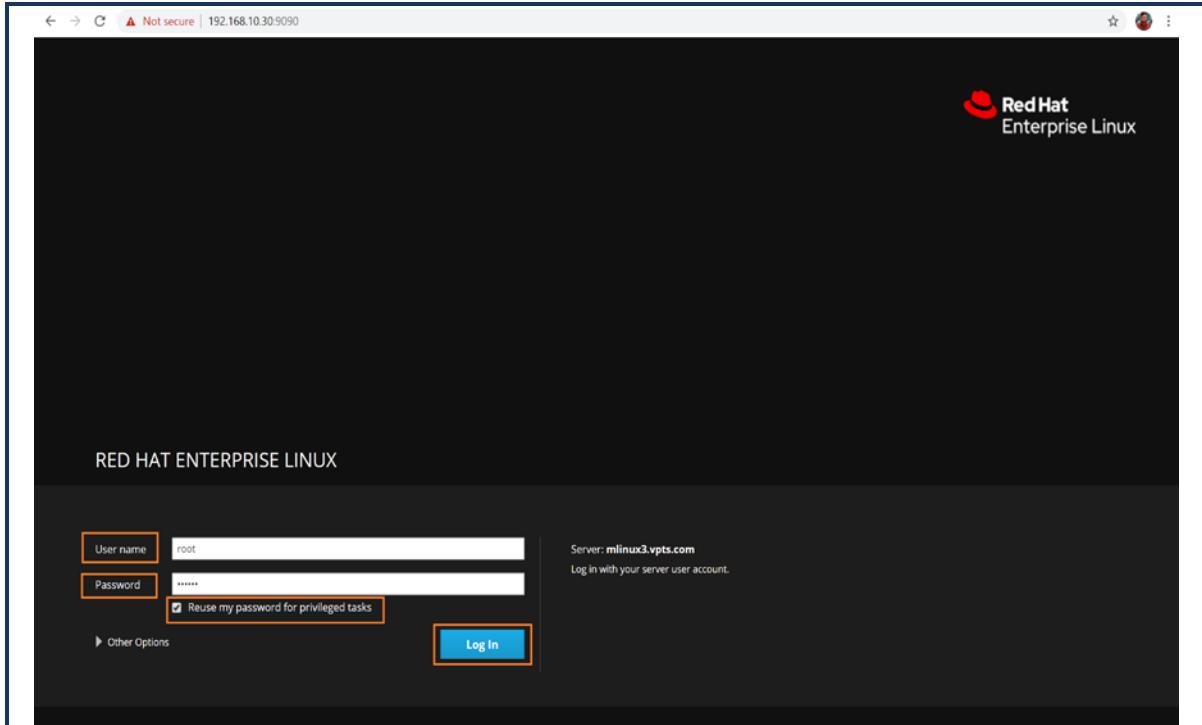
<https://<server IP>:9090>



Note: Initially it may reflect privacy error, click on **Advanced tab** and proceed to continue



Type root or any other user credentials to login with

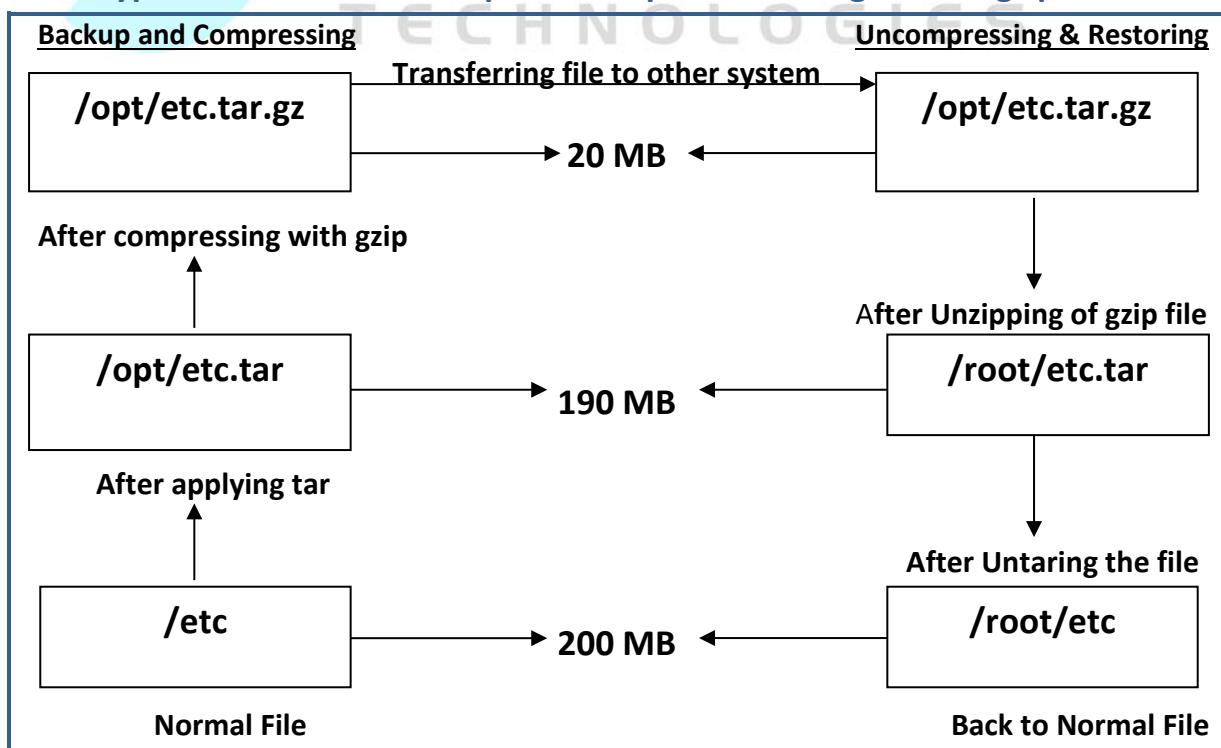


As you can see there are many options which can be explored to do various activities. Go ahead and explore tabs for more activities, in-fact the many important activities can be controlled or performed using cockpit.

## BACKUP AND RESTORE

- In information technology, a **backup** or the process of **backing up** is making copies of data which may be used to *restore* the original after a data loss event.
- Backups have two distinct purposes
- The primary purpose is to recover data after its loss, be it by data deletion or corruption. Data loss is a very common experience of computer users. 67% of Internet users have suffered serious data loss.
- The secondary purpose of backups is to recover data from an earlier time, according to a user-defined data retention policy, typically configured within a backup application for how long copies of data are required.
- Backup is the most important job of a system administrator, as a system admin it is your duty to take backup of the data every day.
- Many companies have gone out of the market because of poor backup planning.
- The easiest way to back up your files is just copying. But if you have too many files to backup, copying and restoring may take too long time and it is not convenient. If there is a tool that can put many files into one file, the world will be better. Fortunately, 'tar' is used to create archive files. It can pack files or directories into a 'tar' file. It is like WinZip in Windows, without much compression.
- The **gzip** program compresses a single file. One important thing to remember about **gzip** is that, unlike **tar**, it replaces your original file with a compressed version. (The amount of compression varies with the type of data, but a typical text file will be reduced by 70 to 80 percent.)

### A Typical scenario of backup and compression using tar and gzip



## LAB WORK:-

### To backup the file using tar

- To backup the file using tar the syntax is

```
#tar -cvf <destination and name to be > < source file>
#tar -cvf /opt/etc.tar /etc
```

```
[root@ linux ~]# tar -cvf /opt/etc.tar /etc
/etc/rc.d/rc0.d/K95firstboot
/etc/rc.d/rc0.d/K89iscsid
/etc/rc.d/rc0.d/K92iptables
/etc/rc.d/rc0.d/K50snmpd
/etc/rc.d/rc0.d/K03rhnsd
/etc/rc.d/rc0.d/K15httpd
/etc/rc.d/rc0.d/K80sssd
/etc/rc.d/rc0.d/K99microcode_ctl
/etc/rc.d/rc0.d/K83rpcgssd
/etc/rc.d/rc0.d/K84NetworkManager
/etc/rc.d/rc0.d/K50vsftpd
/etc/rc.d/rc0.d/K74nscd
/etc/rc.d/rc0.d/K83bluetooth
/etc/rc.d/rc0.d/K01smartd
/etc/rc.d/rc0.d/K02oddjobd
```

- Check the size of tar file by using du -h <file name > command

```
#du -h /opt/etc.tar
```

```
[root@ linux ~]# du -h /opt/etc.tar
29M    /opt/etc.tar
[root@ linux ~]#
```

### Apply gzip on tar file and check the size.

- To apply gzip on a tar file, the syntax is

```
#gzip <file name>
#gzip /opt/etc.tar
```

```
[root@ linux ~]# gzip /opt/etc.tar
[root@ linux ~]#
```

- Now check the size of the file

```
[root@ linux ~]# cd /opt/
[root@ linux opt]# ls
etc.tar.gz  home  lost+found
[root@ linux opt]# du etc.tar.gz
7544    etc.tar.gz
[root@ linux opt]#
```

**Transfer the file to other system and remove gzip and tar from it and check the size on every step.**

- Let's transfer the file to other computer using scp

```
#scp /opt/etc.tar.gz 192.168.10.95:/root/
```

```
[root@ linux ~]# scp /opt/etc.tar.gz 192.168.10.95:/root/
etc.tar.gz                                         100% 7544KB   7.4MB/s  00:01
[root@ linux ~]#
```

- Login to the remote system, remove gzip it and check the size.
- To gunzip a file the syntax is

```
#gunzip <file name>
```

```
#gunzip etc.tar.gz
```

```
[root@ cl5 ~]# ls
anaconda-ks.cfg  Documents  etc.tar.gz  install.log.syslog  ktfile
Desktop          Downloads  install.log  ktdir                  Music
[root@ cl5 ~]# du -h etc.tar.gz
7.4M  etc.tar.gz
[root@ cl5 ~]# gunzip etc.tar.gz
[root@ cl5 ~]# ls
anaconda-ks.cfg  Documents  etc.tar      install.log.syslog  ktfile
Desktop          Downloads  install.log  ktdir                  Music
[root@ cl5 ~]# du -h etc.tar
29M  etc.tar
[root@ cl5 ~]#
```

**Untar the file and check for the size of the file/directory**

- To untar a file the syntax is

```
#tar -xvf <file name>
```

```
#tar -xvf etc.tar
```

```
[root@ cl5 ~]# tar -xvf etc.tar
|etc/sgml/xml-docbook-4.1.2-1.0-51.el6.cat
|etc/sgml/sgml-docbook-4.3-1.0-51.el6.cat
|etc/sgml/sgml.conf
|etc/sgml/xml-docbook.cat
|etc/sgml/sgml-docbook-4.1-1.0-51.el6.cat
[root@ cl5 ~]# ls
anaconda-ks.cfg  Downloads  install.log
Desktop          etc      install.log.syslog
Documents        etc.tar    ktdir
[root@ cl5 ~]# du -h etc
8.0K  etc/avahi/etc
8.0K  etc/avahi/services
32K   etc/avahi
4.0K   etc/openldap/cacerts
12K   etc/openldap
```

- To un-tar without unzipping, use #tar -zxvf (file name)
- To tar and zip together use #tar -zcvf destination and source.
- To view the content of tar file # tar -tvf filename.

## JOB AUTOMATION

### Automation with cron

- In any operating system, it is possible to create jobs that you want to reoccur. This process, known as **job scheduling**, is usually done based on user-defined jobs. For Red Hat or any other Linux, this process is handled by the cron service or a daemon called **crond**, which can be used to schedule tasks (also called *jobs*). By default, Red Hat comes with a set of predefined jobs that occur on the system (hourly, daily, weekly, monthly, and with arbitrary periodicity). As an administrator, however, you can define your own jobs and allow your users to create them as well.
- The importance of the job scheduling is that the critical tasks like taking backups, which the clients usually wants to be taken in nights, can easily be performed without the intervention of the administrator by scheduling a cron job. If the cron job is scheduled carefully than the backup will be taken at any given time of the client and there will be no need for the administrator to remain back at nights to take the backup.

#### Important Files related to cron and at

- **/etc/crontab** is the file which contains cron job format
- **/var/spool/cron/UserName** : contains job scheduled by users
- **/etc/cron.deny** is the file used to restrict the users from using cron jobs.
- **/etc/cron.allow** is used to allow only users whose names are mentioned in this file to use cron jobs. (this file does not exist by default)
- **/var/log/cron** is the log file for cron jobs

#### Crontab format

- To assign a job in the Crontab file the format used is the following



Options	Description
*	Is treated as a wild card. Meaning any possible value.
*/5	Is treated as every 5 minutes, hours, days, or months. Replacing the 5 with another numerical value will change this option.
2,4,6	Treated as an OR, so if placed in the hours, this could mean at 2, 4, or 6 o'clock.
9-17	Treats for any value between 9 and 17. So if placed in day of month this would be days 9 through 17. Or if put in hours it would be between 9 and 5.

### Crontab Commands

Command	Explanation
crontab -e	Edit your crontab file, or create one if it doesn't already exist.
crontab -l	Display your crontab file.
crontab -r	Remove your crontab file.
crontab -u	If combined with -e, edit a particular user's Crontab file and if combined with -l, display a particular user's crontab file. If combined with -r, deletes a particular user's Crontab file

### LAB WORK:-

To check the assigned cron jobs of currently logged in user

- To check the cron jobs the command is  
**#crontab -l**

```
[root@ linux ~]# crontab -l
no crontab for root
[root@ linux ~]#
```

To check the cron jobs of a particular user

- To check a user's cron jobs, the syntax is  
**#crontab -l -u <user name>**  
**#crontab -l -u myuser**

```
[root@ linux ~]# crontab -l -u myuser
no crontab for myuser
[root@ linux ~]# crontab -lu myuser
no crontab for myuser
[root@ linux ~]#
```

### Setting a job to display the current date for every minute on present console

- To set the above job the steps are
- Check the console on which you are working by following command  
**#tty**

```
[root@ linux ~]# tty
/dev/pts/1
[root@ linux ~]#
```

**Note:** /dev/pts/1 is the console address

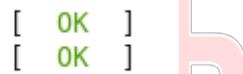
- Schedule the task as shown below  
**#crontab -e** and enter the field as shown below and save it as in **VI editor**

```
*/1 * * * * date > /dev/pts/1
~[root@ linux ~]# crontab -e
crontab: installing new crontab
```

**Note:** where \* means every possible value.

- Restart the cron services  
**#service crond restart (RHEL6) or #systemctl restart crond (RHEL7)**

```
[root@ linux ~]# service crond restart
Stopping crond:
Starting crond:
[root@ linux ~]#
```



- Wait for a minute and check whether time is displaying or not. Every min time will be displayed as below.

```
[root@ linux ~]# Thu Oct 13 15:24:01 IST 2011
Thu Oct 13 15:25:01 IST 2011
Thu Oct 13 15:26:01 IST 2011
```

### Schedule a cron job to create a directory “mydir” under “/root” on “Sunday 22 October at 1:30 AM”

- To schedule above job edit the crontab file as shown below and restart the service  
**#crontab -e**

```
30 1 22 10 0 mkdir /root/mydir
~
:wq!#
[root@ linux ~]# crontab -e
crontab: installing new crontab
[root@ linux ~]#
```

**Note:** you can use 0 or 7 for Sunday.

Check whether it got created or not on scheduled day, if it created you can see the directory otherwise an error mail will be generated to your mail.

### Schedule a job to run the backup script “bkpscript.sh” on every “Saturday 12:30 PM”

- In order to schedule above job the steps are.
- Check the location of script and also check whether it is having execute permission or not. If not then add the execute permissions to all user on it.

```
[root@ linux ~]# ls
anaconda-ks.cfg Desktop Downloads install.log.syslog
bkpscript.sh Documents install.log ktdir
[root@ linux ~]# pwd
/root
[root@ linux ~]# ls -l bkpscript.sh
-rw-r--r--. 1 root root 0 Oct 13 15:47 bkpscript.sh
[root@ linux ~]# chmod a+x bkpscript.sh
[root@ linux ~]# ls -l bkpscript.sh
-rwxr-xr-x. 1 root root 0 Oct 13 15:47 bkpscript.sh
[root@ linux ~]#
```

- Apply the job in **crontab**

```
#crontab -l
```

```
30 1 22 10 0 mkdir /root/mydir
30 12 * * 7 /root/bkpscript.sh
~  
:wq!
```

### Schedule a job so that a user “myuser” should get a mail regarding meeting on 24<sup>th</sup>, 29<sup>th</sup> and 31<sup>st</sup> October at 2:25 PM.

- To set above task edit the crontab in following passion, and restart the service

```
#crontab -e -u <user name>
#crontab -e -u myuser
```

```
#crontab -eu myuser
25 14 27,29,31 10 * echo "Meeting at 3:00 PM Today"
~  
:wq!
```

### Schedule a job so that a user “myuser” should get the mail from 15<sup>th</sup> to 20<sup>th</sup> and 25<sup>th</sup> to 30<sup>st</sup> November as a reminder of some session at 2:25 PM

- This task is very much similar to the previous one but there is only a small change in format.

```
#crontab -e -u myuser
```

```
#crontab -eu myuser
25 14 27,29,31 10 * echo "Meeting at 3:00 PM Today"
25 14 15-20,25-30 11 * echo "Class at study hall 3:00 PM Today"
~  
~  
:wq!
```

- There are still various method you can schedule the cron jobs, Do some **R&D** on it to find out more.

### Restrict users “myuser” “john” “sam” from using cron jobs

- To restrict any user from using cron job facility, enter their names in **/etc/cron.deny** and save it  

```
#vim /etc/cron.deny
```

```
myuser
john
sam
~
~
~
:wq!■
```

- Now login as one of those users and try to use crontab.

```
[root@ linux ~]# vim /etc/cron.deny
[root@ linux ~]# su - myuser
[ktuser@ linux ~]$ crontab -l
You (myuser) are not allowed to use this program (crontab)
See crontab(1) for more information
[myuser@ linux ~]$ crontab -e
You (myuser) are not allowed to use this program (crontab)
See crontab(1) for more information
[myuser@ linux ~]$ ■
```

- If you want to allow them to use cron job facilities again, remove their names from **/etc/cron.deny** file.

### Allow only two users “musab” and “rahul” to use cron jobs out of all the users in the system

- Assuming that we have 100 users in our system, putting all 98 names in **/etc/cron.deny** file is a time consuming process. Instead of that, we can create one more file **/etc/cron.allow**, in which we can assign names of those users who are allowed to use cron jobs.
- Remove the **/etc/cron.deny** file and create **/etc/cron.allow**, still if both files are existing **cron.allow** file will be having precedence over **cron.deny** file. Just to avoid confusion it is good to remove **cron.deny** file

**Note:** **/etc/cron.deny** file exists by default, but we need to create **/cron.allow** file. If your name is not there in **cron.allow** file then you will not be allowed to use cron jobs, and as mentioned above, if both files are existing **cron.allow** file will be having precedence over **cron.deny** file. If neither **cron.deny** nor **cron.allow** files exists then only **root** can use cron jobs.

- Now, let's put those two users "musab" and "rahul" name in **/etc/cron.allow** file and check the results.

```
#vim /etc/cron.allow
```

```
musab
rahul
~
~
:wq!■
[root@ linux /]# vim /etc/cron.allow
[root@ linux /]# rm -f /etc/cron.deny
[root@ linux /]# su - vivek
[vivek@ linux ~]$ crontab -l
You (vivek) are not allowed to use this program (crontab)
See crontab(1) for more information
[vivek@ linux ~]$ exit
logout
[root@ linux /]# su - musab
[musab@ linux ~]$ crontab -l
no crontab for musab
[musab@ linux ~]$ exit
logout
[root@ linux /]# su - rahul
[rahul@ linux ~]$ crontab -l
no crontab for rahul
[rahul@ linux ~]$ exit
logout
[root@ linux /]# su -myuser
[myuser@ linux ~]$ crontab -l
You (myuser) are not allowed to use this program (crontab)
```



**Note:** To see man pages on cron job use **#man 4 crontabs** command

*All the above are few examples to use cron, do some constant R&D's to know more about it.*

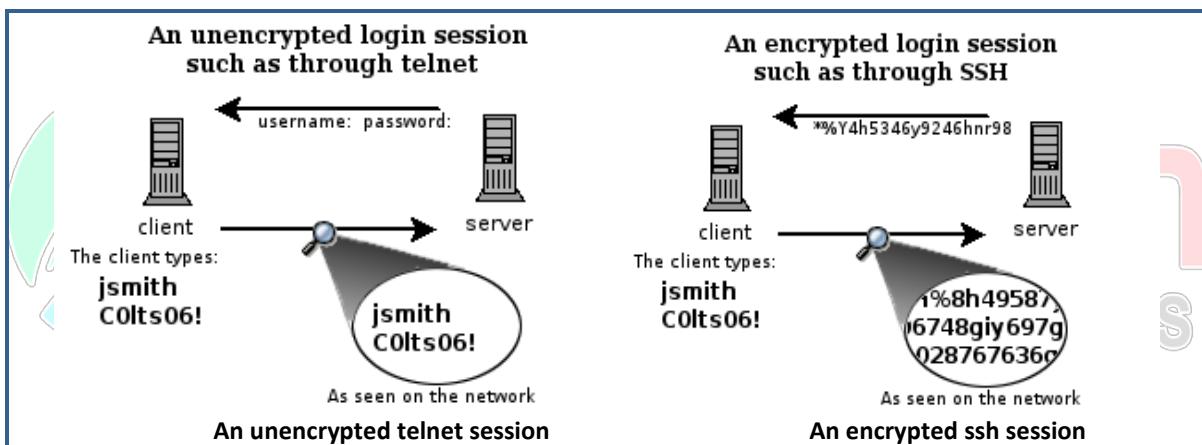
## ADMINISTRATING REMOTE SYSTEM

- Remote shell Access using SSH

### What Is SSH?

There are a couple of ways that you can access a shell (command line) remotely on most Linux/Unix systems. One of the older ways is to use the telnet program, which is available on most network capable operating systems. Accessing a shell account through the telnet method though poses a danger in that everything that you send or receive over that telnet session is visible in plain text on your local network, and the local network of the machine you are connecting to. So anyone who can "sniff" the connection in-between can see your username, password, email that you read, and command that you run. For these reasons you need a more sophisticated program than telnet to connect to a remote host.

SSH, which is an acronym for Secure SHell, was designed and created to provide the best security when accessing another computer remotely. Not only does it encrypt the session, it also provides better authentication facilities.



These two diagrams above show how a telnet session can be viewed by anyone on the network by using a sniffing program like Ethereal (now called Wireshark) or tcpdump. It is really rather trivial to do this and so anyone on the network can steal your passwords and other information. The first diagram shows user jsmith logging in to a remote server through a telnet connection. He types his username jsmith and password Colts06! which are viewable by anyone who is using the same networks that he is using.

The second diagram shows how the data in an encrypted connection like SSH is encrypted on the network and so cannot be read by anyone who doesn't have the session-negotiated keys, which is just a fancy way of saying the data is scrambled. The server still can read the information, but only after negotiating the encrypted session with the client.

- SSH configuration file is **/etc/ssh/sshd\_config**
- SSH demon or service is **sshd**

## LAB WORK:-

### Accessing the remote machine using ssh

- To access the remote machine using ssh, the syntax is

**#ssh <ip address/ host name of remote machine>**

**Note:** hostname can only be used when the hostname is saved in **/etc/hosts** file or, if DNS is configured.

**#ssh 192.168.10.98**

```
[root@ linux .ssh]# ssh 192.168.10.95
The authenticity of host '192.168.10.95 (192.168.10.95)' can't be established.
RSA key fingerprint is 35:f4:34:b8:29:00:02:87:14:47:56:f5:bb:6b:4c:68.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.10.95' (RSA) to the list of known hosts.
```

The first time around it will ask you if you wish to add the remote host to a list of known\_hosts, go ahead and say **yes**.

- Enter the password of the remote system correctly, once logged in check hostname and ip address to confirm login.

```
root@192.168.10.95's password:
Last login: Sun Sep 4 02:42:54 2011 from 192.168.1.10
[root@ cl5 ~]# hostname
cl5.mb.com
[root@ cl5 ~]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:0C:29:97:79:78
          inet addr:192.168.10.95 Bcast:192.168.10.255 Mask:255.255.255.0
```

- To leave the session, just type exit or logout command and you will be back to your own machine through which you are logged in.

```
[root@ cl5 ~]# hostname
cl5.mb.com
[root@ cl5 ~]# exit
logout
Connection to 192.168.10.95 closed.
[root@ linux .ssh]# hostname
linux.mb.com
[root@ linux .ssh]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:0C:29:A4:5E:C8
          inet addr:192.168.10.98 Bcast:192.168.10.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fea4:5ec8/64 Scope:Link
```

## Trust Relationship / Password less login using SSH keys

- As a system administrator, one person will be assigned to manage many systems, for example one person has to manage more than 10 systems at a time. In this situation admin has to transfer some files from one system to another 9 systems or vice versa, for every login on remote system it will prompt for password. Even for transferring files for every transfer we need to enter the password.
- Above situation will be very annoying for system admin to type password for every step. Therefore SSH provides a best way to escape password prompting every now and then.
- By generating SSH keys, a public key and a private key, an admin can copy the public key into other system and done, it will work as authorized access from the admin's system. Now whenever we are logging from admin's system to other system in which we have stored the public key of admin's system, it will not prompt us for password and we can login to that system as many time as we want without being prompt for the password.
- SSH keys are an implementation of public-key cryptography. They solve the problem of brute-force password attacks by making them computationally impractical.
- Public key cryptography uses a **public key** to **encrypt data** and a **private key** to **decrypt it**.



- To generate the SSH key pair, the syntax is  
# ssh-keygen

```
[root@ linux ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): █
```

It will prompt above to mention the file where these keys shoud be stored, to keep its default directory just press “Enter”. The default location will be **/root/.ssh/** directory

```
[root@ linux ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): █
```

Now it will ask for passphrase, which will be used instead of password. The passphrase will only be asked every time you connect from other machine instead of its original password. Enter your desired passphrase twice, or leave it blank for no passphrase and press enter.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
14:e4:49:b2:c4:b5:a3:b9:31:3e:f4:d6:11:5f:29:ee root@ktlinux.kt.com
The key's randomart image is:
+---[ RSA 2048]---+
|       .00=
|       ..= +
|       . * . . 0
|       + . + 0
|       * S . 0
|      o = . 0
|      + o . E
|      o
+-----+
```

Okay, now our keys are successfully generated, go to **/root/.ssh/** directory and check for the keys.

```
#cd /root/.ssh
```

```
[root@ linux ~]# cd /root/.ssh
[root@ linux .ssh]# ls
id_rsa id_rsa.pub known_hosts
[root@ linux .ssh]# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQEAw0sFIAZwmnB0KcBrRm71kze+RueIa0qx6HEPThdqdBpq0VG8IIBgKdLvrVZbm
PXlqmMnNDvJt7o8V9DSfh2vwqEk8SoCuSQz53PwGJWSfmFYepkVF+0qpe3hsv2vFzFJmAoMINZzobkiwNH6Up9cQFPqMdpmP
J5cTe24dQLQuasFUQwg/IF15PK8o7dk0CUf+86Pxdb9XS3qGIZ7n6ABsIhE0MTQ0F4uX/pnZNRWCzb7f8HFZ12x9Lsg20V0SU
bbSELZmDfT0mLCP0ErUZ7oK8oTELcXl/sQ3Yddr5b09Caeb410hKoNCUSiUOSIUmMrp3aSyaLIoSktJ89IK35DQ== root@kt
linux.mb.com
```

- The **id\_rsa** is a private key and **id\_rsa.pub** is the public key which will be used later to make password less login.

#### Copying the public key on Client system

- To copy the server's public key in client system, the command is  
**#ssh-copy-id -i <public key location> <clients IP address>** (or user @ client IP)  
**#ssh-copy-id -i /root/.ssh/id\_rsa.pub 192.168.10.95**

```
[root@ linux ~]# ssh-copy-id -i /root/.ssh/id_rsa.pub 192.168.10.95
[root@192.168.10.95's password]
Now try logging into the machine, with "ssh '192.168.10.95'", and check in:
    .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Enter the password of the client to proceed, check it on client side whether it is copied or not

**Note: Public key can also be copied in a simple way by using**  
**#ssh-copy-id <remote machine address>; ex: ssh-copy-id 192.168.10.95**

### Move to client machine and check whether the key is copied properly or not

- To check the key navigate to `/root/.ssh/` directory and check for `authorized_keys` file which will hold all the system which are authorized and will not be asked for password.

```
#cd /root/.ssh/
#cat authorized_keys
```

```
[root@ cl5 ~]# cd /root/.ssh/
[root@ cl5 .ssh]# ls
authorized_keys
[root@ cl5 .ssh]# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAQEAw0sFIAZwmnB0KcBrRm71kze+RueIa0qx6HEPThdqdBpq0VG8IIBgKdLvrVZbm
PX1qmMnNDvJt7o8V9DSfHl2vwqEk8SoCuSQz53PwGJWSfmFYepkVF+0qpe3hsv2vFzFjmAoMInZZobkiwNH6Up9cQFPqMdpmP
J5cTe24dQLQuasFUQwg/IF15PK8o7dk0CUf+86Pxdb9XS3qGIZ7n6ABsIhE0MTQOF4uX/pnZNRWCZb7f8HFZ12x9Lsg20V0SU
bbSELZmDfT0mLCPOErUZ7oK8oTELcXl/sQ3Yddr5b09Caeb410hKoNCUSIUOSIUmrp3aSyaLIo5kTJ89IK35DQ== [root@
linux.mb.com
[root@ cl5 .ssh]#
```

### Try login to the client machine using SSH, check whether it is asking for password

- For logging into client machine the procedure is same as shown earlier.
- #ssh 192.168.10.95**

```
[root@mlinux1 ~]# ssh 192.168.10.95
Last login: Mon Sep 19 13:48:24 2016 from 192.168.10.95
[root@mlinux2 ~]# hostname
mlinux2.mb.com
[root@mlinux2 ~]#
```

### Remote file transfer with SCP and RSYNC

#### SCP (SECURE COPY)

- scp stands for secure cp (copy), which means that you can copy files across an ssh connection that will be encrypted, and therefore secured. As scp will be using ssh protocol to transfer the data, hence it is termed as the safest method of transferring data from one location to another.

#### LAB WORK:

##### To copy a file using SCP to remote machine from source location

- We are having a file `myfile` in `"/"` directory, in the server `linux.mb.com` who's IP is **192.168.10.98**, and we need to copy the same in **other** server's i.e. `cl5.mb.com` with an IP **192.168.10.95**, `/root` directory.
- The syntax for SCP a file from source location.

```
#scp <file name > <remote hosts IP >:<location to copy the file >
#scp /myfile 192.168.10.95:/root/
```

```
[root@ linux ~]# hostname
linux.my.com
[root@ linux ~]# cat /myfile
Welcome to my Technologies
[root@ linux ~]# scp /ktfile 192.168.10.95:/root/
myfile
```

100% 31

- Now log in to destination system and check whether if the file is there.

```
[root@linux ~]# ssh 192.168.10.95
Last login: Fri Oct 14 15:29:23 2011 from 192.168.10.98
[root@cl5 ~]# cd /root/
[root@tcl5 ~]# ls
anaconda-ks.cfg  Documents  install.log      myfile  Pictures
Desktop          Downloads  install.log.syslog  Music   Public
[root@cl5 ~]# cat myfile
Welcome to my Technologies
```

### To copy a file using SCP from a remote machine being in destination's location(reverse scp)

- Let's reverse the previous task, login to cl5 machine whose IP is 192.168.10.95, and transfer a file from linux machine whose IP is **192.168.10.98**
- Let's first remove the earlier copied file **myfile**, then copy it again from destination's location.
- The syntax for SCP a file from destination location.

**#scp <source system's IP>:<location of file to be copied> <destination location to copy>**

```
[root@cl5 ~]# rm myfile
rm: remove regular empty file `myfile'? y
[root@cl5 ~]# scp 192.168.10.98:/myfile /root/
myfile                                         100%   31
[root@cl5 ~]# cat myfile
Welcome to my Technologies
```

**Note:** Password will be asked for every transfer if public key is not saved on both locations, in our case we have already generated and copied the key, hence there is no password prompts.

### To copy a directory using SCP to remote machine from source's location

- We are having a directory **mydir** in "/" directory, in the server **linux.mb.com** who's IP is **192.168.10.98**, and we need to copy the same in **other** server's i.e. **cl5(mb.com)** with an IP **192.168.10.95**, **/root** directory. Then,
- The syntax SCP a directory from source's location, the syntax is  
**#scp <option> <dir name > <remote hosts IP >:<location to copy the directory >**  
**#scp -r /mydir 192.168.10.95:/root/**

```
[root@linux ~]# tree /mydir
/ktdir
└── file1
    ├── file2
    ├── file3
    ├── file4
    └── file5

0 directories, 5 files
[root@linux ~]# scp -r /mydir 192.168.10.95:/root/
file1                                         100%   0
file4                                         100%   0
file5                                         100%   0
file2                                         100%   0
file3                                         100%   0
```

### To copy a directory using SCP from a remote machine being in destination's location (reverse scp)

- Let's reverse the previous task, login to **cl5** machine who's IP is 192.168.10.95, and transfer a directory **mydir** from **linux** machine whose IP is **192.168.10.98**
- Let's first remove the earlier copied directory **mydir**, then copy it again being in destination's location.
- The syntax for SCP a file from destination location.  
**#scp <option> <source system's IP>:<location of file to be copied> <destination location to copy>**  
**#scp -r 192.168.10.98:/mydir /root/**

```
[root@ cl5 ~]# rm -rf /root/mydir/
[root@ cl5 ~]# scp -r 192.168.10.98:/mydir /root/
file1                                100%   0
file4                                100%   0
file5                                100%   0
file2                                100%   0
file3                                100%   0
[root@ cl5 ~]#
```

### RSYNC (REMOTE SYNCHRONIZATION)

- rsync is a very good program for backing up/mirroring a directory tree of files from one machine to another machine, and for keeping the two machines "in sync." It's designed to speed up file transfer by copying the differences between two files rather than copying an entire file every time.
- For example, Assume that we are supposed to take the backup of a system and copy the same to another system. For first time we will copy entire directory, but every day if we copy entire directory it will kill lots of time. In such situation if rsync is used it will only copy the updated files/directories rather than copying all files/directories inside main directory, which saves lots of time and speedup the transfer
- If rsync is combined with ssh it makes a great utility to sync the data securely. If rsync is not used with ssh, the risk sniffing will always be there.

#### LAB WORK:-

**Copy a directory using SCP, then update it and try rsync with SSH and check if the data is synced.**

- As we have already copy a directory earlier using SCP from **linux** to **cl5** system, let's use it for rsync.
- Update the directory with some files in **linux** machine

```
[root@ linux ~]# cd /mydir
[root@ linux mydir]# ls
file1  file2  file3  file4  file5
[root@ linux mydir]# touch file{6..9}
[root@ linux mydir]# ls
file1  file2  file3  file4  file5  file6  file7  file8  file9
[root@ linux mydir]#
```

- Check the content of same directory in **cl5**

```
[root@ cl5 ~]# cd /root/mydir
[root@ cl5 mydir]# ls
file1  file2  file3  file4  file5
[root@ cl5 mydir]#
```

- Use rsync to sync the directory on **cl5** machine, with the one in **linux** machine
- The syntax to rsync a directory is

```
#rsync <options> <encryption> <source dir> <destination IP>:<location of destination dir>
#rsync -rv -e ssh /mydir 192.168.10.95:/root/
```

```
[root@ linux ~]# rsync -rv -e ssh /mydir 192.168.10.95:/root/
sending incremental file list
mydir/  file6
mydir/  file7
mydir/  file8
mydir/  file9
```

Observe that it is only copying the files which are not there in destination's folder.

**Note:** If you don't want to use ssh just remove –e option from above syntax, but the drawback of it is there will be no encryption

```
[root@ linux ~]# rsync -rv /mydir 192.168.10.95:/root/
sending incremental file list
mydir/  file1
mydir/  file2
mydir/  file3
mydir/  file4
mydir/  file5
mydir/  file6
mydir/  file7
mydir/  file8
mydir/  file9
```

- To compress the data and send it in archive mode use **-avz** instead of **-rv** in rsync

#### Sync a file using rsync with ssh

- Let's check the file called file1 on both **linux** and **cl5** machines

[root@ <b>linux</b> mydir]# cat file1 Welcome to virtualpathTechnologies	[root@ <b>cl5</b> mydir]# cat file1 Welcome to virtualpathTechnologies
---	---

- Update the file **file1** in **linux**, sync it with rsync to **cl5** and check the file again.

- The syntax for syncing a file is

```
#rsync -avz -e ssh <source file> <destination ip>:<location of file >
```

```
[root@ linux mydir]# vim file1
[root@ linux mydir]# cat file1
Welcome to virtualpathTechnologies
AMEERPET HYDERABAD
[root@ linux mydir]# rsync -avz -e ssh /mydir/ file1 192.168.10.95:/root/mydir/
sending incremental file list
file1

sent 123 bytes received 37 bytes 106.67 bytes/sec
[root@ linux mydir]# cat file1
[root@ cl5 mydir]# cat file1
Welcome to virtualpathTechnologies
AMEERPET HYDERABAD
[root@ cl5 mydir]# cat file1
Welcome to virtualpathTechnologies
AMEERPET HYDERABAD
```

*Like this you can use rsync in many ways to transfer the updated file or files/directory to other system.*

#### Other important things in rsync

- **--dry-run** : dry-run is used to see the preview of transfer before doing actual transfer  
**Syntax:** rsync -avz -e ssh /mydir 192.168.104.82:/opt --dry-run
- **Reverse rsync** : reverse rsync is referred to sync folder from destination to source machine. It is exactly same like reverse scp seen in scp chapter  
**Syntax:** rsync -avz -e ssh 192.168.104.82/opt/mydir /mydir  
Where 192.168.104.82/opt/mydir is remote machine directory and /mydir is local machine directory

## SOFTWARE MANAGEMENT

To manage the software in Linux, two utilities are used,

1. RPM – REDHAT PACKAGE MANAGER
2. YUM – YELLOWDOG UPDATER MODIFIED OR DNF- DANDIFIED YUM

### RPM –REDHAT PACKAGE MANAGER

RPM is a package managing system (collection of tools to manage software packages). RPM is a powerful software management tool for installing, uninstalling, verifying, querying and updating software packages. RPM is a straight forward program to perform the above software management tasks.

#### **Features:**

- RPM can verify software packages.
- RPM can be served as a powerful search engine to search for software's.
- Components, software's etc can be upgraded using RPM without having to reinstall them
- Installing, reinstalling can be done with ease using RPM
- During updates RPM handles configuration files carefully, so that the customization is not lost.

### LAB WORK:-

#### **To check all the installed packages in the system**

- To check all the installed packages in the system, the syntax is
- **#rpm -qa** (where q stands for query, and a stands for all)

```
[root@ linux ~]# rpm -qa
Red_Hat_Enterprise_Linux-Release_Notes-6-en-US-1-21.el6.noarch
procps-3.2.8-14.el6.i686
net-snmp-libs-5.5-27.el6.i686
m17n-contrib-urdu-1.1.10-3.el6.noarch
bluez-libs-4.66-1.el6.i686
man-1.6f-29.el6.i686
xorg-x11-fonts-ISO8859-1-100dpi-7.2-9.1.el6.noarch
libXrender-0.9.5-1.el6.i686
nscd-2.12-1.7.el6.i686
dejavu-serif-fonts-2.30-2.el6.noarch
libXfixes-4.0.4-1.el6.i686
libchewing-0.3.2-27.el6.i686
dejavu-sans-mono-fonts-2.30-2.el6.noarch
libXdamage-1.1.2-1.el6.i686
```

**Note:** The output of above command will be very lengthier.

## To check whether a particular package is installed or not

- To check whether a package is installed or not out of the list of installed package, the syntax is

```
#rpm -qa <package name> or  
#rpm -q < package name>  
#rpm -qa vsftpd or #rpm -q vsftpd
```

```
[root@ linux ~]# rpm -qa vsftpd  
vsftpd-2.2.2-6.el6.i686  
[root@ linux ~]# rpm -q vsftpd  
vsftpd-2.2.2-6.el6.i686  
[root@ linux ~]# █
```

- One more method of checking the installed package, when you are not sure about the package name, like whether it starts with capital letter and full name etc.

```
#rpm -qa | grep -i < package name>  
#rpm -qa |grep -i vsft*
```

```
[root@ linux ~]# rpm -qa |grep -i vsf*  
cvs-1.11.23-11.el6.i686  
lklug-fonts-0.6-4.20090803cvs.el6.noarch  
libedit-2.11-4.20080712cvs.1.el6.i686  
vsftpd-2.2.2-6.el6.i686  
xdg-utils-1.0.2-15.20091016cvs.el6.noarch  
[root@ linux ~]# █
```

## To check whether a package is consistent or not, before installing it. (Testing the installation)

- To check the package's consistency,
- Move to the directory where you have kept the rpm package which you wish to install

```
[root@ linux ~]# cd /var/ftp/pub/rhel6/Packages/  
[root@ linux Packages]# ls |grep -i finger  
finger-0.17-39.el6.i686.rpm  
finger-server-0.17-39.el6.i686.rpm  
gdm-plugin-fingerprint-2.30.4-21.el6.i686.rpm
```

- The command used to check the package's consistency is

```
#rpm -ivh --test <package name>  
Where i = install, v= verbose view, and h = hash progress.  
#rpm -ivh -- test finger-0.17-39.el7.i686.rpm
```

```
[root@ linux Packages]# rpm -ivh --test finger-0.17-39.el6.i686.rpm  
warning: finger-0.17-39.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY  
Preparing... #### [100%]  
[root@ linux Packages]# █
```

If the installation status shows 100%, then the package is good or consistent.  
But while showing the hash progress if it shows any error, then the package is inconsistent.

## To install a package using rpm command and check whether it is installed properly or not.

- To install the package first we need to be in the directory of the package

```
[root@ linux ~]# cd /var/ftp/pub/rhel7/Packages/
[root@ linux Packages]# ls |grep -i finger
finger-0.17-39.el6.i686.rpm
finger-server-0.17-39.el6.i686.rpm
gdm-plugin-fingerprint-2.30.4-21.el6.i686.rpm
```

- To install the package the syntax is

```
#rpm -ivh <package name>
#rpm -ivh finger-0.17-39.el6.i686.rpm
```

```
[root@ linux Packages]# rpm -ivh finger-0.17-39.el6.i686.rpm
warning: finger-0.17-39.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Preparing... ################################################ [100%]
1:finger ################################################ [100%]
[root@ linux Packages]#
```

- To check whether it is installed or not

```
#rpm -qa finger
```

```
[root@ linux Packages]# rpm -qa finger
finger-0.17-39.el6.i686
[root@ linux Packages]#
```

- Check the installed package by using it command; finger is used to check user's details.

```
#finger <user name>
#finger myuser
```

```
[root@ linux Packages]# finger myuser
Login: myuser                                     Name: myuser
Directory: /home/myuser                           Shell: /bin/bash
Never logged in.
No mail.
No Plan.
[root@ linux Packages]#
```

## To remove a package or uninstall the package

- To remove a package the syntax is

```
#rpm -evh < package name>
#rpm -evh finger
```

Verify it by #rpm -q or rpm -qa command

```
[root@node1 modules]# rpm -q finger
finger-0.17-52.el7.x86_64
[root@node1 modules]# rpm -evh finger
Preparing... ################################################ [100%]
Cleaning up / removing...
1:finger-0.17-52.el7 ################################################ [100%]
[root@node1 modules]# rpm -q finger
package finger is not installed
```

## To see the information about the package before installing

- To see the info about a particular package which is not installed, move to the directory where you have kept the packages.

```
[root@ linux ~]# cd /var/ftp/pub/rhel6/Packages/
[root@ linux Packages]# ls |grep -i finger
finger-0.17-39.el6.i686.rpm
finger-server-0.17-39.el6.i686.rpm
gdm-plugin-fingerprint-2.30.4-21.el6.i686.rpm
```

- To see the info of a package, the syntax is

**#rpm -qip <package name>** (where **q** is for query, **i** is for install and **p** is for package)  
**#rpm -qip finger-0.17-39-el6.1686.rpm**

```
[root@ linux Packages]# rpm -qip finger-0.17-39.el6.i686.rpm
warning: finger-0.17-39.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Name        : finger                           Relocations: (not relocatable)
Version     : 0.17                            Vendor: Red Hat, Inc.
Release     : 39.el6                          Build Date: Fri 20 Nov 2009 09:03:29 AM IST
Install Date: (not installed)           Build Host: ls20-bc1-14.build.redhat.com
Group       : Applications/Internet      Source RPM: finger-0.17-39.el6.src.rpm
Size        : 25730                           License: BSD
Signature   : RSA/8, Mon 16 Aug 2010 09:36:07 PM IST, Key ID 199e2f91fd431d51
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary     : The finger client
Description :
Finger is a utility which allows users to see information about system
users (login name, home directory, name, how long they've been logged
in to the system, etc.). The finger package includes a standard
finger client.
```



## To see the information about the installed package

- To see the information or details about the installed package, the syntax is

**#rpm -qi < package name >**  
**#rpm -qi vsftpd**

```
[root@ linux ~]# rpm -qi vsftpd
Name        : vsftpd                           Relocations: (not relocatable)
Version     : 2.2.2                            Vendor: Red Hat, Inc.
Release     : 6.el6                           Build Date: Wed 26 May 2010 06:16:46 PM IST
Install Date: Wed 12 Oct 2011 05:22:23 PM IST   Build Host: x86-009.build.bos.redhat.com
Group       : System Environment/Daemons      Source RPM: vsftpd-2.2.2-6.el6.src.rpm
Size        : 351576                           License: GPLv2 with exceptions
Signature   : RSA/8, Tue 17 Aug 2010 01:49:04 AM IST, Key ID 199e2f91fd431d51
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL         : http://vsftpd.beasts.org/
Summary     : Very Secure Ftp Daemon
Description :
vsftpd is a Very Secure FTP daemon. It was written completely from
scratch.
[root@ linux ~]#
```

## To check the package related to a command

- To check the package of a particular command, first check the installed location of a command

```
#which <command name>
#which cat
```

```
[root@node1 ~]# which cat
/usr/bin/cat
```

- Now, use the following command,

```
#rpm -qf <path of the command>
#rpm -qf /bin/cat
```

```
[root@node1 ~]# rpm -qf /usr/bin/cat
coreutils-8.22-23.el7.x86_64
```

## To install a package forcefully

- Before installing a package forcefully, first understand a situation where we need this force option.
- Let us corrupt one command and show you how to install its package forcefully.
- First check the package of the command we are going to corrupt. Let say **mount** command

```
#which mount, #which date
[root@node1 ~]# which mount
/usr/bin/mount
[root@node1 ~]# which date
/usr/bin/date
```

- Okay, so we know the package of mount let's copy other commands content over mount command. Let copy **date** command's contents over **mount** command.

```
#cp /usr/bin/date /usr/bin/mount
```

```
[root@node1 ~]# cp /usr/bin/date /usr/bin/mount
cp: overwrite â/usr/bin/mountâ? y
```

- Now when you run mount command it will show date, that means it is corrupted.

```
[root@ linux ~]# mount
Sat Oct 15 03:25:34 IST 2011
[root@ linux ~]# date
Sat Oct 15 03:25:41 IST 2011
[root@ linux ~]#
```

- So, to fix the mount command we need to reinstall its package, let's install the package and check whether mount command is fixed or not. Move to the folder where you kept the packages and install it

```
#rpm -ivh util-linux-ng 2.17.2-6.el6.i686
```

```
[root@node1 ~]# rpm -qf /usr/bin/mount
util-linux-2.23.2-59.el7.x86_64
```

```
[root@node1 rhe17]# cd /var/ftp/pub/rhe17/Packages/
[root@node1 Packages]# ls util-linux*
util-linux-2.23.2-59.el7.i686.rpm  util-linux-2.23.2-59.el7.x86_64.rpm
[root@node1 Packages]# rpm -ivh util-linux-2.23.2-59.el7.x86_64.rpm
warning: util-linux-2.23.2-59.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature
        ID fd431d51: NOKEY
Preparing...                                           #####
package util-linux-2.23.2-59.el7.x86_64 is already installed
```

It says the package is already installed, check by using mount command whether it is working fine.

```
[root@ linux Packages]# mount
Sat Oct 15 03:30:54 IST 2011
[root@ linux Packages]# █
```

- Oops...!!! It isn't fixed yet, now to fix it, force installation is to be done, the syntax is

```
#rpm -ivh <package name> --force
```

```
# rpm -ivh util-linux-ng 2.17.2-6.el7_x86_64 -force
```

```
[root@node1 Packages]# rpm -ivh util-linux-2.23.2-59.el7.x86_64.rpm --force
warning: util-linux-2.23.2-59.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature,
        key ID fd431d51: NOKEY
Preparing...                                           ###### [100%]
Updating / installing...
  1:util-linux-2.23.2-59.el7          ##### [100%]
[root@node1 Packages]# mount |tail
/dev/mapper/rhel-root on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=32,pgrp=1,timeout=5,maxproto=5,direct,pipe_ino=17792)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,seclabel)
mqueue on /dev/mqueue type mqueue (rw,relatime,seclabel)
/dev/sdal on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=188264k)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
```

Okay then, we've not only installed the package successfully but we have also fixed the command. Congratulations.

## To see the configuration files of the installed package

- To see the configuration files of the installed package, the syntax is

```
#rpm -qlc <package name>
```

```
[root@ linux Packages]# rpm -qlc vsftpd
/etc/logrotate.d/vsftpd
/etc/pam.d/vsftpd
/etc/vsftpd/ftpusers
/etc/vsftpd/user_list
/etc/vsftpd/vsftpd.conf
```

## To see the directory with which a particular package is associated.

- To see the directory with which a package is associated, the syntax is

```
#rpm -qld <package name>
```

```
#rpm -qld vsftpd
```

```
[root@ linux ~]# rpm -qld vsftpd
/usr/share/doc/vsftpd-2.2.2/AUDIT
/usr/share/doc/vsftpd-2.2.2/BENCHMARKS
/usr/share/doc/vsftpd-2.2.2/BUGS
/usr/share/doc/vsftpd-2.2.2/COPYING
/usr/share/doc/vsftpd-2.2.2/Changelog
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/README
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/README.configuration
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/vsftpd.conf
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/vsftpd.xinetd
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET SITE NOINETD/README
```

## To install a package without installing dependencies

- Some rpm requires some other packages to be installed before it can be installed; this requirement is termed as '**dependency**'. This means that before installing a package we need to install the required packages first, so that it can work properly.
- But sometimes we can skip installing the dependency, if we don't have that dependent software with us.
- The syntax for it is  

```
#rpm -ivh <package name> --nodeps
#rpm -ivh util-linux-ng 2.17.2-6.el6.i686 --nodeps
```

## To update a particular package

- To update a package the syntax is

```
#rpm -Uvh <package name>
```

```
#rpm -Uvh vsftpd -2.2.4
```

## To check the changes are made after installation of package

- First let's make some changes in the configuration file of a package say **vsftpd**

```
#vim /etc/vsftpd/vsftpd.conf
```

```
# Example config file /etc/vsftpd/vsftpd.conf
#
##### The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
```

- Now run the following command and check for the result.

```
[root@ linux ~]# vim /etc/vsftpd/vsftpd.conf
```

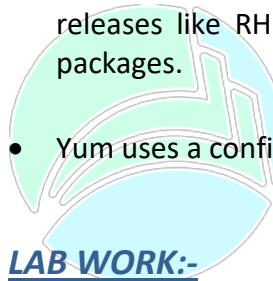
```
[root@ linux ~]# rpm -V vsftpd
```

```
S.5....T. c /etc/vsftpd/vsftpd.conf
```

It is showing that some changes have been made. Isn't it cool!!!

## YUM – YELLOWDOG UPDATER MODIFIED

- The Yellow dog Updater Modified (YUM) is a package management application for computers running Linux operating systems.
- Yum is a standard method of managing the installation and removal of software. Several graphical applications exist to allow users to easily add and remove packages; however, many are simply friendly interfaces with yum running underneath. These programs present the user with a list of available software and pass the user's selection on for processing. It is yum that actually downloads the packages and installs them in the background.
- Packages are downloaded from collections called **repositories**, which may be online, on a network, and/or on installation media. If one package due to be installed relies on another being present, this **dependency** can usually be resolved without the user needing to know the details. For example, a game being installed may depend on specific software to play its music. The problem of solving such dependencies can be handled by yum because it knows about all the other packages that are available in the repository.
- Yum will work only from Centos 5 / Red hat 5 and latest versions of fedora. For Old releases like RHEL 4 you need to use up2date command to update your rpm based packages.
- Yum uses a configuration file at **/etc/yum.conf**



**VIRTUALPATH**  
**TECHNO SOLUTIONS**

### **Configuring a YUM server in RHEL6/7**

**To configure a YUM server the steps are.**

- Make sure that vsftpd package is installed, if not install it.
- Copy entire RHEL6/7 DVD to “/var/ftp/pub/rhel” directory, where rhel dir is to made by us only it is not default dir.
- Make a repo file as “my.repo”in /etc/yum.repos.d directory
- Clean the yum cache and check the package list using yum command

**Let's start with the first step**

- Checking the vsftpd package is installed or not.

```
[root@node1 Packages]# rpm -q vsftpd
vsftpd-3.0.2-25.el7.x86_64
```

- If it is not installed, then go to dvd's mount point and navigate to "Packages" directory and install it as shown below.

```
[root@ktlinux ~]# mount
gvfsd-fuse on /run/user/0/gvfs type fuse.gvfsd-fuse
/dev/sr0 on /run/media/root/RHEL-7.1 Server.x86_64 t
 0400,amode=0500,umask=udisks2)
[root@mlinux71 ~]#
```

- As we know the mount point of dvd is **/media/RHEL\_6** (in rhel6) & **/run/media/root/RHEL7** (in rhel7), move to its location and enter into **Packages** directory.

```
[root@mlinux71 ~]# cd /run/media/ktuser/RHEL-7.1\ Server.x86_64/
[root@mlinux71 RHEL-7.1 Server.x86_64]# cd Packages/
[root@mlinux71 Packages]# ls vsftpd*
vsftpd-3.0.2-9.el7.x86_64.rpm
[root@mlinux71 Packages]# rpm -ivh vsftpd-3.0.2-9.el7.x86_64.rpm
warning: vsftpd-3.0.2-9.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Preparing...                                           #####
Updating / installing...
 1:vsftpd-3.0.2-9.el7                                #####
[root@mlinux71 Packages]#
```

As it is already installed, it is not being installed.

**Copy entire RHEL7 DVD to "/var/ftp/pub/rhel7" directory, Where rhel7 dir is to be created manually, it is not a default directory**

- First make an directory "rhe7" under **/var/ftp/pub**  
**#mkdir /var/ftp/pub/rhel7**

```
[root@mlinux71 Packages]# cd /var/ftp/pub/
[root@mlinux71 pub]# ls
[root@mlinux71 pub]# mkdir rhel7
[root@mlinux71 pub]#
```

- Now copy the RHEL6/7 DVD to **/var/ftp/pub/rhel** directory with its default permission  
**#cp -rvfp /run/media/root/RHEL-7.1\ Server.x86\_64/\* /var/ftp/pub/rhel7**

```
[root@mlinux71 ~]# cp -rvfp /run/media/root/RHEL-7.1\ Server.x86_64/* /var/ftp/pub/rhel7
'/run/media/root/RHEL-7.1 Server.x86_64/addons' -> '/var/ftp/pub/rhel/addons'
'/run/media/root/RHEL-7.1 Server.x86_64/addons/HighAvailability' -> '/var/ftp/pub/rhel/addons/HighAvailability'
'/run/media/root/RHEL-7.1 Server.x86_64/addons/HighAvailability/TRANS.TBL' -> '/var/ftp/pub/rhel/HighAvailability/TRANS.TBL'
'/run/media/root/RHEL-7.1 Server.x86_64/addons/HighAvailability/corosync-2.3.4-4.el7.x86_64.rpm' -> '/var/ftp/pub/rhel/HighAvailability/corosync-2.3.4-4.el7.x86_64.rpm'
'/run/media/root/RHEL-7.1 Server.x86_64/addons/HighAvailability/corosynclib-2.3.4-4.el7.i686.rpm'
```

Note:- it will take around 3-5 minutes copy all the data, based on the DVD

- Check the directory after copying is finished.

```
[root@node1 ~]# cd /var/ftp/pub/rhel7/
[root@node1 rhel7]# ls
addons  extra_files.json  isolinux  Packages          RPM-GPG-KEY-redhat-release
EFI      GPL              LiveOS   repodata        TRANS.TBL
EULA    images           media.repo  RPM-GPG-KEY-redhat-beta
```

Okay, then half of our configuration is completed.

### Make a repo file as "my.repo" in /etc/yum.repos.d directory

- The file which we make inside /etc/yum.reops.d, will be functioning as the repository address and configuration file. Create the file with following details.

```
#vim /etc/yum.repos.d/my.repo
```

```
[MYREPO]
name=RHEL7.1
baseurl=file:///var/ftp/pub/rhel
enabled=1
gpgcheck=0
```

I guess there's some explanation requires about the fields we have entered.

- [MYREPO]** is the repo ID, which will be displayed while using yum repository.
- name** is the name given for the repository.
- baseurl** is the location of the dvd dump we have made.
- enabled** is to enable or disable the repository. The possible value for it is **0** and **1**, where **0** means disable and **1** means enabled.
- gpgcheck** gpgcheck= GNU privacy guard. gpgcheck used for signature verification from its central database. If signature verification is successful then you are sure about the security. If you set the value of gpgcheck is 1 then it asks for signature verification else it doesn't.
- Example of gpgkey in RHEL- *RPM-GPG-KEY-redhat-release*

### Clean the yum cache and check the package list using yum command

- To clear the cache use the following command

```
#yum clean all
```

```
[root@mlinux71 ~]# yum clean all
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management.
subscription-manager to register.
Cleaning repos: MYREPO
Cleaning up everything
[root@mlinux71 ~]#
```

If the configuration is correct, then the following output will be displayed, otherwise there will be some errors displayed.

- Now let's check whether our repository is functioning properly or not.

```
#yum repolist (to list all the repositories in the system)
```

```
[root@mlinux71 ~]# yum repolist
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register
.
repo id                                repo name                status
MYREPO                                    RHEL7.1                  4,371
repolist: 4,371
[root@mlinux71 ~]#
```

### Start the vsftpd service and make it permanent

```
[root@mlinux71 ~]# systemctl start vsftpd; systemctl enable vsftpd
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
[root@mlinux71 ~]#
```

### Allow ftp in firewalld in rhel7 if it is running

```
[root@mlinux71 ~]# firewall-cmd --add-service=ftp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
```

### Configure the yum client and check whether yum server is responding to it.

Configuring a yum client is very simple with just three steps.

- Make a repo file /etc/yum.repo.d/ as “mycl.repo”
- Clean the cache and check whether yum server is responding or not

#### Make a repo file /etc/yum.repo.d/ as “mycl.repo”

- Just make a repo file like we did for server but with only one change in baseurl as shown below

```
#vim /etc/yum.repos.d/mycl.repo
```

```
[MYCL]
name=rhel7cl
baseurl=ftp://192.168.106.81/pub/rhel7
enabled=1
gpgcheck=0
```

Note: - baseurl =ftp://192.168.10.81/pub/rhel refers to the server's ftp address.

#### Clean the cache and check whether yum server is responding or not

- Just clean the cache as we have done earlier in server's configuration.

```
[root@ktcl5 ~]# yum clean all
Loaded plugins: refresh-packagekit, rhnplugin
Cleaning up Everything
[root@ktcl5 ~]#
```

- Check whether the server is responding to client's yum request.

```
#yum repolist
```

```
[root@mlinux72 ~]# yum repolist
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

MYCL
(1/2): MYCL/group_gz                                | 4.1 kB   00:00
(2/2): MYCL/primary_db                               | 134 kB   00:01
                                                | 3.4 MB   00:01
repo id                                              repo name          status
MYCL                                                 rhel7cl           4,371

repolist: 4,371

[root@mlinux72 ~]# yum list |more
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Installed Packages
GConf2.x86_64                                         3.2.6-8.el7          @anaconda/7.1
ModemManager.x86_64                                    1.1.0-6.git20130913.el7    @anaconda/7.1
ModemManager-glib.x86_64                               1.1.0-6.git20130913.el7    @anaconda/7.1
NetworkManager.x86_64                                 1:1.0.0-14.git20150121.b4ea599c.el7
                                                       @anaconda/7.1
NetworkManager-adsl.x86_64                            1:1.0.0-14.git20150121.b4ea599c.el7
                                                       @anaconda/7.1
```

If your output is like this then you have successfully configured a yum client as well.

Congrats!!! Now you can configure as many as clients you want.

## INTRODUCTION TO DNF – DANDIFIED YUM FROM RHEL8:

DNF stands for Dandified YUM is a software package manager for RPM-based Linux distributions. It is used to install, update and remove packages in the Fedora/RHEL/CentOS operating system. It is the default package manager of Fedora 22, CentOS8 and RHEL8. DNF is the next generation version of YUM and intended to be the replacement for YUM in RPM-based systems. DNF is powerful and has robust features than you'll find in yum. DNF makes it easy to maintain groups of packages and capable of automatically resolving dependency issues.

### Configuring a DNF server in RHEL8

To configure a DNF YUM server the steps are.

- Make sure that vsftpd package is installed, if not install it.
- Copy entire RHEL8 DVD to “/var/ftp/pub/rhel8” directory, where rhel8 dir hast to be created manually by us only, as it is not default dir.
- Make a repo file as “my.repo”in /etc/yum.repos.d directory
- Clean the yum cache and check the package list using yum command

Let's start with the first step

- Checking the vsftpd package is installed or not.

```
[root@localhost ~]# rpm -q vsftpd
package vsftpd is not installed
```

- If it is not installed, mount RHEL8 DVD on some mount point and navigate to “Packages” directory and install it as shown below.

```
[root@localhost ~]# mount /dev/sr0 /media
mount: /media: WARNING: device write-protected, mounted
[root@localhost ~]# mount |grep sr0
/dev/sr0 on /media type iso9660 (ro,relatime,nojoliet,c:
```

- As we know the mount point of DVD is /media, move to its location and enter into **AppStream/ Packages** directory and install vsftpd package using rpm

```
[root@localhost ~]# cd /media
[root@localhost media]# ls
AppStream  EFI  extra_files.json  images  media.repo          RPM-GPG-KEY-redhat-release
BaseOS     EULA  GPL              isolinux  RPM-GPG-KEY-redhat-beta  TRANS.TBL
[root@localhost media]# cd AppStream/
[root@localhost AppStream]# ls
Packages  repodata
[root@localhost AppStream]# cd Packages/
[root@localhost Packages]# ls vsftpd*
vsftpd-3.0.3-28.el8.x86_64.rpm
[root@localhost Packages]#
[root@localhost Packages]# rpm -ivh vsftpd-3.0.3-28.el8.x86_64.rpm
warning: vsftpd-3.0.3-28.el8.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Verifying...                                              #####
Preparing...                                               #####
Updating / installing...
 1:vsftpd-3.0.3-28.el8                                [100%]
```

Copy entire RHEL8 DVD to “/var/ftp/pub/rhel8” directory, Where rhel8 dir is to be created manually, as it is not a default directory.

**Note: make sure around 8 GB space is free in /var to copy the data**

- First make an directory “rhel8” under /var/ftp/pub

```
#mkdir /var/ftp/pub/rhel8
```

```
[root@localhost ~]# cd /var/ftp/pub/
[root@localhost pub]# mkdir rhel8
[root@localhost pub]# ls
rhel8
```

- Now copy the RHEL8 DVD to /var/ftp/pub/rhel directory with its default permission

```
#cp -rvfp /media/* /var/ftp/pub/rhel8
```

```
[root@localhost pub]# cp -rvfp /media/* /var/ftp/pub/rhel8/
'/media/AppStream' -> '/var/ftp/pub/rhel8/AppStream'
'/media/AppStream/repo' -> '/var/ftp/pub/rhel8/AppStream/repo'
'/media/AppStream/repo/0153152556c65c34e8e52a6aeacae37951b697ac654
ar/ftp/pub/rhel8/AppStream/repo/0153152556c65c34e8e52a6aeacae37951
z'
```

**Note:-** it will take around 3-5 minutes copy all the data, based on the DVD

- Check the directory after copying is finished.

```
[root@localhost ~]# cd /var/ftp/pub/rhel8/
[root@localhost rhel8]# ls
AppStream  EFI  extra_files.json  images  media.repo      RPM-GPG-KEY-redhat-release
BaseOS    EULA  GPL              isolinux  RPM-GPG-KEY-redhat-beta  TRANS.TBL
```

Okay, then half of our configuration is completed.

**Make a repo file as “my.repo”in /etc/yum.repos.d directory**

- The file which we make inside /etc/yum.repos.d, will be functioning as the repository address and configuration file. Create the file with following details.

```
#vim /etc/yum.repos.d/my.repo
```

```
[BaseOS]
name=rhel8_BaseOS
baseurl=file:///var/ftp/pub/rhel8/BaseOS
enabled=1
gpgcheck=1
gpgkey=file:///var/ftp/pub/rhel8/RPM-GPG-KEY-redhat-release

[App_Steam]
name=rhel8_App_Steam
baseurl=file:///var/ftp/pub/rhel8/AppStream
enabled=1
gpgcheck=1
gpgkey=file:///var/ftp/pub/rhel8/RPM-GPG-KEY-redhat-release
```

I guess there's some explanation requires about the fields we have entered.

- [MYREPO]** is the repo ID, which will be displayed while using yum repository.
- name** is the name given for the repository.
- baseurl** is the location of the dvd dump we have made.

- **enabled** is to enable or disable the repository. The possible value for it is **0** and **1**, where **0** means disable and **1** means enabled.
- **gpgcheck** gpgcheck= GNU privacy guard. gpgcheck used for signature verification from its central database. If signature verification is successful then you are sure about the security. If you set the value of gpgcheck is 1 then it asks for signature verification else it doesn't.
- Example of gpgkey in RHEL- *RPM-GPG-KEY-redhat-release*

### Clean the yum cache and check the package list using yum command

- To clear the cache use the following command

```
#dnf clean all
```

```
[root@mlinux3 ~]# dnf clean all
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management.
12 files removed
```

If the configuration is correct, then the following output will be displayed, otherwise there will be some errors displayed.

- Now let's check whether our repository is functioning properly or not.

```
#dnf repolist (to list all the repositories in the system)
```

```
[root@mlinux3 ~]# dnf repolist
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You
rhel8_BaseOS
rhel8_BaseOS
Last metadata expiration check: 0:00:01 ago on Tuesday 31 March 2020
repo id
repo name
rhel8_App_Stream
rhel8_BaseOS
rhel8_BaseOS
```

In order to share files with ftp, make the following changes in ftp config

```
#vim /etc/vsftpd/vsftpd.conf
```

```
#
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).
anonymous_enable=YES
#
```

Start the vsftpd service and make it permanent

```
[root@mlinux71 ~]# systemctl start vsftpd; systemctl enable vsftpd
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
[root@mlinux71 ~]#
```

Allow ftp in firewalld, if it is running

```
[root@mlinux71 ~]# firewall-cmd --add-service=ftp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
```

## Configure the yum client and check whether yum server is responding to it.

Configuring a yum client is very simple with just three steps.

- Make a repo file /etc/yum.repos.d/ as “mycl.repo”
- Clean the cache and check whether yum server is responding or not

**Make a repo file /etc/yum.repos.d/ as “mycl.repo”**

- Just make a repo file like we did for server but with only one change in baseurl as shown below

```
#vim /etc/yum.repos.d/mycl.repo
```

```
[BaseOS]
name=rhel8_BaseOS
baseurl=ftp://192.168.10.30/pub/rhel8/BaseOS
enabled=1
gpgcheck=1
gpgkey=ftp://192.168.10.30/pub/rhel8/RPM-GPG-KEY-redhat-release

[App_Stream]
name=rhel8_App_Stream
baseurl=ftp://192.168.10.30/pub/rhel8/AppStream
enabled=1
gpgcheck=1
gpgkey=ftp://192.168.10.30/pub/rhel8/RPM-GPG-KEY-redhat-release
```

Note: - baseurl =ftp://192.168.10.30/pub/rhel8 refers to the server's ftp address.

**Clean the cache and check whether yum server is responding or not**

- Just clean the cache as we have done earlier in server's configuration.

```
[root@mlinux4 ~]# dnf clean all
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management.
0 files removed
```

- Check whether the server is responding to client's yum request.

```
#yum repolist
```

```
[root@mlinux4 ~]# dnf repolist
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You
rhel8_BaseOS
rhel8_App_Stream
Last metadata expiration check: 0:00:02 ago on Tuesday 31 March 2020
repo id                                repo name
App_Stream                               rhel8_App_Stream
BaseOS                                  rhel8_BaseOS
```

```
[root@mlinux4 ~]# dnf list |more
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use
Last metadata expiration check: 0:03:03 ago on Tuesday 31 March 2020 09:55:37
Installed Packages
GConf2.x86_64                                3.2.6-22.el8
ModemManager.x86_64                            1.10.4-1.el8
ModemManager-glib.x86_64                        1.10.4-1.el8
NetworkManager.x86_64                           1:1.20.0-3.el8
NetworkManager-adsl.x86_64                      1:1.20.0-3.el8
NetworkManager-bluetooth.x86_64                 1:1.20.0-3.el8
```

If your output is like this then you have successfully configured a yum client as well.  
 Congrats!!! Now you can configure as many as clients you want

## Working with YUM commands

### To list the available packages in the repository

- #yum list ( or ) #yum list all ( or ) #yum list |more (to view line wise)  
**As we have seen the first command, second will also give exactly the same output. Let us see the third command**  
**#yum list |more or #dnf list |more**

```
[root@ linux ~]# yum list |more
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Installed Packages
ConsoleKit.i686                               0.4.1-3.el6          @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                0.4.1-3.el6          @anaconda-RedHatEnterpr
ConsoleKit-libs.i686                            1:0.8.1-5.el6        @anaconda-RedHatEnterpr
NetworkManager.i686                            1:0.8.1-5.el6        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                1:0.8.1-5.el6        @anaconda-RedHatEnterpr
NetworkManager-glib.i686                         1:0.8.1-5.el6        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                1:0.8.1-5.el6        @anaconda-RedHatEnterpr
NetworkManager-gnome.i686                        1:0.8.1-5.el6        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                2.14.17-3.1.el6     @anaconda-RedHatEnterpr
DRBit2.i686                                    0.5.8-13.el6         @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                -More--
```



### To list all the installed packages in the system.

- To view all the installed packages in the system, the syntax is

**#yum list installed or #dnf list installed**

```
[root@ linux ~]# yum list installed
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Installed Packages
ConsoleKit.i686                               0.4.1-3.el6          @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0            0.4.1-3.el6          @anaconda-RedHatEnt
ConsoleKit-libs.i686                            0.4.1-3.el6          @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0            0.4.1-3.el6          @anaconda-RedHatEnt
ConsoleKit-x11.i686                            0.4.1-3.el6          @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0            014-1.el6           @anaconda-RedHatEnt
```

### To check a particular package is installed or not

- To check whether a package is installed or not the syntax is

**#yum list installed <package name> or #dnf list installed <package name>**

**#yum list installed vsftpd or #dnf installed vsftpd**

```
[root@ linux ~]# yum list installed vsftpd
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Installed Packages
vsftpd.i686           2.2.2-6.el6          @anaconda-RedHatEnterpriseLinux-201009221732.i386/6.0
[root@ linux ~]#
```

### To install a package using yum

- Installing a package using yum does not require full package name as in the case of rpm, and it also automatically resolves the dependencies as well.
- The syntax for installing a package is

**#yum install <package name> or #dnf install <package name>**

**#yum install dovecot\* or #dnf install dovecot\* (where \* means anything with name "dovecot")**

```
[root@mlinux4 ~]# dnf install dovecot*
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
Last metadata expiration check: 0:16:23 ago on Tuesday 31 March 2020 09:55:37 PM IST.
Dependencies resolved.
=====
 Package           Architecture Version      Repository
 =====
Installing:
dovecot            x86_64        1:2.2.36-10.el8   App_Stream
dovecot-mysql      x86_64        1:2.2.36-10.el8   App_Stream
dovecot-pgsql      x86_64        1:2.2.36-10.el8   App_Stream
dovecot-pigeonhole x86_64        1:2.2.36-10.el8   App_Stream
Installing dependencies:
clucene-core       x86_64        2.3.3.4-31.20130812.e8e3d20git.el8   App_Stream
libpq              x86_64        10.5-1.el8      App_Stream
mariadb-connector-c x86_64        3.0.7-1.el8      App_Stream
mariadb-connector-c-config noarch        3.0.7-1.el8      App_Stream
=====
Transaction Summary
=====
Install 8 Packages

Total download size: 6.1 M
Installed size: 24 M
Is this ok [y/N]:
```

It will prompt you for y/n (Yes/No) to continue, type y and continue installing the package

- installing a package without being prompt for y or n, the syntax is  
**#yum install <package name> -y or # dnf install <package name> -y**  
**#yum install dovecot -y or #dnf install dovecot -y**

```
[root@mlinux4 ~]# dnf install dovecot -y
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
Last metadata expiration check: 0:07:49 ago on Tuesday 31 March 2020 10:13:05 PM IST.
Dependencies resolved.
=====
Package           Architecture     Version          Repository
=====
Installing:
dovecot           x86_64          1:2.2.36-10.el8      App_Stream
Installing dependencies:
clucene-core      x86_64          2.3.3.4-31.20130812.e8e3d20git.el8      App_Stream
Transaction Summary
=====
Install 2 Packages

Total download size: 5.2 M
Installed size: 21 M
Downloading Packages:
(1/2): clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64.rpm
(2/2): dovecot-2.2.36-10.el8.x86_64.rpm
=====
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing   :
  Installing  : clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64
  Running scriptlet: dovecot-1:2.2.36-10.el8.x86_64
  Installing   : dovecot-1:2.2.36-10.el8.x86_64
  Running scriptlet: dovecot-1:2.2.36-10.el8.x86_64
  Verifying    : clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64
  Verifying    : dovecot-1:2.2.36-10.el8.x86_64
Installed products updated.

Installed:
  dovecot-1:2.2.36-10.el8.x86_64                               clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64

Complete!
[root@mlinux4 ~]#
```

### To remove the package with yum command

- To remove the package using yum command, the syntax is  
**#yum remove <package name> or #dnf remove <package name>**  
**#yum remove finger -y or #dnf remove <package name>**

```
[root@mlinux4 ~]# dnf remove dovecot -y
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
Modular dependency problems:

  Problem 1: conflicting requests
    - nothing provides module/perl:5.26 needed by module perl-DBD-SQLite:1.58:8010020190322125518:073fa5fe-0.x86_64
  Problem 2: conflicting requests
    - nothing provides module/perl:5.26 needed by module perl-DBI:1.641:8010020190322130042:16b3ab4d-0.x86_64
Dependencies resolved.
=====
Package           Architecture     Version
=====
Removing:
dovecot           x86_64          1:2.2.36-10.el8
Removing unused dependencies:
clucene-core      x86_64          2.3.3.4-31.20130812.e8e3d20git.el8
Transaction Summary
=====
Remove 2 Packages
```

### To update the package using yum

- To update the package using yum command, the syntax is  
**#yum update <package name> or #dnf update <package name>**  
**#yum update httpd or #dnf update httpd**

```
[root@ linux ~]# yum update httpd
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Update Process
No Packages marked for Update
[root@ linux ~]#
```

As there are no updates available for it, it is not showing anything to update

### To install a package locally from a folder, pen drive or cd rom

- Move to the package where you have stored the package to be installed

```
[root@ cl5 ~]# cd /
[root@ cl5 /]# ls
bin      etc          lib      mnt      root      sys
boot    finger-0.17-39.el6.i686.rpm lost+found  net      sbin      tmp
cgroup   ftp-0.17-51.1.el6.i686.rpm  media     opt      selinux   usr
dev      home         misc      proc     srv      var
```

- The syntax for installing a package locally is  
**#yum localinstall <package name> -y # dnf localinstall <package name>**  
**#yum localinstall finger\* -y (or) #yum localinstall finger-0.17-39.el6.i686.rpm -y**  
**#dnf localinstall finger\* -y #dnf localinstall finger-0.17-39.el6.i686.rpm -y**

```
[root@ cl5 /]# yum localinstall finger-0.17-39.el6.i686.rpm -y
Setting up Local Package Process
Examining finger-0.17-39.el6.i686.rpm: finger-0.17-39.el6.i686
Marking finger-0.17-39.el6.i686.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package finger.i686 0:0.17-39.el6 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package      Arch      Version       Repository      Size
=====
Installing:
finger      i686      0.17-39.el6      /finger-0.17-39.el6.i686      25 k

Transaction Summary
=====
Install      1 Package(s)
Upgrade      0 Package(s)

Total size: 25 k
Installed size: 25 k
Downloading Packages:
```

### To see the information about the package

#yum info <package name> or #dnf info <package name>

#yum info vsftpd or dnf info vsftpd

```
[root@mlinux3 ~]# dnf info vsftpd
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use subscrhel8_BaseOS
rhel8_App_Stream
Installed Packages
Name        : vsftpd
Version     : 3.0.3
Release    : 28.el8
Architecture: x86_64
Size       : 356 k
Source     : vsftpd-3.0.3-28.el8.src.rpm
Repository  : @System
Summary    : Very Secure Ftp Daemon
URL        : https://security.appspot.com/vsftpd.html
License    : GPLv2 with exceptions
Description : vsftpd is a Very Secure FTP daemon. It was written completely from
              : scratch.
```

### To list and install a group of packages using yum

- To list the group of package the syntax is

#yum grouplist or dnf grouplist

```
[root@mlinux4 ~]# dnf grouplist
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subsc
Last metadata expiration check: 0:23:25 ago on
Available Environment Groups:
  Server
  Minimal Install
  Workstation
  Custom Operating System
  Virtualization Host
Installed Environment Groups:
  Server with GUI
Installed Groups:
  Container Management
  Headless Management
Available Groups:
  Legacy UNIX Compatibility
  Development Tools
  .NET Core Development
  Graphical Administration Tools
  Network Servers
  RPM Development Tools
  Scientific Support
  Security Tools
  Smart Card Support
  System Tools
```

- To install package group called “Development Tools”, the syntax is

#yum groupinstall <group name> -y or dnf groupinstall ‘group name’

#yum groupinstall ‘Development Tools’ -y #dnf grouplist ‘Development Tools’ -y

```
[root@node1 ~]# yum groupinstall 'Development Tools' -y
```

#### Removing a Group package using yum

- To remove a group, the syntax is

#yum groupremove <group name> or dnf groupremove <group name>

#yum groupremove ‘Development Tools’ -y or #dnf groupremove ‘Development Tools’

```
[root@node1 ~]# yum groupremove 'Development Tools' -y
```

```
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
```

#### Other commands with yum :

##### To reinstall a package

#yum reinstall <package name> -y or #dnf reinstall <package name>

##### To see group information

#yum groupinfo <group name> or #dnf groupinfo ‘group name’



The software management can be learnt more by using manual pages like man yum or dnf and also man rpm etc.

## MANAGING PROCESS

- A Linux process is a program running in the Linux system. Depending on Linux distributions, it's also known as **service**. In Linux community however, a Linux process is called **daemon**.
- When you start a program or running an application in Linux, you actually execute that program. A Linux process (a **daemon**), running in foreground or in the background, uses memory and CPU resources. That's why we need to manage Linux process. Keeping unused Linux process running in the system is a waste and also exposes your system to security threat.
- In Linux, every running process or daemon is given an identity number called **PID (Process ID)**. The process id is unique. We can terminate unused program in the system by stopping its process id.
- In order to manage Linux processes, we need to identify some process information such as who's responsible for the process, which terminal the process is running from and what command used to run the process.

**There are generally three types of processes that run on Linux.**

- **Interactive Processes**
- **System Process or Daemon**
- **Automatic or batch**

### Interactive Processes

Interactive processes are those processes that are invoked by a user and can interact with the user. VI is an example of an interactive process. Interactive processes can be classified into foreground and background processes. The foreground process is the process that you are currently interacting with, and is using the terminal as its stdin (standard input) and stdout (standard output). A background process is not interacting with the user and can be in one of two states - paused or running.

## System Process or Daemon

The second general type of process that runs on Linux is a **System Process or Daemon** (daemon). Daemon is the term used to refer to processes that are running on the computer and provide services but do not interact with the console. Most server software is implemented as a daemon. Apache, Samba, and inn are all examples of daemons.

Any process can become a daemon as long as it is run in the background, and does not interact with the user. A simple example of this can be achieved using the [ls -R] command. This will list all subdirectories on the computer, and is similar to the [dir /s] command on Windows. This command can be set to run in the background by typing [ls -R &], and although technically you have control over the shell prompt, you will be able to do little work as the screen displays the output of the process that you have running in the background. You will also notice that the standard pause (ctrl+z) and kill (ctrl+c) commands do little to help you.

## Automatic Processes

**Automatic** processes are not connected to a terminal. Rather, these are tasks that can be queued into a spooler area, where they wait to be executed on a FIFO (first-in, first-out) basis. Such tasks can be executed using one of two criteria:

At certain date and time: done using the “at” command

At times when the total system load is low enough to accept extra jobs: done using the Cron command. By default, tasks are put in a queue where they wait to be executed until the system load is lower than 0.8. In large environments, the system administrator may prefer cron job processing when large amounts of data have to be processed or when tasks demanding a lot of system resources have to be executed on an already loaded system. Cron job processing is also used for optimizing system performance.

## Parent and Child Process

- The Process which starts or creates another process is called **parent process** and the one which got created is known as **child process**.
- Every process will be having a parent process except **init/systemd** process.
- The **init/systemd** process is the parent of all the process in the system. It is the first process which gets started by the kernel at the time of booting
- The PID of init/systemd will be **1**.
- Only after init/systemd process gets started the remaining process are called by it, and hence it is responsible for all the remaining processes in the system.

## LAB WORK:-

### To monitor the process using ps command

- The ps command gives the running process of the present terminal and present command.  
The syntax for ps command is

```
#ps
```

```
[root@ linux ~]# ps
 PID TTY      TIME CMD
 10951 pts/0    00:00:00 bash
 11523 pts/0    00:00:00 ps
[root@ linux ~]#
```

### To see total number of processes running in the system

- The possible options which can be used with ps command are

```
#ps -a
```

```
[root@ linux ~]# ps -a
 PID TTY      TIME CMD
 11545 pts/0    00:00:00 ps
[root@ linux ~]#
```

### To see the processes running by the logged in user (ex root)

- #ps -u <user name>

```
#ps -u musab
```

#ps -u ( if no name is given it will show the processes of the logged in user)

```
[root@ linux ~]# ps -u musab
 PID TTY      TIME CMD
 11566 pts/1    00:00:00 bash
 11591 pts/1    00:00:00 vim
[root@ linux ~]#
```

```
[root@ linux ~]# ps
 PID TTY      TIME CMD
 10951 pts/0    00:00:00 bash
 11523 pts/0    00:00:00 ps
[root@ linux ~]#
```

### To see which process are attached with some terminals (tty) and which are not

- #ps -x

```
[root@ linux ~]# ps -x
 PID TTY      STAT   TIME COMMAND
  1 ?        Ss     0:03 /sbin/init
  2 ?        S      0:00 [kthreadd]
  3 ?        S      0:00 [migration/0]
 2015 tty2   Ss+    0:00 /sbin/mingetty /dev/tty2
 2017 tty3   Ss+    0:00 /sbin/mingetty /dev/tty3
```

Note: The process which are showing "?" are not attached to any tty, mostly background processes

To see which process are running by a particular group

- #ps -G <group name> or #pgrep -G <group name>
- #ps -G musab or #pgrep -G musab

```
[root@ linux ~]# ps -G musab
  PID TTY      TIME CMD
11566 pts/1    00:00:00 bash
11591 pts/1    00:00:00 vim
[root@ linux ~]# pgrep -G musab
11566
11591
```

To see the auxiliary information of all the process, like cpu and memory consumptions

#ps -aux

```
[root@ linux ~]# ps -aux
USER      PID %CPU %MEM      VSZ   RSS TTY      STAT START   TIME COMMAND
root        1  0.0  0.3 126580  7420 ?        Ss  05:42  0:09 /usr/lib/systemd/systemctl --switched-root --system --deserialize 21
root        2  0.0  0.0      0     0 ?        S   05:42  0:00 [kthreadd]
root        3  0.0  0.0      0     0 ?        S   05:42  0:01 [ksoftirqd/0]
root        7  0.0  0.0      0     0 ?        S   05:42  0:00 [migration/0]
root        8  0.0  0.0      0     0 ?        S   05:42  0:00 [rcu_bh]
root        9  0.0  0.0      0     0 ?        S   05:42  0:00 [rcuob/0]
root       10  0.0  0.0      0     0 ?        S   05:42  0:00 [rcuob/1]
```

## Signals in Linux

- Signals are a way of sending simple messages to processes. Most of these messages are already defined and can be found in <linux/signal.h>. However, signals can only be processed when the process is in user mode. If a signal has been sent to a process that is in kernel mode, it is dealt with immediately on returning to user mode.
- Every signal has a unique signal name, an abbreviation that begins with SIG (SIGINT for interrupt signal, for example). Each signal name is a macro which stands for a positive integer - the signal number for that kind of signal. Your programs should never make assumptions about the numeric code for a particular kind of signal, but rather refer to them always by the names defined. This is because the number for a given kind of signal can vary from system to system, but the meanings of the names are standardized and fairly uniform.
- Signals can be generated by the process itself, or they can be sent from one process to another. A variety of signals can be generated or delivered, and they have many uses for programmers. (To see a complete list of signals in the Linux® environment, uses the command kill -l.)

- There are total 64 signals in Linux, the list of all the signal can be seen by  
#kill -l

```
[root@ linux ~]# kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
[root@ktlinux ~]#
```

### Few Important Signals with its descriptions:

Signal	Value	Action	Comment
<b>SIGHUP</b>	1	Term	Hangup detected on controlling terminal or death of controlling process
<b>SIGINT</b>	2	Term	Interrupt from keyboard
<b>SIGQUIT</b>	3	Core	Quit from keyboard
<b>SIGILL</b>	4	Core	Illegal Instruction
<b>SIGABRT</b>	6	Core	Abort signal from abort(3)
<b>SIGFPE</b>	8	Core	Floating point exception
<b>SIGKILL</b>	9	Term	Kill signal
<b>SIGSEGV</b>	11	Core	Invalid memory reference
<b>SIGPIPE</b>	13	Term	Broken pipe: write to pipe with no readers
<b>SIGALRM</b>	14	Term	Timer signal from alarm(2)
<b>SIGTERM</b>	15	Term	Termination signal
<b>SIGUSR1</b>	30,10,16	Term	User-defined signal 1
<b>SIGUSR2</b>	31,12,17	Term	User-defined signal 2
<b>SIGCHLD</b>	20,17,18	Ign	Child stopped or terminated
<b>SIGCONT</b>	19,18,25	Cont	Continue if stopped
<b>SIGSTOP</b>	17,19,23	Stop	Stop process
<b>SIGTSTP</b>	18,20,24	Stop	Stop typed at tty
<b>SIGTTIN</b>	21,21,26	Stop	tty input for background process
<b>SIGTTOU</b>	22,22,27	Stop	tty output for background process

### The most common signals used are

- 1 for reloading the process
- 9 for killing the process
- 15 for Terminating the process
- 20 for stopping the process

### To kill the process completely

- To kill the signal
- First find out the process running in the system, let's say by a user  
`#ps -u <user name>`  
`#ps -u musab`  
`#kill <signal no> <process id>`  
`#kill -9 11591`

```
[root@ linux ~]# ps -u musab
 PID TTY      TIME CMD
11566 pts/1    00:00:00 bash
[11591 pts/1]  00:00:00 vim
[root@ linux ~]# kill -9 11591
[root@ linux ~]# ps -u musab
 PID TTY      TIME CMD
11566 pts/1    00:00:00 bash
[root@ linux ~]# █
```

Likewise you can use other signals to kill the process like

```
#kill -15 <pid>
#kill -1 <pid>
```

### To stop the process using a signal no. 20

- To stop a process first login as a normal user and start a process  
`#su - musab`  
`#cat > hello`

```
[musab@ linux ~]$ cat > hello
█
```

- Check its pid and kill it by using 20, `#ps -u musab`  
`#kill -20`

```
[root@ linux ~]# ps -u musab
 PID TTY      TIME CMD
11566 pts/1    00:00:00 bash
[12041 pts/1]  00:00:00 cat
[root@ linux ~]# kill -20 12041
```

- check its effect at the user's console

```
[musab@ linux ~]$ cat > hello
[1]+  Stopped                  cat > hello
```

- Restart the process continue working

```
#fg <pid>
#fg 1
```

```
[musab@ linux ~]$ fg 1
cat > hello
Hi Maarij How are you █
```

## Setting up the Priority of a Process

- When talking about processes priority is all about managing processor time. The Processor or CPU is like a human juggling multiple tasks at the same time. Sometimes we can have enough room to take on multiple projects. Sometimes we can only focus on one thing at a time. Other times something important pops up and we want to devote all of our energy into solving that problem while putting less important tasks on the back burner.
- In Linux we can set guidelines for the CPU to follow when it is looking at all the tasks it has to do. These guidelines are called **niceness** or **nice value**. The Linux niceness scale goes from **-20 to 20**. **The lower the number the more priority that task gets. If the nice value is high number like 20 the task will be set to the lowest priority and the CPU will process it whenever it gets a chance. The default nice value is zero.**
- By using this scale we can allocate our CPU resources more appropriately. Lower priority programs that are not important can be set to a higher nice value, while high priority programs like daemons and services can be set to receive more of the CPU's focus. You can even give a specific user a lower nice value for all of his/her processes so you can limit their ability to slow down the computer's core services.
- There are two options to reduce/increase value of a process. You can either do it using the **nice** command or the **renice** command.

### LAB WORK:-

#### To schedule a priority of a process before starting it

- To set a priority to a process before starting it, the syntax is  
**#nice -n <nice value range (-20 to 20)> <command>**  
**#nice -n 5 cat > myfile**

```
[root@ linux ~]# nice -n 5 cat > myfile
Hello World
Welcome to linux Technologies
```

- Log in to other terminal and check the nice value for the above command/ process.  
**#ps -elf**

F S	UID	PID	PPID	C PRI	NI	ADDR	SZ	WCHAN	STIME	TTY	TIME	CMD
4 S	root	1	0	0 80	0 -	707	-	Oct14 ?		00:00:03	/sbin/init	
1 S	root	2	0	0 80	0 -	0	-	Oct14 ?		00:00:00	[kthreadd]	
1 S	root	13152	2	0 80	0 -	0	-	01:56 ?		00:00:00	[flush-253:0]	
0 S	root	13155	10951	0 85	5 -	1010	-	01:56 pts/0		00:00:00	cat	
1 S	root	13163	2	0 80	0 -	0	-	01:58 ?		00:00:00	[flush-253:3]	

### To change the nice value of any process while it is running.

- To reschedule the nice value of existing process, first check the PID of that process by running **#ps -elf** command.
- As from previous task we know the PID of cat command i.e. **13155**
- Use the following command to renice the value of a cat command which is still running  
**#renice <nice value (-20 to 19)> <PID>**  
**#renice 2 13155**

```
[root@ linux ~]# renice 2 13155
13155: old priority 5, new priority 2
[root@ linux ~]#
```

### Important monitoring commands

There are four critical resources of the system which is also known as 4 critical bottle neck

1. CPU
2. MEMORY
3. I/O (INPUT OUTPUT)
4. NETWORK

1. CPU: To monitor cpu there are following commands

**#ps, sar, lscpu, /proc/cpuinfo**  
**#sar (system activity report)**

Linux 3.10.0-229.el7.x86\_64 (localhost.localdomain) 10/03/2016 \_x86\_64\_ (1 CPU)

07:22:12 AM        LINUX RESTART

07:30:01 AM	CPU	%user	%nice	%system	%iowait	%steal	%idle
07:40:01 AM	all	0.07	0.00	0.04	0.00	0.00	99.90
07:50:01 AM	all	0.07	0.00	0.02	0.00	0.00	99.91

### To see continuous output of sar

**#sar 1 (1 sec is the time interval)**

root@ ~]# sar 1

Linux 3.10.0-229.el7.x86\_64 (kernel.kt.com) 10/03/2016 \_x86\_64\_ (1 CPU)

01:56:04 PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
01:56:05 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:06 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:07 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:08 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:09 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:10 PM	all	0.00	0.00	0.00	0.00	0.00	100.00

### To restrict the output for 3 counts for every second

**#sar 1 3 (where 1 is time interval and 3 is counts)**

root@ ~]# sar 1 3

Linux 3.10.0-229.el7.x86\_64 (mlinux7.mb.com) 10/03/2016 \_x86\_64\_ (1 CPU)

02:01:07 PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
02:01:08 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
02:01:09 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
02:01:10 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
Average:	all	0.00	0.00	0.00	0.00	0.00	100.00

To see the details of cpu

#lscpu

```
[root@ ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:  0
Thread(s) per core:   1
Core(s) per socket:   1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 42
Model name:            Intel Xeon E312xx (Sandy Bridge)
Stepping:               1
CPU MHz:               3192.746
BogoMIPS:              6385.49
Hypervisor vendor:    KVM
Virtualization type:  full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
NUMA node0 CPU(s):    0
```

#cat /proc/cpuinfo

```
[root@ ~]# cat /proc/cpuinfo
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 42
model name    : Intel Xeon E312xx (Sandy Bridge)
stepping       : 1
microcode     : 0x1
cpu MHz       : 3192.746
cache size    : 4096 KB
physical id   : 0
siblings       : 1
core id       : 0
cpu cores     : 1
apicid         : 0
initial apicid: 0
fpu           : yes
fpu_exception : yes
cpuid level   : 13
```

2. Memory: To monitor memory following commands are used

#free, swapon -s , vmstat, /proc/meminfo

#vmstat

```
[root@ ~]# vmstat
procs -----memory----- swap-- io---- system-- cpu-----
 r b  swpd  free   buff  cache   si   so   bi   bo   in   cs us sy id wa st
 2 0    300 443576     32 270840    0    0   343   321   32   56  0  0 98  2  0
[root@ ~]#
```

To continuously see the output of vmstat for every 1 second

#vmstat 1

```
[root@ ~]# vmstat 1
procs -----memory----- ---swap-- -----io---- -system-- -----cpu-----
 r b swpd free buff cache si so bi bo in cs us sy id wa st
 2 0 300 443452 32 270872 0 0 341 319 32 56 0 0 98 2 0
 0 0 300 443436 32 270872 0 0 0 0 22 34 0 0 100 0 0
 0 0 300 443468 32 270872 0 0 0 0 17 32 0 0 100 0 0
```

To restrict the output of vmstat to 3 counts with time interval of 1 second

#vmstat 1 3 (where 1 is time interval and 3 is count)

```
[root@ ~]# vmstat 1 3
procs -----memory----- ---swap-- -----io---- -system-- -----cpu-----
 r b swpd free buff cache si so bi bo in cs us sy id wa st
 2 0 300 442624 32 271312 0 0 338 317 32 56 0 0 98 2 0
 0 0 300 442624 32 271312 0 0 0 0 9 12 0 0 100 0 0
 0 0 300 442600 32 271312 0 0 0 0 12 19 0 0 100 0 0
[root@ ~]#
```

#vmstat -s and #proc/meminfo

```
[root@node1 ~]# vmstat -s
 1882624 K total memory
 349340 K used memory
 348568 K active memory
 158388 K inactive memory
1073192 K free memory
 2116 K buffer memory
 457976 K swap cache
1257468 K total swap
 0 K used swap
1257468 K free swap
 254359 non-nice user cpu ticks
 215 nice user cpu ticks
 30618 system cpu ticks
 913874 idle cpu ticks
 4659 IO-wait cpu ticks
 0 IRQ cpu ticks
 3442 softirq cpu ticks
 0 stolen cpu ticks
304059 pages paged in
152803 pages paged out
 0 pages swapped in
 0 pages swapped out
3926937 interrupts
 2651686 CPU context switches
1547276963 boot time
 22138 forks
```

```
[root@ ~]#
[root@ ~]# cat /proc/meminfo
MemTotal: 1017216 kB
MemFree: 443008 kB
MemAvailable: 543564 kB
Buffers: 32 kB
Cached: 201244 kB
SwapCached: 300 kB
Active: 138320 kB
Inactive: 312216 kB
Active(anon): 16916 kB
Inactive(anon): 239268 kB
Active(file): 121404 kB
Inactive(file): 72948 kB
Unevictable: 0 kB
Mlocked: 0 kB
```

### 3. I/O: To Monitor I/O devices following commands are used **#fdisk, parted, df -h, iostat, lsblk, lsusb, lspci**

**#iostat : to see the statistic of i/o devices**

```
[root@ ~]# iostat
Linux 3.10.0-229.el7.x86_64 ( mlinux6.mb.com ) 10/03/2016 _x86_64_ (1 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle
          0.23   0.00   0.24   1.84   0.00  97.68

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
scd0        2.82    305.39      0.00  7800586      0
sda         2.35     28.08    392.75  717296 10031951
```

**To continuously repeat the output of iostat every second and to restrict the same for 3 counts**

**#iostat 1, #iostat 1 3**

```
[root@ ~]# iostat 1
Linux 3.10.0-229.el7.x86_64 ( mlinux7.mb.com ) 10/03/2016 _x86_64_ (1 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle
          0.23   0.00   0.24   1.83   0.00  97.69

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
scd0        2.81    304.19      0.00  7800586      0
sda         2.34     27.97    391.20  717304 10031981
dm-1        0.07     0.09      4.22   2217  108202
dm-0        0.04     0.07     22.19   1833  569002

avg-cpu: %user %nice %system %iowait %steal %idle
          0.00   0.00   0.00   0.00   0.00 100.00

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
scd0        0.00     0.00      0.00      0      0
sda         0.00     0.00      0.00      0      0
dm-1        0.00     0.00      0.00      0      0
dm-0        0.00     0.00      0.00      0      0
```

**To see a particular device statistics continuously**

**#iostat -d <dev name> 1**

**#iostat -d sda3 1**

```
[root@ ~]# iostat -d sda3 1
Linux 3.10.0-229.el7.x86_64 ( mlinux7.mb.com ) 10/03/2016 _x86_64_ (1 CPU)

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
sda3        0.01     0.20      0.02    5072    520

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
sda3        0.00     0.00      0.00      0      0

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
sda3        0.00     0.00      0.00      0      0
```

To restrict the same for 3 counts

#iostat -d <dev name> 1 3 (where 1 is time interval and 3 is counts)

#iostat -d sda3 1 3

```
[root@ ~]# iostat -d sda3 1 3
Linux 3.10.0-229.el7.x86_64 (mlinux7.mb.com) 10/03/2016 _x86_64_ (1 CPU)

Device:      tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda3        0.01     0.20       0.02     5072       520

Device:      tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda3        0.00     0.00       0.00       0          0

Device:      tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda3        0.00     0.00       0.00       0          0

[root@ ~]#
```

#### 4. To monitoring network following commands

#ifconfig, ethtool, mii-tool, ping, netstat , route

To see the network configuration

#netstat

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0  mlinux7.mb.com:ssh      192.168.30.36:63426 ESTABLISHED
tcp      0      64 mlinux7.mb.com:ssh      192.168.30.36:63819 ESTABLISHED
tcp      0      0  mlinux7.mb.com:ssh      192.168.30.36:63279 ESTABLISHED

Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type      State         I-Node      Path
unix    2      [ ]     DGRAM          12914      @/org/freedesktop/systemd/notify
unix    5      [ ]     DGRAM          6609       /run/systemd/journal/socket
```

To see interface statistics

#netstat -i

```
[root@ ~]# netstat -i
Kernel Interface table
Iface      MTU     RX-OK RX-ERR RX-DRP RX-OVR     TX-OK TX-ERR TX-DRP TX-OVR Flg
ens3      1500    63082      0    171 0      42175      0      0      0 BMRU
lo        65536   2124       0      0 0      2124      0      0      0 LRU
[root@ ~]#
```

To see routing table

#netstat -rn

```
[root@ ~]# netstat -rn
Kernel IP routing table
Destination     Gateway            Genmask          Flags  MSS Window irtt Iface
0.0.0.0         192.168.106.1    0.0.0.0         UG        0 0          0 ens3
192.168.106.0  0.0.0.0          255.255.255.0   U        0 0          0 ens3
[root@ ~]#
```

**Routing table can also be seen by**

#route

```
[root@ ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
default         192.168.106.1   0.0.0.0         UG    100      0      0 ens3
192.168.106.0  0.0.0.0        255.255.255.0  U     100      0      0 ens3
[root@ ~]#
```

## Monitoring the process using top command

- When you need to see the running processes on your Linux in real time, you have top as your tool for that.
- top also displays other info besides the running processes, like free memory both physical and swap

### **Monitoring all process using top command**

- To monitor all processes in the system use the following command

#top

```
top - 02:23:18 up 1 day, 13:57, 3 users, load average: 0.01, 0.00, 0.23
Tasks: 273 total, 1 running, 272 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.4%us, 0.5%sy, 0.0%ni, 98.8%id, 0.3%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 543948k total, 526204k used, 17744k free, 11748k buffers
Swap: 2097144k total, 49064k used, 2048080k free, 129928k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	2828	1192	1044	S	0.0	0.2	0:03.15	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:00.14	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
6	root	20	0	0	0	0	S	0.0	0.0	0:00.60	events/0
7	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuset
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khelper

**The first line in top:**

```
top - 02:23:18 up 1 day, 13:57, 3 users, load average: 0.01, 0.00, 0.23
```

- “**02:23:18**” is the current time; “**up 1 day**” shows how long the system has been up for; “**3 user**” how many users are logged in; “**load average: 0.01, 0.00, 0.23**” the load average of the system (1minute, 5 minutes, 15 minutes).

**The second line in top:**

```
Tasks: 273 total, 1 running, 272 sleeping, 0 stopped, 0 zombie
```

- Shows the number of processes and their current state.

**The third line in top:**

```
Cpu(s): 0.4%us, 0.5%sy, 0.0%ni, 98.8%id, 0.3%wa, 0.0%hi, 0.0%si, 0.0%st
```

- Shows CPU utilization details. “9.5%us” user processes are using 9.5%; “31.2%sy” system processes are using 31.2%; “27.0%id” percentage of available cpu; “7.6%wa” time CPU is waiting for IO.

**The fourth and fifth lines in top:**

Mem:	543948k total,	526204k used,	17744k free,	11748k buffers
Swap:	2097144k total,	49064k used,	2048080k free,	129928k cached

- “**543948k total**” is total memory in the system; “**526204K used**” is the part of the RAM that currently contains information; “**17744k free**” is the part of RAM that contains no information; “**17748K buffers and 129928k cached**” is the buffered and cached data for IO.

**By default, top starts by showing the following task's property:**

Field	Description
PID	<b>Process ID</b>
USER	<b>Effective User ID</b>
PR	<b>Dynamic priority</b>
NI	<b>Nice value, also known as base priority</b>
VIRT	<b>Virtual Size of the task. This includes the size of process's executable binary, the data area and all the loaded shared libraries.</b>
RES	<b>The size of RAM currently consumed by the task. Swapped out portion of the task is not included.</b>
SHR	<b>Some memory areas could be shared between two or more task, this field reflects that shared areas. The example of shared area are shared library and SysV shared memory.</b>
S	<b>Task status</b>
%CPU	<b>The percentage of CPU time dedicated to run the task since the last top's screen update.</b>
%MEM	<b>The percentage of RAM currently consumed by the task.</b>
TIME+	<b>The total CPU time the task has been used since it started. "+" sign means it is displayed with hundredth of a second granularity. By default, TIME/TIME+ doesn't account the CPU time used by the task's dead children.</b>
Command	<b>Showing program names</b>

## Interacting with TOP

Now that we are able to understand the output from TOP lets learn how to change the way the output is displayed.

Use the following key while running top and the output will be sorted in real time.

- **h - help**
- **M – Sort by memory usage**
- **P – Sort by CPU usage**
- **T – Sort by cumulative time**
- **z – Color display**
- **L – locate/search a string**
- **r – to renice a process**
- **k – Kill a process**
- **q – quit**

To kill the process with PID 21, then press “k” and a prompt will ask you for the PID number, and enter 21. When asked about singal number give 9 or 15

```
top - 02:54:53 up 1 day, 14:29, 3 users, load average: 0.07, 0.02, 0.01
Tasks: 272 total, 1 running, 271 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 543948k total, 522892k used, 21056k free, 12232k buffers
Swap: 2097144k total, 49688k used, 2047456k free, 126492k cached
```

PID to kill: 21

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	[kthreadd]

```
top - 02:54:53 up 1 day, 14:29, 3 users, load average: 0.07, 0.02, 0.01
Tasks: 272 total, 1 running, 271 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 543948k total, 522892k used, 21056k free, 12232k buffers
Swap: 2097144k total, 49688k used, 2047456k free, 126492k cached
```

Kill PID 21 with signal [15]: 9

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
-----	------	----	----	------	-----	-----	---	------	------	-------	---------

To renice a process with PID 4, then press “r” and a prompt will ask you for PID enter 4 and press enter. When prompted for renice value give any value .

```
Swap: 2097144k total, 49688k used, 2047456k free, 126496k cached
```

PID to renice: 4

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	[kthreadd]
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	[migration/0]
4	root	20	0	0	0	0	S	0.0	0.0	0:00.15	[ksoftirqd/0]

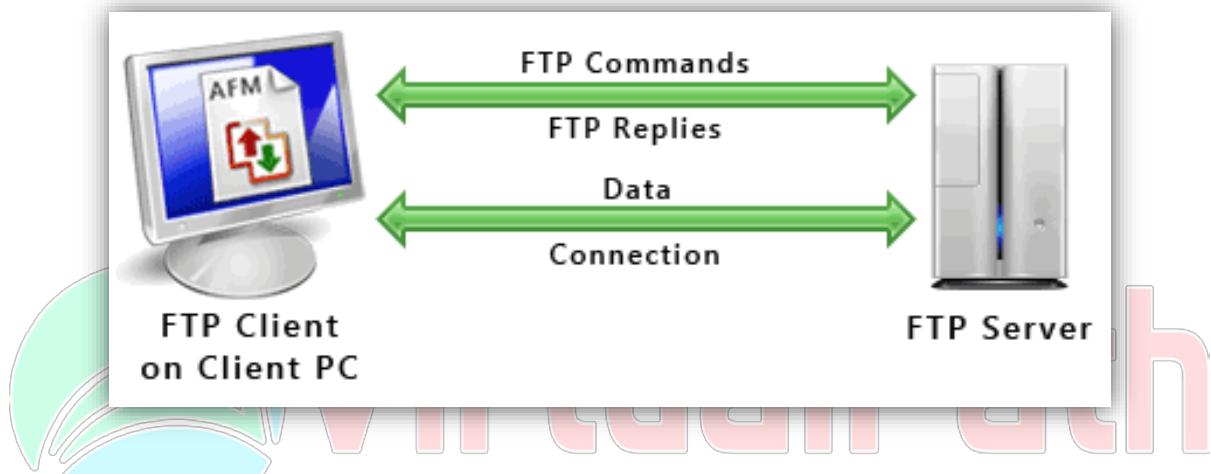
Renice PID 4 to value: -2

4	root	18	-2	0	0	0	S	0.0	0.0	0:00.15	[ksoftirqd/0]
---	------	----	----	---	---	---	---	-----	-----	---------	---------------

Find out more on top command from internet and keep practicing

## FTP (File Transfer Protocol) SERVER

- **File Transfer Protocol (FTP)** is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet. FTP is built on client-server architecture and utilizes separate control and data connections between the client and server. FTP users may authenticate themselves using a clear-text sign-in protocol but can connect anonymously if the server is configured to allow it.
- In **Red hat Enterprise Linux**. You can access FTP from both the Command Line Interface mode and GUI mode.



- Usually, the FTP server, which stores files to be transferred, uses two ports for the transferring purpose, one for Commands and the other for sending and receiving Data. Requests from client computers are received at the port 21 of the server, which is exclusively reserved for sending Commands; therefore, it is called the Command Port. Once an incoming request is received, the data requested or uploaded by the client computer is transferred through a separate port referred to as a Data Port. At this point, depending on the Active or Passive mode of the FTP connection, the port number used for the Data Transfer varies.
- Security is a major concern with any computer connected to the internet, therefore any computer connected to the internet should be protected by a Firewall. In order to connect to certain services, such as FTP, you have to allow those connections in the Firewall, on both the Client and Server side.
- Although a client's computer may not have a firewall enabled, a server should always have this enabled for maximum security. In order to connect to an FTP server that has a firewall enabled, you have to connect using a specific connection mode in your FTP program.
- There are different connection modes to choose from when connecting to an FTP server, typically either **"Active" or "Passive" mode**.

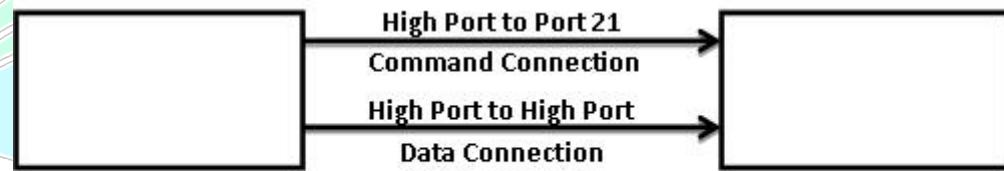
## What is Active FTP?



*Figure 01 – Active FTP Connection*

- Active FTP connection mode is where Command connection is initiated by the Client, and the Data connection is initiated by the Server. And as the server actively establishes the data connection with the Client, this mode is referred to as Active. The Client opens up a port higher than 1024, and through it connects to the port 21 or the command port of the Server. Then the Server opens up its port 20 and establishes a data connection to a port higher than 1024 of the Client. In this mode, Client must set its firewall settings to accept all the incoming connections that are received at the opened port.

## What is Passive FTP?



*Figure 02 – Passive FTP Connection*

- In the Passive FTP connection mode, the server acts entirely passively as the Command connection and the Data connection are both initiated and established by the Client. In this mode, Server listens for incoming requests through its port 21 (command port), and when a request is received for a data connection from the Client (using a high port), Server randomly opens up one of its High ports. Then Client initiates a data connection between the opened port of the Server and its own randomly selected port higher than 1024. In this mode, the Client does not have to change its firewall settings, as it only requires outgoing connections and the firewall do not block outgoing connections. However, the Server administrators must make sure that the Server allows incoming connections at all its opened ports.

## What is the difference between Active FTP and Passive FTP?

The difference between the Active FTP and Passive FTP is based on who initiates the Data connection between the Server and the Client. If data connection is initiated by the Server, the FTP connection is active, and if the Client initiates the Data connection, FTP connection is passive.

Depending on the Active or Passive mode of the connection, port used for Data connection changes. In an Active FTP, data connection is established between port 20 of the Server and High Port of the Client. On the other hand, in Passive FTP, data connection is established between a High port of the Server and a High port of the Client.

When using an Active FTP connection, firewall settings of the Client must be changed to accept all incoming connection to the Client, while in Passive FTP connection, the Server must allow all incoming connections to the Server. Most FTP servers prefer the Passive FTP connection due to security issues.



- **Use** : Ftp is used for uploading and downloading the files.
- **Limitation** : Directory cannot be uploaded or downloaded.
- **Package** : vsftpd
- **Daemon** : vsftpd (Very Secure Ftp daemon)
- **Port no** : 21 (Tcp) > 1024 (Udp, Random)
- **Configuration files** : /etc/vsftpd/vsftpd.conf  
/etc/vsftpd/user\_list  
/etc/vsftpd/ftpuser
- **Home directory** : /var/ftp (which will be created only when the package is installed)

## Steps to configuring ftp server for downloading files:

1. Install the package
2. Create some files in /var/ftp/pub directory
3. Restart the service
4. Make the service enable even after reboot of the system
5. Connect from client and access the files and download it

### Step1: Install the package

- Install the package using yum or rpm command.

```
#yum install vsftpd* -y
```

```
[root@ cl3 ~]# yum install vsftpd* -y
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package vsftpd.x86_64 0:2.2.2-6.el6_0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
| Package      | Arch | Version | Repository | Size |
|=====|=====|=====|=====|=====|
| Installing: |       |          |            |        |
| vsftpd       | x86_64 | 2.2.2-6.el6_0.1 | RHEL6    | 150 k |
|               |       |          |            |        |

Transaction Summary

  Installing : vsftpd-2.2.2-6.el6_0.1.x86_64                                1/1
RHEL6/productid
duration: 117(ms)
| 1.7 kB   00:00
Installed products updated.

  Installed:
    vsftpd.x86_64 0:2.2.2-6.el6_0.1

Complete!
```

Okay now we are done with the installation. Check it with

**#rpm -q vsftpd command**

```
[root@ cl3 ~]# rpm -q vsftpd
vsftpd-2.2.2-6.el6_0.1.x86_64
[root@ktcl3 ~]# ■
```

- If you don't have yum repository created, then installed it using rpm from RHEL 7 DVD

**Step2:** Copy or create some files in “/var/ftp/pub” directory

- Navigate to /var/ftp/pub directory and create some files in it
- ```
#cd /var/ftp/pub
#touch file{1..5}
```

```
[root@ cl3 ~]# cd /var/ftp/pub/
[root@ cl3 pub]# ls
[root@ cl3 pub]# touch file{1..5}
[root@ cl3 pub]# ls
file1 file2 file3 file4 file5
[root@ cl3 pub]#
```

**Step3:** Make the following changes in config to allow anonymous user in ftp

```
#vim /etc/vsftpd/vsftpd.conf
```

```
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).
anonymous_enable=YES
#
```

**Step4:** Start the ftp service and make it enabled after reboot

```
#systemctl enable vsftpd --now
```

```
[root@mlinux3 ~]# systemctl enable vsftpd --now
Created symlink /etc/systemd/system/multi-user.target.wants/vsftpd.service
e.
```

**Step4:** Add ftp service in firewall’s trusted list to avoid blocking via firewall

```
#firewall-cmd --add-service=ftp --permanent
```

```
#firewall-cmd --reload
```

```
[root@mlinux71 ~]# firewall-cmd --add-service=ftp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpcv6-client ftp ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

**Step6:** Connect from client and access the files and download it

- To access the ftp server the client should have “ftp” package installed. If not installed, install it using rpm, because yum will not work if ftp package is not installed.
- Check whether ftp package is installed or not

```
#rpm -q ftp
```

```
[root@ cl1 ~]# rpm -q ftp
package ftp is not installed
[root@ cl1 ~]#
```

- To install ftp package, either move to the package folder and installed it using rpm or run yum command if yum is configured

```
#rpm -ivh <package name> or # yum install ftp -y
```

```
[root@ cl1 ~]# rpm -ivh ftp-0.17-51.1.el6.x86_64.rpm
warning: ftp-0.17-51.1.el6.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID fd
431d51: NOKEY
Preparing... ## [100%]
1:ftp ## [100%]
```

- Check it by using rpm -q command

```
[root@ cl1 ~]# rpm -q ftp
ftp-0.17-51.1.el6.x86_64
```

- Connect to ftp server using its IP as shown below

```
#ftp <ftp server's IP>
```

```
#ftp 192.168.10.93
```

Use “ftp or anonymous” as login name

Press enter without giving any password

```
[root@ cl1 ~]# ftp 192.168.10.93
Connected to 192.168.10.93 (192.168.10.93).
220 (vsFTPd 2.2.2)
Name (192.168.10.93:root): ftp
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```



- Navigate to pub directory and check the files available

```
#cd pub
```

```
ftp> cd pub
250 Directory successfully changed.
ftp> ls
227 Entering Passive Mode (192,168,10,93,62,234).
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file1
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file2
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file3
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file4
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file5
226 Directory send OK.
ftp> 
```

Note: when you run ls command you can see that it showing that we are using Passive mode.

- Download some files using get or mget command

#get <file name> for single file

```
ftp> get file1
local: file1 remote: file1
227 Entering Passive Mode (192,168,10,93,221,15).
150 Opening BINARY mode data connection for file1 (0 bytes).
226 Transfer complete.
ftp> █
```

#mget <file names> for multiple files

Before going for mget turn off the interactive mode, otherwise it will ask permission for every file you are downloading. Use #prompt command to turn off interactive mode.

```
227 Entering Passive Mode (192,168,10,93,52,28).
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file1
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file2
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file3
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file4
-rw-r--r-- 1 0 0 0 Nov 03 12:58 file5
226 Directory send OK.
ftp> prompt
Interactive mode off.
ftp> mget file1 file2 file3
local: file1 remote: file1
227 Entering Passive Mode (192,168,10,93,244,230).
150 Opening BINARY mode data connection for file1 (0 bytes).
226 Transfer complete.
local: file2 remote: file2
227 Entering Passive Mode (192,168,10,93,65,88).
150 Opening BINARY mode data connection for file2 (0 bytes).
226 Transfer complete.
local: file3 remote: file3
227 Entering Passive Mode (192,168,10,93,154,73).
150 Opening BINARY mode data connection for file3 (0 bytes).
226 Transfer complete.
ftp> █
```



- Exit the ftp server and check whether the files are there or not

To exit the ftp server either use

#bye or #quit or ctrl+d key (shortcut for logout)

```
ftp> quit
221 Goodbye.
[root@ cl1 ~]# ls
anaconda-ks.cfg      file1          install.log.syslog  south
ap                   file2          karnataka        tamil
bharat               file3          ktr              Template
Desktop              ftp-0.17-51.1.el6.x86_64.rpm  Music
Documents            gana           Pictures         Videos
Downloads            install.log    Public
[root@ cl1 ~]# █
```

- Other important commands in ftp
   
ftp> !ls (To list client side information)
   
ftp> !pwd (To see client side present directory)
   
ftp> !mkdir (To make a directory at client side)
   
ftp> lcd (to change the client side directory note: without "!")
   
ftp>status( To check the status of ftp options)

To connect to the ftp server via web browser like firefox, type the ftp server's ip address as following

- <ftp://192.168.10.93>

## Configuring the ftp server for uploading a file

To upload the files in the ftp server the steps are:

**Step1:** Create an upload dir in the document root of ftp server i.e., /var/ftp

#mkdir upload

```
[root@ linux ~]# cd /var/ftp
[root@ linux ftp]# ls
pub
[root@ linux ftp]# mkdir upload
[root@ linux ftp]# ls
pub upload
[root@ linux ftp]#
```

**Step2:** Change the group to “ftp” and write permission to the “upload” directory

- Changing the group of upload to ftp

#chgrp <group name> <directory name>

#chgrp ftp upload

```
[root@ linux ftp]# ls -ld upload
drwxr-xr-x. 2 root root 4096 Nov  4 19:12 upload
[root@ linux ftp]# chgrp ftp upload
[root@ linux ftp]# ls -ld upload
drwxr-xr-x. 2 root ftp 4096 Nov  4 19:12 upload
[root@ linux ftp]#
```

- Adding the write permission to upload directory

#chmod g+w upload

```
[root@ linux ftp]# ls -ld upload
drwxr-xr-x. 2 root ftp 4096 Nov  4 19:12 upload
[root@ linux ftp]# chmod g+w upload
[root@ linux ftp]# ls -ld upload
drwxrwxr-x. 2 root ftp 4096 Nov  4 19:12 upload
[root@ linux ftp]#
```

### **Step3: Log into client machine, access ftp server and try to upload some files**

- Log into client machine and access the ftp server from the directory in which the files to be uploaded are there.

```
[root@ cl5 ~]# cd sample
[root@ cl5 sample]# ls
f1  tf2  f3  f4  f5
[root@ktcl5 sample]# ftp 192.168.10.98
Connected to 192.168.10.98 (192.168.10.98).
220 (vsFTPd 2.2.2)
Name (192.168.10.98:root): ftp
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

- Navigate to upload directory and try to upload some files  
Once you logged into ftp and if you are not sure what is names of the files you want to upload then use “#!ls” command to see the content of the directory from which you have logged into ftp server.

```
ftp> cd upload
250 Directory successfully changed.
ftp> !ls
f1  f2  f3  f4  f5
ftp> prompt
Interactive mode off.
ftp> mput  f1  f2  f3
local: ktf1 remote:  f1
227 Entering Passive Mode (192,168,10,98,138,43).
550 [Permission denied.]
local: ktf2 remote:  f2
227 Entering Passive Mode (192,168,10,98,87,254).
550 [Permission denied.]
local: ktf3 remote:  f3
227 Entering Passive Mode (192,168,10,98,166,28).
550 [Permission denied.]
ftp> ■
```

- “Permission denied” is because the upload permission in the ftp configuration file is not enabled in the ftp server. So, navigate to the ftp configuration file and change the following attributes in it.

#vim /etc/vsftpd/vsftpd.conf

Uncomment (remove the #) the following line

```
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
# When SELinux is enforcing check for SE bool allow_ftpd_anon_write, allow_ftpd_full_access
#anon upload enable=YES
```

```
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
# When SELinux is enforcing check for SE bool allow_ftpd_anon_write, allow_ftpd_full_access
anon_upload_enable=YES
```

- **Restart the ftp service**

```
#systemctl restart vsftpd.service
```

```
[root@node1 ~]# systemctl restart vsftpd
```

#### **Step4: Login to client system and try again to upload the files into ftp server**

```
[root@ cl5 sample]# ls
f1 f2 f3 f4 f5
[root@ktcl5 sample]# ftp 192.168.10.98
Connected to 192.168.10.98 (192.168.10.98).
220 (vsFTPd 2.2.2)
Name (192.168.10.98:root): ftp
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd upload
250 Directory successfully changed.
ftp> [prompt]
Interactive mode off.
ftp> [put f1]
local: f1 remote: f1
227 Entering Passive Mode (192,168,10,98,192,136).
553 Could not create file.
ftp> [ ]
```



If SELinux is enabled in the ftp server, this error “Could not create file” will be displayed.  
To solve above error log into server and change the following permission

- Add read write permission in context of upload directory using following command

```
#chcon -t public_content_rw_t
```

```
[root@ linux ftp]# ls -ldZ upload
drwxrwxr-x. root ftp unconfined_u:object_r:public_content_t:s0 upload
[root@ linux ftp]# chcon -t public_content_rw_t upload
[root@ linux ftp]# ls -ldZ upload
drwxrwxr-x. root ftp unconfined_u:object_r:public_content_t:s0 upload
[root@ linux ftp]# [ ]
```

- Check the Booleans for ftp using following command

```
#getsebool -a |grep ftp
```

```
[root@node1 ~]# getsebool -a |grep ftp
ftpd_anon_write --> off
ftpd_connect_all_unreserved --> off
ftpd_connect_db --> off
ftpd_full_access --> off
ftpd_use_cifs --> off
```

- Make the above Booleans value as “on”

- To make it on use the following command

```
#setsebool -P ftpd_anon_write on; setsebool -P ftpd_full_access on
```

```
[root@node1 ~]# setsebool -P ftpd_anon_write on; setsebool -P ftpd_full_access on
[root@node1 ~]# getsebool -a |grep ftp
ftpd_anon_write --> on
ftpd_connect_all_unreserved --> off
ftpd_connect_db --> off
ftpd_full_access --> on
ftpd_use_cifs --> off
```

- Finally login into client machine, access the ftp server and try uploading the files in it.

```
[root@ cl5 sample]#
[root@ cl5 sample]# ftp 192.168.10.98
Connected to 192.168.10.98 (192.168.10.98).
220 (vsFTPd 2.2.2)
Name (192.168.10.98:root): ftp
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd upload
250 Directory successfully changed.
ftp> !ls
    f1    f2    f3    f4    f5
ftp> prompt
Interactive mode off.
ftp> mput  f1  f2
local:  f1 remote:  f1
227 Entering Passive Mode (192,168,10,98,121,220)
[50 Ok to send data.
226 Transfer complete.
local:  f2 remote:  f2
227 Entering Passive Mode (192,168,10,98,53,112).
150 Ok to send data.
ftp> ls
227 Entering Passive Mode (192,168,10,98,207,111).
150 Here comes the directory listing.
-rw-----  1 14      50          0 Nov  04 16:28 f1
-rw-----  1 14      50          0 Nov  04 16:28 f2
226 Directory send OK.
```



Okay now you've made an ftp server for uploading files as well

## Allowing root access to the ftp server

- By default root user is blocked to be used in ftp user, try logging with root in ftp server

```
[root@ cl5 sample]# ftp 192.168.10.98
Connected to 192.168.10.98 (192.168.10.98).
220 (vsFTPd 2.2.2)
Name (192.168.10.98:root): root
530 Permission denied.
login failed.
ftp> ■
```

- To Allow the root access to ftp server edit the “/etc/vsftpd/user\_list” and “/etc/vsftpd/ftpuser” and just add the comment (#mark) before “root”

#vim /etc/vsftpd/user\_list

```
# vsftpd userlist
# If userlist_deny=NO, only allow users in this file
# If userlist_deny=YES (default), never allow users in this file, and
# do not even prompt for a password.
# Note that the default vsftpd pam config also checks /etc/vsftpd/ftpusers
# for users that are denied.
#root
bin
daemon
```

#vim /etc/vsftpd/ftpuser

```
# Users that are not allowed to login via ftp
#root
bin
daemon
```

Note: - restart the service #*systemctl restart vsftpd.service*

- Now try login from client into ftp server as root

```
[root@ktcl5 ~]# ftp 192.168.10.98
Connected to 192.168.10.98 (192.168.10.98).
220 (vsFTPd 2.2.2)
Name (192.168.10.98:root): root
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ■
```

- **Blocking or Denying access to a user in ftp**

Go to `/etc/vsftpd/ftpuser` and `/etc/vsftpd/user_list` and enter the name which you want to block

```
#vim /etc/vsftpd/ftpuser
# Users that are not allowed to login via ftp
#root
myuser
bin
daemon
adm
lp

#vim /etc/vsftpd/user_list
# vsftpd userlist
# If userlist_deny=NO, only allow users in this file
# If userlist_deny=YES (default), never allow users in this file, and
# do not even prompt for a password.
# Note that the default vsftpd pam config also checks /etc/vsftpd/ftpusers
# for users that are denied.
#root
myuser
bin
daemon
adm
```

- **Restart the ftp service**

```
#systemctl restart vsftpd
```

```
[root@mlinux71 ~]# systemctl restart vsftpd
[root@mlinux71 ~]#
```

- Try logging with the blocked user

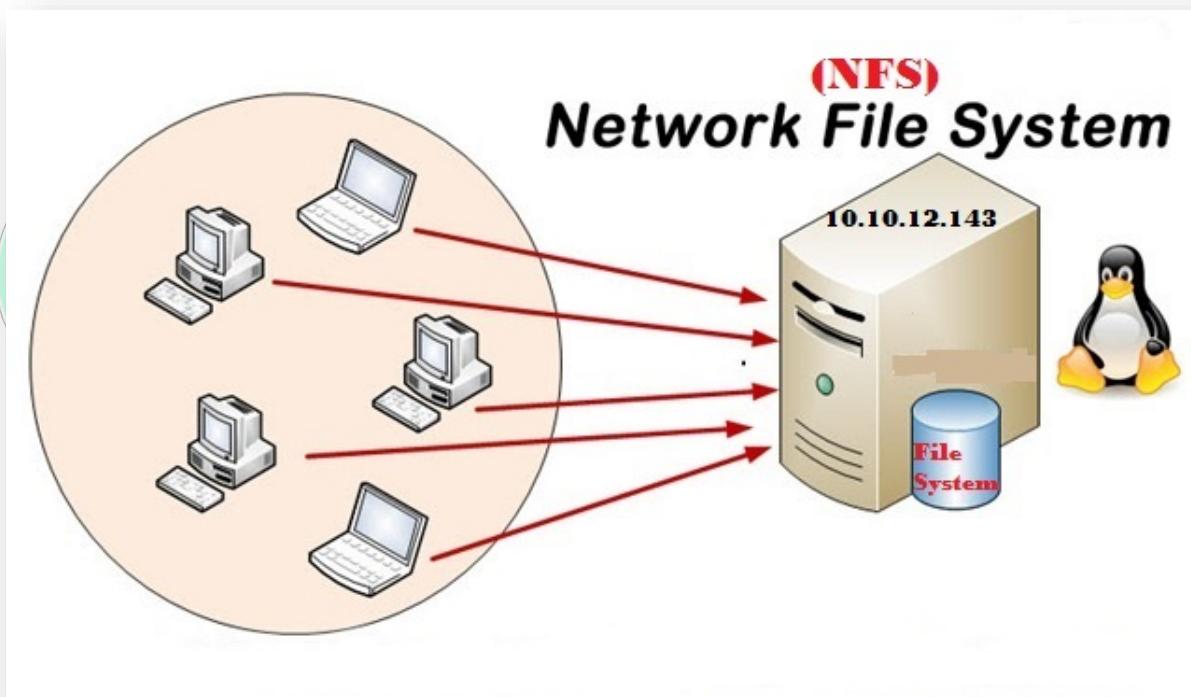
```
[root@mlinux72 ~]# ftp 192.168.106.81
Connected to 192.168.106.81 (192.168.106.81).
220 (vsFTPd 3.0.2)
Name (192.168.106.81:root): myuser
530 Permission denied.
Login failed.
ftp>
```

*You can also use browser to connect to the ftp server; or some 3<sup>rd</sup> party software can be used like winscp or filezilla to connect from windows clients.*

*That's all folks*

## NFS (NETWORK FILE SYSTEM/SHARING)

- NFS stands for Network File System, and is a way to share files between machines as if they were on your local hard drive. Linux can be both an NFS server and an NFS client, which means that it can export filesystems to other systems, and mount filesystems exported from other machines.
- For example **NFS server** could be a Linux system and UNIX could be a client. But it can't be a window system because windows is not NFS compatible. The NFS server exports one or more directories to the client systems, and the client systems mount one or more of the shared directories to local directories called mount points. After the share is mounted, all I/O operations are written back to the server, and all clients notice the change as if it occurred on the local filesystem.
- A manual refresh is not needed because the client accesses the remote filesystem as if it were local. Because access is granted by IP address, a username and password are not required. However, there are security risks to consider because the **NFS server** knows nothing about the users on the client system.



A Typical view of the NFS structure in Linux/Unix system

### Profile for NFS:

- |                      |   |                                                                               |
|----------------------|---|-------------------------------------------------------------------------------|
| • Package            | : | nfs-utils                                                                     |
| • Daemons            | : | nfs-server.service<br>rpc.nfsd, rpc.mountd, rpc.statd, rpc.lockd, rpc.rquotad |
| • Port number        | : | 2049                                                                          |
| • Configuration File | : | /etc/exports                                                                  |
| • Other imp files    | : | /var/lib/nfs/etab, /var/lib/nfs/rmtab                                         |

## Steps to configure NFS server:

**Step1:** check and if needed install the NFS package using yum or rpm.

**Step2:** Create a directory and add some data in it.

**Step3:** Export the directory by editing /etc/exports file and using exportfs command

**Step4:** Restart the services and make it permanent.

**Step1:** Install the NFS package.

- Check whether the package is installed

```
#rpm -q nfs-utils
```

```
[root@ cl1 ~]# rpm -q nfs-utils
nfs-utils-1.2.3-7.el6.x86_64
[root@ cl1 ~]#
```

- If it is not installed use following command to install it

```
#yum install nfs-utils* -y
```

**Step2:** Create a directory or create a partition and mount it and make a mount point and add data to it.

- Create a partition, format it and mount it, access the mount point and add data to it  
**#fdisk /dev/vda create a partition**

|                   |             |             |               |           |              |
|-------------------|-------------|-------------|---------------|-----------|--------------|
| /dev/vda1         | 3812        | 3927        | 931738+       | 8e        | Linux LVM    |
| /dev/vda2         | 3928        | 3979        | 417658+       | 8e        | Linux LVM    |
| <b>/dev/vda13</b> | <b>3980</b> | <b>4056</b> | <b>618471</b> | <b>83</b> | <b>Linux</b> |

- Update the partition table and format it  
**#partx -a /dev/vda or #partprobe /dev/vda**  
**#mkfs.ext4/xfs /dev/vda13**
- Create a directory and mount the partition over it and also make it permanent in /etc/fstab

```
[root@ cl1 ~]# mkdir /mydir
[root@ cl1 ~]# vim /etc/fstab
[root@ cl1 ~]# mount -a
[root@ cl1 ~]# mount
/dev/mapper/vg_cl1-rootlv on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/vda2 on /boot type ext4 (rw)
/dev/mapper/vg_cl1-homelv on /home type ext4 (rw)
/dev/mapper/vg_cl1-optlv on /opt type ext4 (rw)
/dev/mapper/vg_cl1-usrlv on /usr type ext4 (rw)
/dev/mapper/vg_cl1-varlv on /var type ext4 (rw)
/dev/mapper/kiranvg-kiranlv on /kiran type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
gvfs-fuse-daemon on /root/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev)
nfsd on /proc/fs/nfsd type nfsd (rw)
/dev/vda13 on /mydir type ext4 (rw)
[root@ cl1 ~]#
```

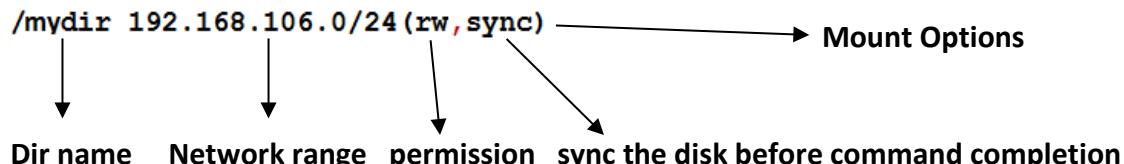
- Access the mount point and add some data in it

```
[root@ cl1 ~]# cd /mydir
[root@ cl1 mydir]# touch file{1..5}
[root@ cl1 mydir]# ls
file1  file2  file3  file4  file5  lost+found
[root@ktcl1 mydir]#
```

### Step3: Export the directory by editing /etc/exports file and using exportfs command

- Edit the /etc/exports file

#vim /etc/exports



/mydir : Name of the directory to be exported  
 192.168.106.0/24 : Range of network where directory can be mounted  
 To give permission to only one node, just give the IP ADDR

|                                            |   |                                                                                  |
|--------------------------------------------|---|----------------------------------------------------------------------------------|
| (rw, sync)                                 | : | Mount options                                                                    |
| <u>The Mount options which can be used</u> |   |                                                                                  |
| rw                                         | : | Sets read/write permissions                                                      |
| ro                                         | : | Sets read-only permissions                                                       |
| sync                                       | : | Specifies that all changes must be written to disk before a Command completes    |
| no_wdelay                                  | : | Forces the writing of changes immediately (useful for logs if Something crashes) |
| root_squash                                | : | Prevent root user's privilege                                                    |

- Now run the exportfs command to export the directory

#exportfs -avr

```
[root@mlinux71 ~]# exportfs -rv
exporting 192.168.106.0/24:/mydir
```

#### Options:

- a      Exports or un-exports all directories
- r      Reexport all directories
- u      Unexports one or more directories
- v      Provides verbose output

**Step4: Start the services and make it permanent.**

- #systemctl start nfs-server.service
- #systemctl enable nfs-server.service

```
[root@mlinux71 ~]# systemctl start nfs-server.service
[root@mlinux71 ~]# systemctl enable nfs-server.service
ln -s '/usr/lib/systemd/system/nfs-server.service' '/etc/systemd/:.
er.target.wants/nfs-server.service'
[root@mlinux71 ~]#
```

**Check the directories which is exported in /var/lib/nfs/etab and /var/lib/nfs/rmtab**

```
[root@mlinux71 ~]# cat /var/lib/nfs/etab
/mydir 192.168.106.0/24(rw,sync,wdelay,hide,nocrossmnt,secure,1
e_check,secure_locks,acl,anonuid=65534,anongid=65534,sec=sys,rw,
[root@mlinux71 ~]# cat /var/lib/nfs/rmtab
[root@mlinux71 ~]#
```

- Note: Add the nfs in firewalld trusted services
- #firewall-cmd --add-service=nfs --permanently
- #firewall-cmd --reload

```
[root@mlinux71 ~]# firewall-cmd --add-service=nfs --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpcv6-client ftp nfs ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```



**Client side configuration for NFS mounting**

**Step1:** Check and Install the NFS package if not installed

**Step2:** Start the NFS services

**Step3:** Check which directory is exported for this machine using showmount command

**Step4:** Make a directory and mount the NFS dir over it.

**Step5:** Add some data to it and check the same is updated on server side.

**Step1:** Check and Install the package for NFS

#rpm -q nfs-utils

```
[root@ cl1 ~]# rpm -q nfs-utils
nfs-utils-1.2.3-7.el6.x86_64
[root@ cl1 ~]#
```

It will be already installed, if it is not installed use yum install nfs-utils\* -y

**Step2:** check and start the NFS services and make it permanent.

- #systemctl start nfs-server.service
- #systemctl enable nfs-server.service

```
[root@mlinux71 ~]# systemctl start nfs-server.service
[root@mlinux71 ~]# systemctl enable nfs-server.service
ln -s '/usr/lib/systemd/system/nfs-server.service' '/etc/systemd:/.
er.target.wants/nfs-server.service'
[root@mlinux71 ~]#
```

**Step3:** Check which directory is exported for this machine using showmount command

- To check the exported directories from server the syntax is  
#showmount -e <server ip address>

```
[root@mlinux72 ~]# showmount -e 192.168.106.81
clnt_create: RPC: Port mapper failure - Unable to receive: errno 113 (No route
o host)
[root@mlinux72 ~]#
```

Note: At first it may show such error, due to firewall blocking some important services on server side. To resolve it login to server and allow following services in firewall.

- #firewall-cmd --add-service=rpc-bind --permanent
- #firewall-cmd --add-service=mountd --permanent
- #firewall-cmd --reload

```
[root@mlinux71 ~]# firewall-cmd --add-service=rpc-bind --permanent
success
[root@mlinux71 ~]# firewall-cmd --add-service=mountd --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpcv6-client ftp mountd nfs rpc-bind ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Now try showmount -e , it would be working fine

```
[root@mlinux72 ~]# showmount -e 192.168.106.81
Export list for 192.168.106.81:
/mydir 192.168.106.0/24
[root@mlinux72 ~]#
```

**Step4: Make a directory and mount NFS over it.**

- #mkdir /nfscl
- #mount -t nfs 192.168.106.81:/mydir /nfscl

```
[root@mlinux72 ~]# mkdir /nfscl
[root@mlinux72 ~]# mount -t nfs 192.168.106.81:/mydir /nfscl
[root@mlinux72 ~]# mount |grep /nfscl
192.168.106.81:/mydir on /nfscl type nfs4 (rw,relatime,vers=4.0
ze=262144,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,:
r=192.168.106.82,local_lock=none,addr=192.168.106.81)
[root@mlinux72 ~]# cd /nfscl
[root@mlinux72 nfscl]# ls
nfs1  nfs2  nfs3  nfs4  nfs5
[root@mlinux72 nfscl]#
```

- To make it permanent mount edit /etc/fstab file as follows

|                                           |        |                  |
|-------------------------------------------|--------|------------------|
| UUID=34ef8764-c31e-40e1-ba46-9d0805aa8e0e | swap   | swap             |
| ts                                        | 0 0    |                  |
| 192.168.106.81:/mydir                     | /nfscl | nfs defaults 0 0 |
| ~                                         |        |                  |

### Auto-mounting the NFS directory

- All the resources of the server is valuable and needs to be available for usage, when we mount a NFS directory over client the network resource gets busy, even when the work is finished the network resource will still be busy as mounting occupy it.
- Autofs automatically mounts file systems for you when they are requested. This has a very handy feature: It's great for handling removable media. Just CD to the right directory, or execute ls or do anything that sends a request to the mount point: and the daemon mounts it. After all, it's the kind of job that's beneath the dignity of a human being First; you need to install the "autofs" package. It should include some appropriate config files. The files you need is /etc/auto.master
- There are two types of Auto-mounting
  - Direct and Indirect Auto-mounting

#### 1. Direct Auto-Mounting:

In direct mounting for each partner server a mount point (dir) needs to be created. For example if there are 10 nfs share to be mounted at client side, there must be 10 directories created and managed manually.

#### 2. Indirect Auto-Mounting:

In this type of mounting for all NFS server shares, only one mother directory needs to be created and for each server a sub-directory will be automatically created and used for mounting.

*Note: Before going for Auto-mounting remove all kind of mounting done previously*

### Steps to configure Indirect auto-mount at client side

#### **Step1:** Log into client side and check whether autofs is install or not, if not install autofs

- Check whether autofs is install or not

```
#rpm -q autofs
```

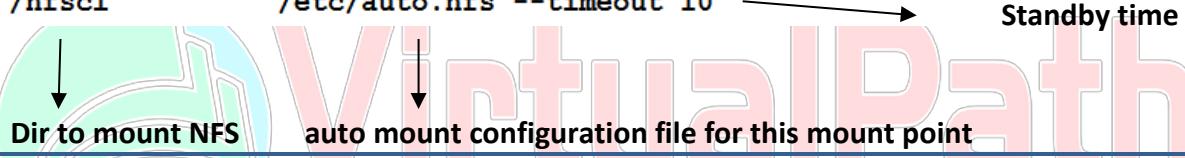
```
[root@ cl1 /]# rpm -q autofs
autofs-5.0.5-31.el6.x86_64
[root@ cl1 /]#
```

- if it is not installed, install it by using yum or rpm  
`#yum install autofs* -y`

#### **Step2:** Edit the /etc/auto.master as follows

```
#vim /etc/auto.master
```

```
# Note that if there are entries for /net or /misc (as
# above) in the included master map any keys that are the
# same will not be seen as the first read key seen takes
# precedence.
#
+auto.master
/nfscl          /etc/auto.nfs --timeout 10
```



Standby time

Dir to mount NFS      auto mount configuration file for this mount point

#### **Step3:** Create /etc/auto.nfs file and /nfscl directory if not created earlier

- `#vim /etc/auto.nfs`

|                             |             |                               |
|-----------------------------|-------------|-------------------------------|
| 81                          | -rw         | 192.168.106.81:/mydir         |
| Name to be used for sub-dir | Permissions | NFS server and directory name |

#### **Step4:** Start/Restart the autofs service and make it permanent

- `#systemctl start/restart autofs`
- `#systemctl enable autofs`

```
[root@mlinux72 ~]# systemctl start autofs.service
[root@mlinux72 ~]# systemctl enable autofs.service
ln -s '/usr/lib/systemd/system/autofs.service' '/etc/systemd/system/
[root@mlinux72 ~]#
```

#### **Step5:** log into the given directory given in /etc/auto.master i.e. /nfscl and check that if NFS is mounted by mount command

```
[root@mlinux72 ~]# cd /nfscl
[root@mlinux72 nfscl]# ls
[root@mlinux72 nfscl]#
```

Note: Still NFS dir will not be mounted on the client side

**Step6:** change the directory to the name given in /etc/auto.nfs i.e. 81 and then auto mounting will be done.

```
#cd 81
```

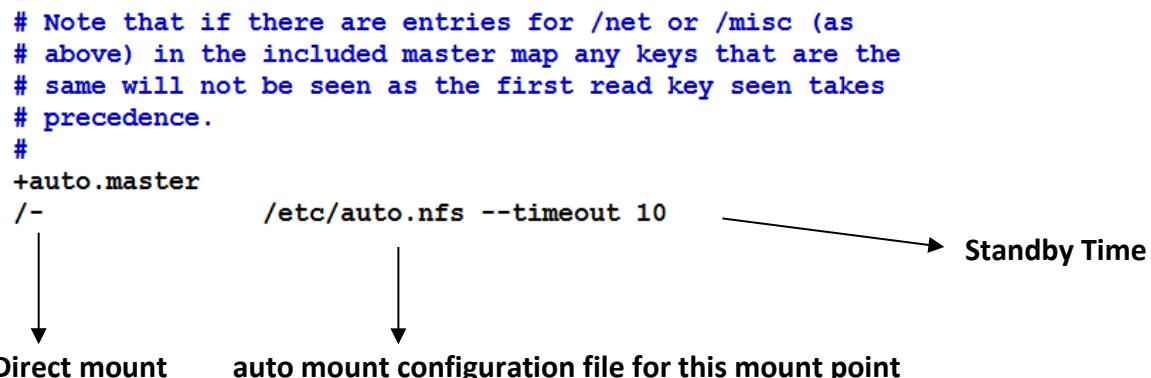
```
#ls
```

```
[root@mlinux72 nfscl]# cd 81
[root@mlinux72 81]# ls
nfs1  nfs2  nfs3  nfs4  nfs5
```

### Steps to configure Direct auto-mount at client side

#### **Step1:** Edit the auto.master file as follows

```
# Note that if there are entries for /net or /misc (as
# above) in the included master map any keys that are the
# same will not be seen as the first read key seen takes
# precedence.
#
+auto.master
/-           /etc/auto.nfs --timeout 10
```



Standby Time

Direct mount      auto mount configuration file for this mount point

#### **Step2:** Edit /etc/auto.nfs file and /nfscl directory if not created earlier

- #vim /etc/auto.nfs

```
/nfscl  -rw    192.168.106.81:/mydir
```



Path of the directory to be used    Permissions    NFS server and directory name

#### **Step3:** Reload the autofs services

- #systemctl reload autofs

```
[root@node1 ~]# systemctl reload autofs
```

#### **Step5:** log into the given directory given in /etc/auto.master i.e. /nfscl and check that if NFS is mounted by mount command

```
[root@mlinux72 ~]# cd /nfscl
[root@mlinux72 nfscl]# ls
nfs1  nfs2  nfs3  nfs4  nfs5
[root@mlinux72 nfscl]#
```

### Steps for removing NFS

**Step1:** Remove all autofs details from all configuration files like /etc/auto.master and /etc/auto.nfs

**Step2:** Delete the entries of the NFS share you want to remove, from /etc/exports on the server and un-export all the directory which was exported earlier using following command

- # exportfs -auv

```
[root@ cl1 ~]# cat /var/lib/nfs/etab
/mydir 192.168.10.0/24(rw,sync,wdelay,hide,nocrossmnt,secure,root_squash,no_all_squash,no_subtree_check,secure_locks,acl,ano
nuid=65534,anongid=65534)
[root@ cl1 ~]# exportfs -auv
[root@ cl1 ~]# cat /var/lib/nfs/etab
[root@ cl1 ~]#
```

*Note: - if you don't have DNS and still want to use hostname instead of IP, update hostname with its ip in /etc/hosts file and then you can use hostname instead of IP*

To check ports and protocol on nfs server from client side, the following commands can be used

#rpcinfo -p <server ip>

```
[root@mlinux4 ~]# rpcinfo -p 192.168.10.30
   program  vers  proto   port  service
    100000    4    tcp    111  portmapper
    100000    3    tcp    111  portmapper
    100000    2    tcp    111  portmapper
    100000    4    udp    111  portmapper
    100000    3    udp    111  portmapper
    100000    2    udp    111  portmapper
    100005    1    udp   20048  mountd
    100005    1    tcp   20048  mountd
    100005    2    udp   20048  mountd
    100024    1    udp   36469  status
    100005    2    tcp   20048  mountd
    100024    1    tcp   43027  status
    100005    3    udp   20048  mountd
    100005    3    tcp   20048  mountd
    100003    3    tcp   2049   nfs
    100003    4    tcp   2049   nfs
    100227    3    tcp   2049   nfs_acl
```

Finally we are done with all the NFS practical. Do hands on practice on it, as it is important in real world

## SAMBA SERVER



- The whole point of networking is to allow computers to easily share information. Sharing information with other Linux boxes, or any UNIX host, is easy—tools such as FTP and NFS are readily available and frequently set up easily “out of the box”. Unfortunately, even the most die-hard Linux fanatic has to admit the operating system most of the PCs in the world are running is one of the various types of Windows. Unless you use your Linux box in a particularly isolated environment, you will almost certainly need to exchange information with machines running Windows. Assuming you're not planning on moving all of your files using floppy disks, the tool you need is Samba.
- Samba is an implementation of a Common Internet File System (CIFS, also known as SMB) protocol server that can be run on almost every variant of Unix in existence. Microsoft clients will use this protocol to access files and printers located on your Unix box just as if it were a native Windows server.
- **Samba** allows **linux** computers to share files and printers across a network connection. By using its SMB protocol, your **linux** box can appear **in** Windows Network Neighborhood or My Network Places just like any other windows machine. You can share files this way, as well as printers. By using **samba** on my home network, for example, my Windows machines have access to a printer directly hooked up to my **Linux** box, and my **Linux** box has access to a printer directly hooked up to one of my Windows machines. **In** addition, everyone can access everyone else's shared files. You can see how **samba** can be very useful if you have a network of both Windows as well as **Linux** machines.

## Profile for SAMBA:

|             |   |                                                                                                                                       |
|-------------|---|---------------------------------------------------------------------------------------------------------------------------------------|
| Usage       | : | used for sharing files and directories in the network Between different platforms, like Linux-windows                                 |
| Package     | : | SAMBA, SAMBA-common, SAMBA-client.                                                                                                    |
| Daemons     | : | smbd, nmbd                                                                                                                            |
| Port no     | : | 137 (net bios –ns{name service}), 138 (net bios–dgm {Datagram}), 139 (net bios-ssn {session service}), 445 (Microsoft –ds {dist sys}) |
| File system | : | CIFS (common internet file system)                                                                                                    |
| Config file | : | /etc/samba/smb.conf                                                                                                                   |
| Sample file | : | /etc/samba/smb.conf.example                                                                                                           |

## Steps to configure SAMBA server

### Step1: Check and Install the SAMBA package, if not installed

#rpm -q samba

```
[root@ cl1 ~]# rpm -q samba
package samba is not installed
[root@ cl1 ~]#
```

- **Install the package using yum**  
#yum install samba\* -y

```
[root@ cl1 ~]# yum install samba* -y
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
Setting up Install Process
Package samba-winbind-clients-3.5.6-86.el6.x86_64 already installed and latest version
Package samba-common-3.5.6-86.el6.x86_64 already installed and latest version
Package samba-client-3.5.6-86.el6.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
--> Package samba.x86_64 0:3.5.6-86.el6 will be installed
--> Package samba-winbind.x86_64 0:3.5.6-86.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version        Repository
=====
Installing:
samba            x86_64   3.5.6-86.el6   RHEL 7
samba-winbind    x86_64   3.5.6-86.el6   RHEL 7

Transaction Summary
=====
Install      2 Package(s)
Installed:
  samba.x86_64 0:3.5.6-86.el6
                           samba-winbind.x86_64 0:3.5.6-86.el6

Complete!
[root@ktcl1 ~]#
```

**Step2:** Make a directory and assign full permission to it, which will be shared

- #mkdir /samba
- #chmod 777 /samba

```
[root@ cl1 ~]# mkdir /samba
[root@ cl1 ~]# chmod 777 /samba
[root@ cl1 ~]#
```

**Step3:** Check the context of the directory and change it according to samba

- #ls -ldZ /samba
- #chcon -t samba\_share\_t /samba

```
[root@ cl1 ~]# ls -ldZ /samba/
drwxrwxrwx. root root unconfined_u:object_r:default_t:s0 /samba/
[root@ cl1 ~]# chcon -t samba_share_t /samba/
[root@ cl1 ~]# ls -ldZ /samba/
drwxrwxrwx. root root unconfined_u:object_r:samba_share_t:s0 /samba/
[root@ cl1 ~]#
```

**Step4:** Create a user or use any existing user who will be allowed to log in as samba user, add that user to samba user

- As we have a existing user “myuser”, let’s just make it samba user

#smbpasswd -a <username>

#smbpasswd -a myuser

Give password twice and wait till it add the user

```
[root@ cl1 ~]# smbpasswd -a myuser
New SMB password:
Retype new SMB password:
Added user myuser.
[root@ cl1 ~]#
```

Note: To delete a user from samba use #smbpasswd -x <user name>

- To check all the samba user use

#pdbeedit -L

```
[root@ cl1 ~]# pdbeedit -L
myuser:515:
[root@ cl1 ~]#
```

**Step5:** Go to the sample configuration file i.e. /etc/samba/smb.conf.example and copy the last paragraph as shown below

- Open the /etc/samba/smb.conf and paste the copied paragraph shown below and edit it.

```
# A publicly accessible directory, but read only, except for people in
# the "staff" group
[public]
comment = Public Stuff
path = /home/samba
public = yes
writable = yes
printable = no
write list = +staff
```

- Once pasted remove ";" mark before it and change it according to following picture

```
#####
[myshare]
comment = Public Stuff
path = /samba
public = no
valid users = myuser
writable = yes
printable = no
hosts allow = 192.168.
```

#### Explanation about the above fields

- [myshare] : Share Name
- Comment = Public Stuff : Comment
- Path = /samba : Share Directory
- Public = no : Public Access (Every user in network)
- Valid user = myuser : Authorized user
- Writable = yes : Write Permission
- Printable = no : Print permission
- Host allow= 192.168. : Network Range or host range

**Note:** Use 192.168.10. To allow only 192.168.10 network, in our case we have allowed any machine in 192.168. Network

#### Step5: Test the samba parameters and restart the service and make it enable after reboot

- To test the parameters us the following command

```
#testparm
```

```
[root@ cl2 /]# testparm
Load smb config files from /etc/samba/smb.conf
rlimit_max: rlimit_max (1024) below minimum Windows limit (16384)
Processing section "[homes]"
Processing section "[printers]"
Processing section "[myshare]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
[

[printers]
comment = All Printers
path = /var/spool/samba
printable = Yes
browseable = No

[myshare]
comment = Public Stuff
path = /samba
valid users = myuser
read only = No
hosts allow = 192.168.

[root@ cl2 /]#
```

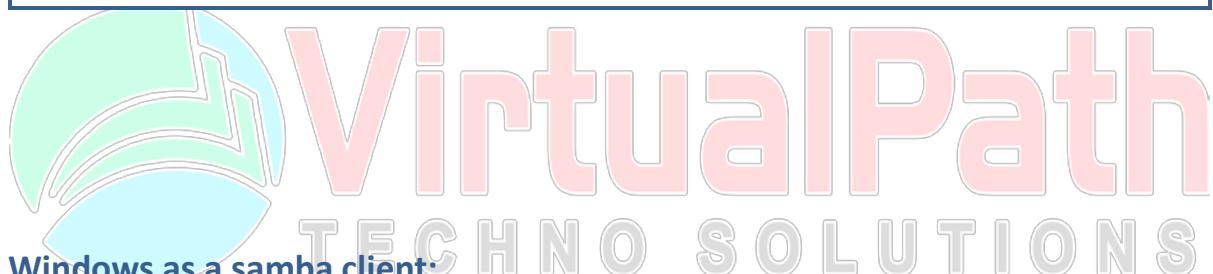
- Start the services and make it permanent by enabling it.

```
#systemctl start smb; systemctl enable smb
#systemctl start nmb; systemctl enable nmb
```

```
[root@mlinux71 ~]# systemctl start smb; systemctl enable smb
[root@mlinux71 ~]# systemctl start nmb; systemctl enable nmb
[root@mlinux71 ~]#
```

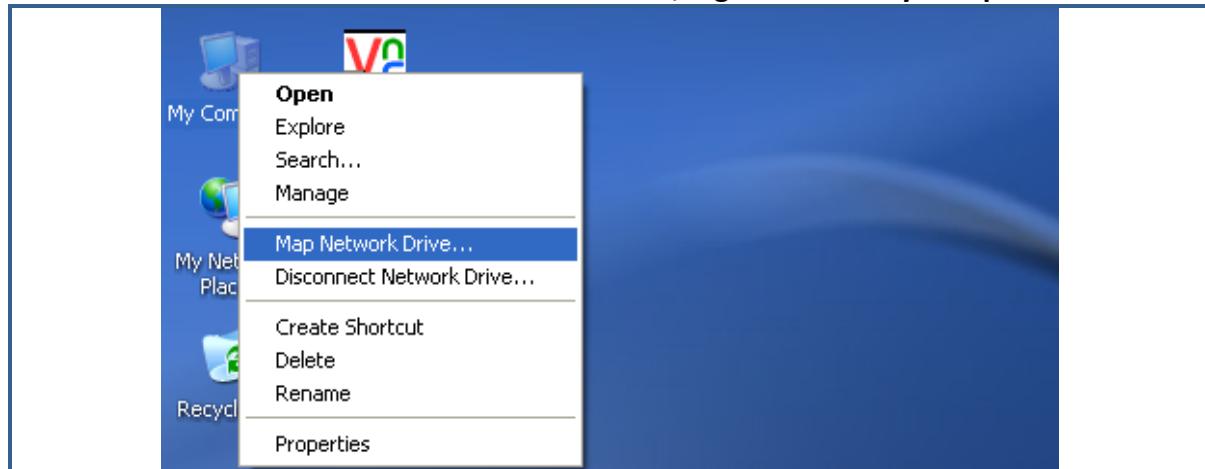
#### Step6: Add samba service in firewall in RHEL7

```
[root@mlinux71 ~]# firewall-cmd --add-service=samba --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default)
interfaces:
sources:
services: dhcpcv6-client ftp mountd nfs rpc-bind samba ssh
ports:
masquerade: no
forward-ports:
icmp-blocks:
rich rules:
```

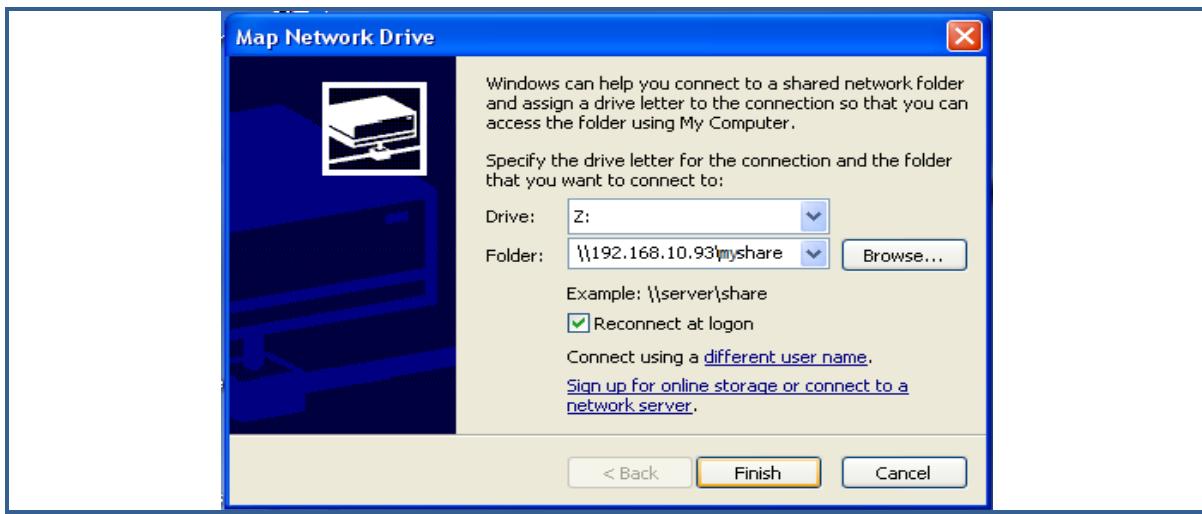


#### Windows as a samba client:

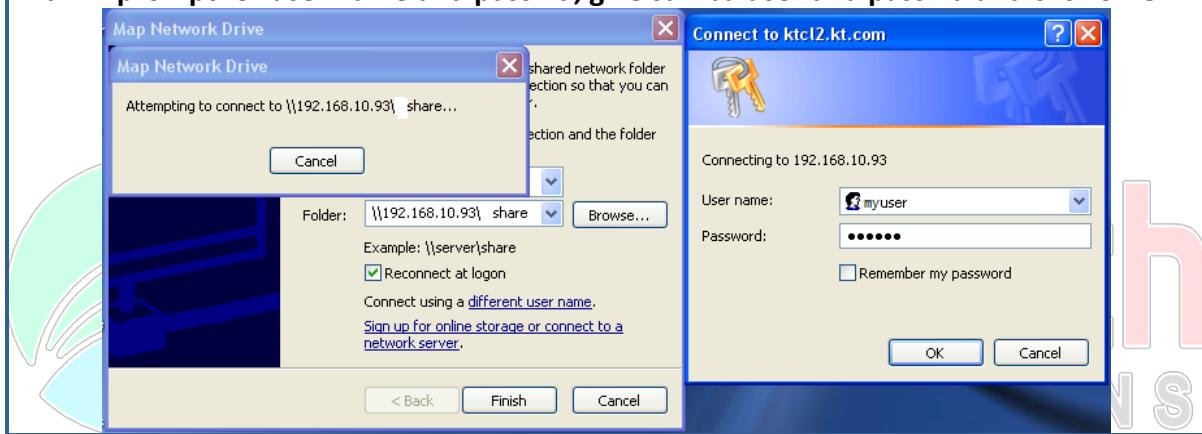
To connect from windows to the samba server, Right click on My Computer icon select



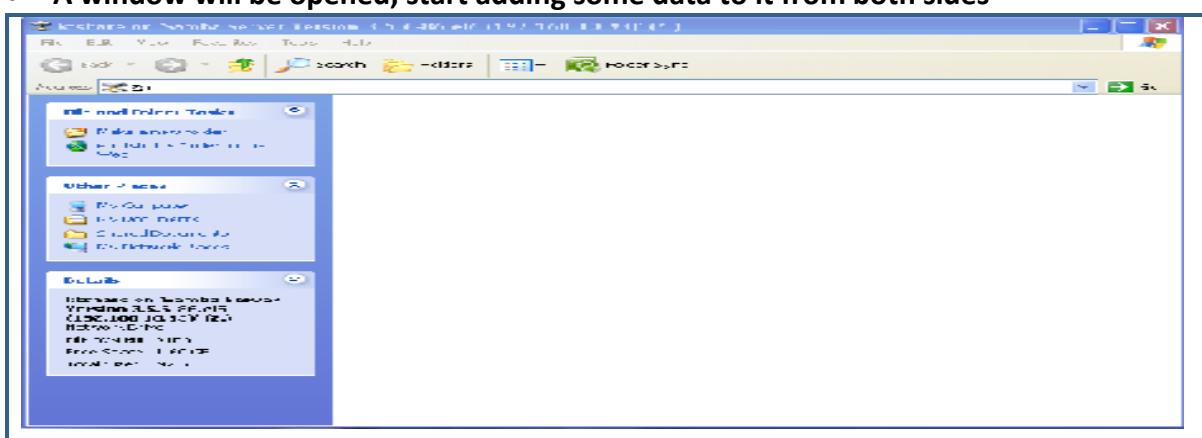
- Give the address of samba server as “\\192.168.10.93\ktshare(sharename), press on finish to continue.

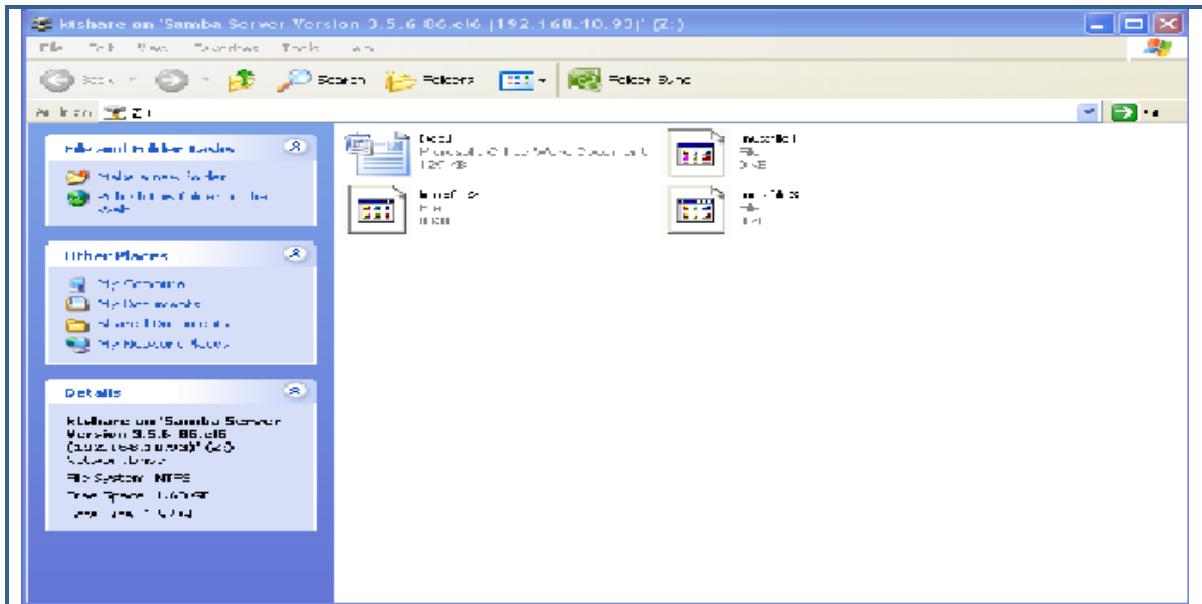


It will prompt for user name and passwd, give samba user and passwd and click on OK



- A window will be opened, start adding some data to it from both sides





## Linux as a SAMBA Client

- Log into Linux machine and check how many samba servers are there in your network  
**#findsmb**

```
[root@ cl5 ~]# findsmb

*=DMB
+=LMB
IP ADDR      NETBIOS NAME      WORKGROUP/OS/VERSION
-----
192.168.10.83 CL3          +[CL] [Windows Server 2003 R2 3790 Service Pack 2]
[Windows Server 2003 R2 5.2]
192.168.10.93 CL2          [MYGROUP] [Unix] [Samba 3.5.6-86.el6]
192.168.10.98 LINUX        +[WORKGROUP] [Windows Server 2003 R2 3790 Service]
```

- Check the share name of that samba server by using following command

**#smbclient -L //192.168.10.93**

when prompted for passwd just press enter without giving any passwd

```
[root@ cl5 ~]# smbclient -L //192.168.10.93
Enter root's password:
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-86.el6]

      Sharename      Type      Comment
      -----      ----      -----
          IPC$        IPC       IPC Service (Samba Server Version 3.5.6-86.el6
)
          myshare      Disk      Public Stuff
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-86.el6]

      Server      Comment
      -----      -----
          CL2        Samba Server Version 3.5.6-86.el6
          LINUX      Samba Server Version 3.5.4-68.el6

      Workgroup      Master
      -----      -----
          CL         CL3
          TS         ADS
          MYGROUP    LINUX
          WORKGROUP  LINUX
```

- To connect to the samba server use the following syntax

```
#smbclient //<server IP>/<share name> -U <User name>
```

```
#smbclient //192.168.10.93/myshare -U myuser
```

```
[root@ cl5 ~]# smbclient //192.168.10.93/myshare -U myuser
Enter myuser's password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-86.el6]
smb: \> ls
.
..
linuxfile3
Doc1.docx
linuxfile1
linuxfile2

D          0  Wed Nov  9 20:40:18 2011
DR         0  Wed Nov  9 20:00:43 2011
          0  Wed Nov  9 20:40:18 2011
A   131602 Sat Oct  1 15:26:29 2011
          0  Wed Nov  9 20:40:18 2011
          0  Wed Nov  9 20:40:18 2011

62994 blocks of size 32768. 52515 blocks available

smb: \> █
```

- To mount the SAMBA directory on remote Linux client
- Make sure *cifs-utils* package is installed, if not install it first before using *mount* command.
- The syntax for mounting samba share is as follows

```
#mount -t <type of fs> //<server IP address>/<share name> /<mount point> -o user=<user name>.
```

```
#mount -t cifs //192.168.10.93/myshare /mnt -o user=myuser
```

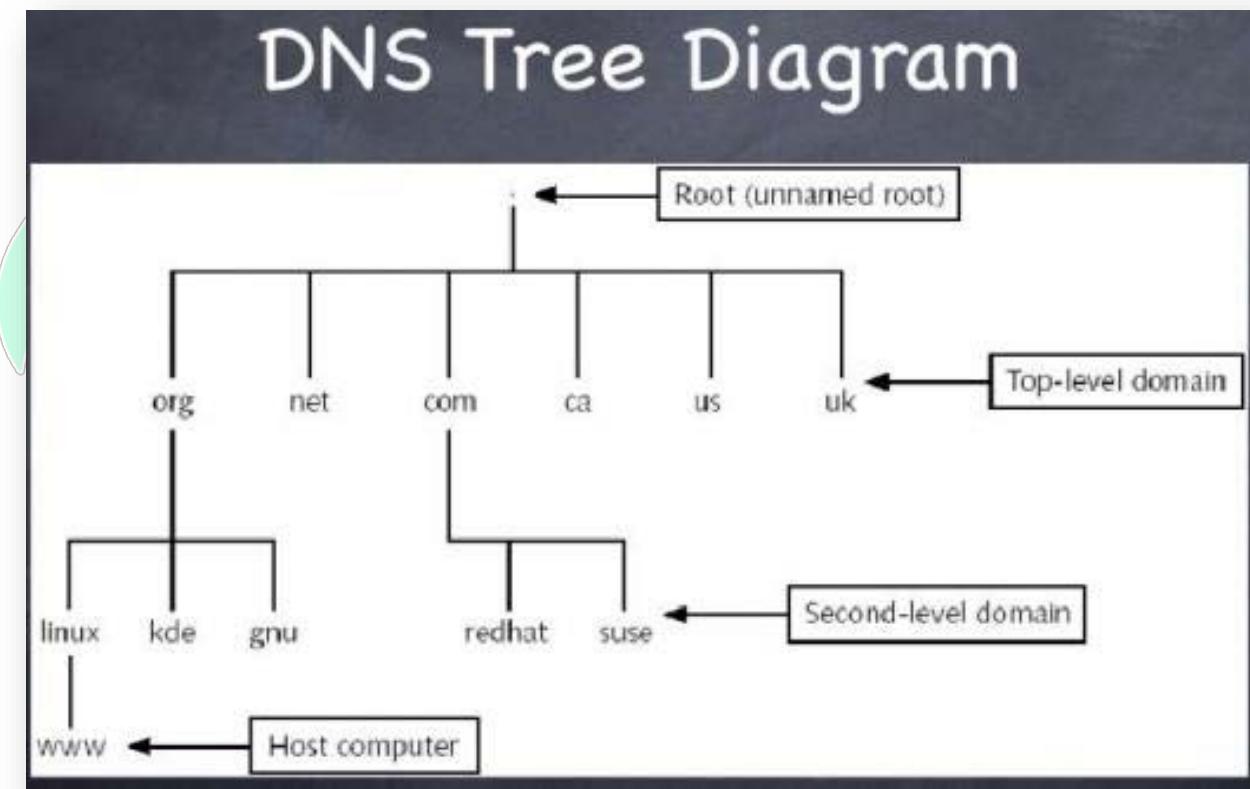
```
[root@ cl5 ~]# mount -t cifs //192.168.10.93/myshare /mnt -o user=myuser
Password:
[root@ cl5 ~]# mount
/dev/mapper/rootvg_ cl5-LogVol00 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda2 on /boot type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol01 on /home type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol04 on /opt type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol05 on /tmp type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol02 on /usr type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol03 on /var type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
gvfs-fuse-daemon on /root/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev)
//192.168.10.93/myshare/ on /mnt type cifs (rw,mand)
[root@ cl5 ~]# █
```

*Note: To learn how to make samba permanent mount and auto-mount visit my website <http://musab.in/2014/02/making-samba-permanent-and-auto-mount-in-linux/>*

## DNS (Domain Name System) SERVER

### Domain Name System

The Domain Name System (DNS) is the crucial glue that keeps computer networks in harmony by converting human-friendly hostnames to the numerical IP addresses computers require to communicate with each other. DNS is one of the largest and most important distributed databases the world depends on by serving billions of DNS requests daily for public IP addresses. Most public DNS servers today are run by larger ISPs and commercial companies but private DNS servers can also be useful for private home networks.



Like the telephone system, every device attached to the Internet has a unique number, its IP address. Also like the telephone system there is a directory services to help you find those numbers called DNS.

If you have someone's name and address you can call a directory services, give them the details you know and they will (usually) give you the telephone number to call them. Likewise, if you know a server's host name (maybe <http://www.google.co.in/>) you can give that name to a DNS server and it will give you the IP address of that server.

### **The format of a domain name**

Like a physical address, Internet domain names are hierarchical (only a little stricter), so while your address might look like:

|                    |                     |
|--------------------|---------------------|
| <b>House name:</b> | <b>TM Residency</b> |
|                    | <b>Ameerpet</b>     |
| <b>Town:</b>       | <b>Hyderabad</b>    |
| <b>County:</b>     | <b>Telangana</b>    |
| <b>Country:</b>    | <b>India</b>        |

An Internet domain name looks like:

|                            |               |
|----------------------------|---------------|
| <b>Host name</b>           | <b>www</b>    |
| <b>Domain</b>              | <b>google</b> |
| <b>Second level domain</b> | <b>co</b>     |
| <b>Top-level domain</b>    | <b>In</b>     |

**A database is made up of records and the DNS is a database. Therefore, common resource record types in the DNS database are:**

- **A** - Host's IP address. Address record allowing a computer name to be translated into an IP address. Each computer must have this record for its IP address to be located. These names are not assigned for clients that have dynamically assigned IP addresses, but are a must for locating servers with static IP addresses.
- **PTR** - Host's domain name, host identified by its IP address
- **CNAME** - Host's canonical name allows additional names or aliases to be used to locate a computer.
- **MX** - Host's or domain's mail exchanger.
- **NS** - Host's or domain's name server(s).
- **SOA** - Indicates authority for the domain (Start of Authority)
- **TXT** - Generic text record
- **SRV** - Service location record
- **RP** - Responsible person
- **HINFO** - Host information record with CPU type and operating system

**The package which is used in Linux for performing DNS activity is BIND (Berkeley Internet Name Domain)**

## Profile for DNS Server

|                    |   |                                            |
|--------------------|---|--------------------------------------------|
| Usage              | : | To Resolve IP into hostname and vice-versa |
| Package            | : | bind, caching-name                         |
| Daemon             | : | named                                      |
| Port               | : | 53                                         |
| Configuration File | : | /etc/named.conf, /etc/named.rfc1912.zones  |
| Document root      | : | /var/named/                                |

## Step by Step configuration of DNS server

### Step1: Check and Install the package for DNS

- The package which is to be installed for DNS is bind and caching  
#rpm -q bind

```
[root@ adm ~]# rpm -q bind
package bind is not installed
[root@ adm ~]#
```

- To install the package use yum or rpm command

```
[root@ adm ~]# yum install bind* caching* -y
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
Setting up Install Process
Package 32:bind-utils-9.7.3-2.el6.x86_64 already installed and latest version
Package 32:bind-libs-9.7.3-2.el6.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
--> Package bind.x86_64 32:9.7.3-2.el6 will be installed
--> Package bind-chroot.x86_64 32:9.7.3-2.el6 will be installed
--> Package bind-dyndb-ldap.x86_64 0:0.2.0-1.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch    Version        Repository      Size
=====
Installing:
bind              x86_64  32:9.7.3-2.el6   RHEL6          3.9 M
bind-chroot       x86_64  32:9.7.3-2.el6   RHEL6          67 k
bind-dyndb-ldap   x86_64  0.2.0-1.el6    RHEL6          49 k

Installed:
bind.x86_64 32:9.7.3-2.el6                  bind-chroot.x86_64 32:9.7.3-2.el6
bind-dyndb-ldap.x86_64 0:0.2.0-1.el6

Complete!
```

**Step2:** Update the /etc/hosts file with the server's ip address, and change the hostname with fully qualified domain name.

- Change the hostname by adding fully qualified domain name

```
#hostnamectl set-hostname mlinux3.vpts.com
```

```
[root@mlinux3 ~]# hostnamectl set-hostname mlinux3.vpts.com
[root@mlinux3 ~]# hostname
mlinux3.vpts.com
```

Note:- change the hostname on all clients by making it FQDN

- Update /etc/hosts on DNS server with hostname and IP address

```
#vim /etc/hosts
```

```
[root@mlinux3 named]# more /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.10.30  mlinux3.vpts.com
[root@mlinux3 named]#
```

**Step3:** Edit the configuration file “/etc/named.conf

- Edit the /etc/named.conf file with our name server's IP address and network range for clients.

```
#vim /etc/named.conf
```

```
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    directory      "/var/named";
    dump-file      "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query     { localhost; };
    recursion yes;
```

Note: Need to add our systems details in highlighted lines

```
options {
    listen-on port 53 { 127.0.0.1; 192.168.10.30; };
    listen-on-v6 port 53 { ::1; };
    directory      "/var/named";
    dump-file      "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    secroots-file   "/var/named/data/named.secroots";
    recursing-file  "/var/named/data/named.recurse";
    allow-query     { localhost; 192.168.10.0/24; };
```

Where 192.168.10.31 is our Name server's IP Address

And 192.168.10.0/24 is the network's range from where clients can query the Name server. If you want it be open for all network, your “any” instead of network addr.

**Step4: Edit the other zone configuration file i.e. “/etc/named.rfc1912.zones”**

- To add the details of the zones i.e. forward lookup zone and reverse lookup zone we need to edit the /etc/named.rfc1912.zones file as shown below  
**#vim /etc/named.rfc1912.zones**

Copy the following 11 lines and paste it at the end of the file

```
zone "localhost.localdomain" IN {
    type master;
    file "named.localhost";
    allow-update { none; };
};

zone "localhost" IN {
    type master;
    file "named.localhost";
    allow-update { none; };
};
```

Once pasted, edit the fields as follows

```
#####
zone "vpts.com" IN {
    type master;
    file "my.flz";
    allow-update { none; };
};

zone "10.168.192.in-addr.arpa" IN {
    type master;
    file "my.rlz";
    allow-update { none; };
};
```

Where “vpts.com” is the name of the domain

And “10.168.192.in-addr.arpa” is the reverse order of our domain network.

“my.flz” is the name of the forward lookup zone file and...

“my.rlz” is the name of the reverse lookup zone file.

Note: extensions like flz or rlx are not required it is only used here to demonstrate forward and reverse zones



**Step5: Navigate to /var/named/ directory and create a forward and reverse zone files.**

- Navigate to /var/named/ directory and copy the named.localhost file with its permissions as my.flz and edit it.

**#cd /var/named**

**#cp -p named.localhost my.flz**

```
[root@mlinux1 ~]# cd /var/named/
[root@mlinux1 named]# ls
chroot  dynamic      named.ca      named.localhost  slaves
data     dyndb-ldap   named.empty   named.loopback
[root@mlinux1 named]# cp -p named.localhost my.flz
[root@mlinux1 named]# vim my.flz
```

- Edit my.flz file as follows

```
$TTL 1D
@ IN SOA mlinux3.vpts.com. root.vpts.com. (
    202004101 ; serial
    1D         ; refresh
    1H         ; retry
    1W         ; expire
    3H )       ; minimum

NS      mlinux3.vpts.com.
mlinux3 A   192.168.10.30
mlinux4 A   192.168.10.40
mlinux5 A   192.168.10.50
mlinux6 A   192.168.10.60
```

- Copy again named.localhost, this time as my.rlz or copy my.flz to my.rlz to avoid re-typing common entries in both files, and edit it as shown below.

```
#cp -p my.flz my.rlz
#vim my.rlz
```

```
[root@mlinux1 named]# cp -p my.flz my.rlz
[root@mlinux1 named]# vim my.rlz
```

```
$TTL 1D
@ IN SOA mlinux3.vpts.com. root.vpts.com. (
    202004101 ; serial
    1D         ; refresh
    1H         ; retry
    1W         ; expire
    3H )       ; minimum

NS      mlinux3.vpts.com.
30     PTR     mlinux3.vpts.com.
40     PTR     mlinux4.vpts.com.
50     PTR     mlinux5.vpts.com.
60     PTR     mlinux6.vpts.com.
```

#### Step6: check whether the zone files are consistent or not

- To check the consistency of zone files the command is  
`#named-chkzone <domain name> zone file`  
`#named-chkzone my.com my.flz (if you are not in named dir give absolute path)`

```
[root@mlinux3 named]# named-checkzone vpts.com my.flz
zone vpts.com/IN: loaded serial 202004101
OK
```

```
#named-chkzone network addr my.rlz
```

```
[root@mlinux3 named]# named-checkzone 192.168.10. my.rlz
zone 192.168.10/IN: loaded serial 202004101
OK
```

### Step7: Restart the appropriate services

- Start the named service and make it permanent  

```
#systemctl start named; systemctl enable named
```

```
[root@mlinux1 ~]# systemctl start named; systemctl enable named
ln -s '/usr/lib/systemd/system/named.service' '/etc/systemd/system/multi-user.target.wants/named.service'
```

### Step8: Add DNS in firewall trust services to avoid blockage by firewall

```
#firewall-cmd --add-service=dns --permanent
```

```
#firewall-cmd --reload
```

```
#firewall-cmd --list-all
```

```
[root@mlinux1 ~]# firewall-cmd --add-service=dns --permanent
success
[root@mlinux1 ~]# firewall-cmd --reload
success
[root@mlinux1 ~]# firewall-cmd --list-all
public (default)
  interfaces:
  sources:
  services: dhcpcv6-client dns ftp mountd nfs rpc-bind samba ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

### Step9: Add the address of DNS server in /etc/resolv.conf on server side

- Edit the /etc/resolv.conf and add the IP of DNS server

```
#vim /etc/resolv.conf
```

```
[root@mlinux3 named]# cat /etc/resolv.conf
# Generated by NetworkManager
search vpts.com
nameserver 192.168.10.30
```

Okay, now we've done with DNS server configuration, check whether it is resolving IP to hostname and hostname to IP using various commands.

- Using dig command to check the DNS resolution
- Check with giving hostname of server  

```
#dig <FQDN> of server/#dig mlinux3.vpts.com
```

```
[root@mlinux3 named]# dig mlinux3.vpts.com

; <>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <>> mlinux3.vpts.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 8378
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8ada506a60227eef31a0fd405e9080f22b2602dff3b0132d (good)
; QUESTION SECTION:
;mlinux3.vpts.com.      IN      A

;; ANSWER SECTION:
mlinux3.vpts.com.    86400   IN      A       192.168.10.30

;; AUTHORITY SECTION:
vpts.com.            86400   IN      NS     mlinux3.vpts.com.

;; Query time: 0 msec
;; SERVER: 192.168.10.30#53(192.168.10.30)
;; WHEN: Fri Apr 10 19:51:38 IST 2020
;; MSG SIZE rcvd: 103
```

- Check with giving IP of hostname

```
#dig -x 192.168.10.30
```

```
[root@mlinux3 named]# dig -x 192.168.10.30

; <>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <>> -x 192.168.10.30
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23967
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: f487e96a25f94e129ae63cf75e9085a37420441a41db0fd3 (good)
; QUESTION SECTION:
;30.10.168.192.in-addr.arpa. IN PTR

; ANSWER SECTION:
30.10.168.192.in-addr.arpa. 86400 IN PTR mlinux3.vpts.com.

; AUTHORITY SECTION:
10.168.192.in-addr.arpa. 86400 IN NS mlinux3.vpts.com.

; ADDITIONAL SECTION:
mlinux3.vpts.com. 86400 IN A 192.168.10.30

; Query time: 0 msec
; SERVER: 192.168.10.30#53(192.168.10.30)
; WHEN: Fri Apr 10 20:11:39 IST 2020
; MSG SIZE rcvd: 143
```

- Check the same with client's IP and Host name

```
#dig mlinux4.vpts.com
```

```
[root@mlinux3 named]# dig mlinux4.vpts.com

; <>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <>> mlinux4.vpts.com
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49732
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: d887fc2d28536b466285585c5e9082bdb4abe4654eafa0ab (good)
; QUESTION SECTION:
;mlinux4.vpts.com. IN A

; ANSWER SECTION:
mlinux4.vpts.com. 86400 IN A 192.168.10.40

; AUTHORITY SECTION:
vpts.com. 86400 IN NS mlinux3.vpts.com.

; ADDITIONAL SECTION:
mlinux3.vpts.com. 86400 IN A 192.168.10.30

; Query time: 0 msec
; SERVER: 192.168.10.30#53(192.168.10.30)
; WHEN: Fri Apr 10 19:59:17 IST 2020
; MSG SIZE rcvd: 127
```

- With IP address:

```
#dig -x 192.168.10.40
```

```
[root@mlinux3 named]# dig -x 192.168.10.40

; <>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <>> -x 192.168.10.40
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38188
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: 7716f92f999692f7519a97425e9085f4d8884b8797d2debf (good)
; QUESTION SECTION:
;40.10.168.192.in-addr.arpa. IN PTR

; ANSWER SECTION:
40.10.168.192.in-addr.arpa. 86400 IN PTR mlinux4.vpts.com.

; AUTHORITY SECTION:
10.168.192.in-addr.arpa. 86400 IN NS mlinux3.vpts.com.

; ADDITIONAL SECTION:
mlinux3.vpts.com. 86400 IN A 192.168.10.30

; Query time: 0 msec
; SERVER: 192.168.10.30#53(192.168.10.30)
; WHEN: Fri Apr 10 20:13:00 IST 2020
; MSG SIZE rcvd: 151
```

### Using ping to test the resolution

- Try pinging with hostname both server and client

```
#ping -c2 mlinux4
```

```
[root@mlinux3 named]# ping -c2 mlinux4
PING mlinux4.vpts.com (192.168.10.40) 56(84) bytes of data.
64 bytes from mlinux4.vpts.com (192.168.10.40): icmp_seq=1 ttl=64 time=0.449 ms
64 bytes from mlinux4.vpts.com (192.168.10.40): icmp_seq=2 ttl=64 time=1.20 ms

--- mlinux4.vpts.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 41ms
rtt min/avg/max/mdev = 0.449/0.822/1.195/0.373 ms
```

### Using host command to check resolution

- Checking the DNS resolution with host command for both server as well as clients

```
#host <hostname>
#host mlinux3
#host mlinux4
```

```
[root@mlinux3 named]# host mlinux3
mlinux3.vpts.com has address 192.168.10.30
[root@mlinux3 named]# host mlinux4
mlinux4.vpts.com has address 192.168.10.40
```

- Using host command with IP address of server as well as client

```
#host 192.168.10.3
#host 192.168.10.4 (or) 5, 6 any client
```

```
[root@mlinux3 named]# host 192.168.10.30
30.10.168.192.in-addr.arpa domain name pointer mlinux3.vpts.com. S
[root@mlinux3 named]# host 192.168.10.40
40.10.168.192.in-addr.arpa domain name pointer mlinux4.vpts.com.
```

### Using nslookup command to check the DNS resolution

- Use nslookup command with server and clients hostname and check it

```
#nslookup mlinux2
#nslookup mlinux3
```

```
[root@mlinux3 named]# nslookup mlinux3
Server:          192.168.10.30
Address:         192.168.10.30#53

Name:   mlinux3.vpts.com
Address: 192.168.10.30
```

```
[root@mlinux3 named]# nslookup mlinux4
Server:          192.168.10.30
Address:         192.168.10.30#53

Name:   mlinux4.vpts.com
Address: 192.168.10.40
```

- Check the same thing with IP addresses

```
#nslookup 192.168.10.30
#nslookup 192.168.10.40
```

```
[root@mlinux3 named]# nslookup 192.168.10.30
30.10.168.192.in-addr.arpa      name = mlinux3.vpts.com.
```

```
[root@mlinux3 named]# nslookup 192.168.10.40
40.10.168.192.in-addr.arpa      name = mlinux4.vpts.com.
```

### Client side configuration for DNS

- Log into any client machine and add the DNS server's information in /etc/resolv.conf file.

```
#vim /etc/resolv.conf
```

```
[root@mlinux3 named]# cat /etc/resolv.conf
# Generated by NetworkManager
search vpts.com
nameserver 192.168.10.30
```

- Now check with any of the options used previously like dig, ping, host or nslookup for dns resolution

```
[root@mlinux4 ~]# nslookup mlinux3
Server:          192.168.10.30
Address:         192.168.10.30#53
```

```
Name:   mlinux3.vpts.com
Address: 192.168.10.30
```

```
[root@mlinux4 ~]# nslookup mlinux4
Server:          192.168.10.30
Address:         192.168.10.30#53
```

```
Name:   mlinux4.vpts.com
Address: 192.168.10.40
```

```
[root@mlinux4 ~]# nslookup 192.168.10.30
30.10.168.192.in-addr.arpa      name = mlinux3.vpts.com.
```

```
[root@mlinux4 ~]# nslookup 192.168.10.40
40.10.168.192.in-addr.arpa      name = mlinux4.vpts.com.
```

**Do the same for every client and check it with various commands on every client  
Also make sure that hostname should Fully Qualified Domain Name.**

## MAIL SERVER

Electronic mail is one of the best way to communicate for computer users anywhere in the world. If I wanted to write an email message to my friend who is sitting somewhere in the world, I simply open up my outlook-click on compose-type my friends email address in the “to box-mention” the subject-draft the message-attach files (if needed)-click on send. That's it. This is what I do to send an email to my friends. Not only me, all the computer users will do the exact same thing. But for most of the time i didn't know how the mail flow takes place. How the transfer takes place and how will it reach the recipient and the intermediate process and so on....

There are a few new keywords we need to look into.....

**Mail User Agent:** MUA is the email client which we use to create-draft-send emails. Generally Microsoft Outlook, Thunderbird, Kmail and so on..... are examples of MUA's

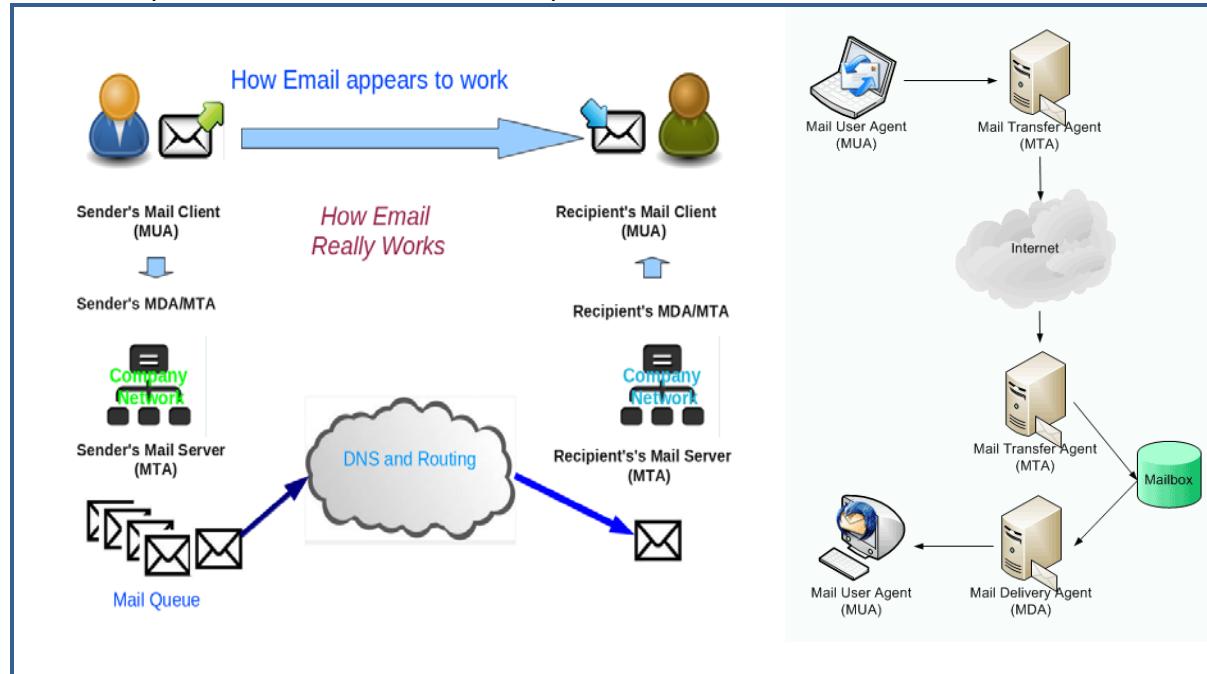
**Mail Transfer Agent:** Message, Mail transfers between sender(s) and recipient(s) will take place between the MTA's. Exchange, Qmail, Sendmail, PostFix and so on.... are example of MTA's

**Mail Delivery Agent:** It is an agent which is responsible for delivery of mail to the devices like laptop, desktop, mobiles and tabs etc. Imtp, pop3, imap4 etc., are the examples of MDA

### **SMTP**

**Simple Mail Transfer Protocol** will transfer the mails between the MTA's

Let's take a deeper look into this with a small example. The below picture will depict how mail flow takes place between sender and recipient.



From the above picture, when the sender clicks on Send in his MUA, the mail will be transferred to the MTA of the sender which exists in the Mail server of the Sender. The MTA of the sender will check for the recipients address (MX records-Mail Exchange Records) and if it finds the recipients address then the mail will be flowed from Senders MTA to Recipient MTA using the SMTP via TCP Port 25. Once the Recipients MTA receives the email, it will be transferred to MUA of recipient. Once the Recipient Clicks on the Send/Receive button then the email will be once click away from him residing in his inbox.

In addition to the above processes there is another agent called as MDA-Mail delivery agent. MDA will receive the email from the MTA and will deliver it to the recipients MUA.

### **Profile for MAIL server**

|                           |          |                                                 |
|---------------------------|----------|-------------------------------------------------|
| <b>Usage</b>              | <b>:</b> | To send and receive emails                      |
| <b>Package</b>            | <b>:</b> | Postfix, Dovecot                                |
| <b>Configuration file</b> | <b>:</b> | /etc/postfix/main.cf, /etc/dovecot/dovecot.conf |
| <b>Port no</b>            | <b>:</b> | 25 – SMTP (Simple Mail Transfer Protocol)       |
| <b>Daemon</b>             | <b>:</b> | postfix, dovecot                                |

### **Lab Work:**

**Pre-requisite:** DNS should be configured and server as well as client should be participant of it

#### **Step 1: Check the hostname of the system**

#hostname

```
[root@myrhel73 ~]# hostname
myrhel73.my.com
```

#### **Step 2: Install the Packages**

```
[root@myrhel73 ~]# yum install postfix* -y
[root@myrhel73 ~]# yum install dovecot* -y
```

**Note:** Most probably postfix will be pre-install, so only dovecot is needed to be installed

#### **Step 3: Search and Edit the configuration file /etc/postfix/main.cf as shown:**

#vim /etc/postfix/main.cf with following line numbers

```
94 myhostname = mlinux3.vpts.com
102 mydomain = vpts.com
118 myorigin = $mydomain
132 inet_interfaces = all
133 #inet_interfaces = $myhostname
134 #inet_interfaces = $myhostname, localhost
135 #inet_interfaces = localhost
183 mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
283 mynetworks = 192.168.10.0/24, 127.0.0.0/8
```

#### **Step 4: Edit the configuration file /etc/dovecot/dovecot.conf as shown**

#vim /etc/dovecot/dovecot.conf ; line no. 24

```
24 protocols = imap pop3 lmtp
```

### Step 5: Start/Restart the services and make them permanent

```
#systemctl restart postfix; systemctl enable postfix
#systemctl start dovecot; systemctl enable dovecot
```

```
[root@myrhel73 ~]# systemctl restart postfix.service; systemctl enable postfix.service
[root@myrhel73 ~]# systemctl start dovecot; systemctl enable dovecot
```

### Step 6: Allow smtp in firewall as below

```
[root@myrhel73 ~]# firewall-cmd --add-service=smtp --permanent
success
[root@myrhel73 ~]# firewall-cmd --reload
success
```

### Step 7: Add the following entries with the MX record in DNS

```
#vim /var/named/my.flz (forward zone)
```

```
[root@mlinux3 ~]# cat /var/named/my.flz
$TTL 1D
@ IN SOA mlinux3.vpts.com. root.vpts.com. (
    202004101 ; serial
    1D         ; refresh
    1H         ; retry
    1W         ; expire
    3H )       ; minimum
NS      mlinux3.vpts.com.
mlinux2 A   192.168.10.20
mlinux3 A   192.168.10.30
mail    CNAME  mlinux3
mlinux4 A   192.168.10.40
mlinux5 A   192.168.10.50
mlinux6 A   192.168.10.60
vpts.com. A  192.168.10.30
vpts.com. MX 10 mlinux3
[root@mlinux3 ~]#
```

### Step 8: Reload the named service

```
[root@myrhel7 ~]# systemctl reload named
```

## Client side configuration for RHEL machine

### Step1: Make the following change in /etc/postfix/main.cf

```
132  inet_interfaces = all
133  #inet_interfaces = $myhostname
134  #inet_interfaces = $myhostname, localhost
135  #inet_interfaces = localhost
338  relayhost = [mlinux3.vpts.com]
```

Note: Line no. varies from rhel6, 7 and 8 versions.

### Step2: Restart the postfix service

```
[root@myrhel72 ~]# systemctl restart postfix
```

### Step 3: Allow smtp in firewall as below

```
[root@myrhel73 ~]# firewall-cmd --add-service=smtp --permanent
success
[root@myrhel73 ~]# firewall-cmd --reload
success
```

### Send the mail from one machine to other as following:

```
#mail -s (subject) user@hostname
#mail -s TESTMAIL root@mlinux4.com
```

```
[root@m... 3 ~]# mail -s TESTMAIL root@mlinux4.vpts.com
HI THERE,
THIS IS A TEST MAIL
EOT ctrl+d to send the mail
[root@myrhel73 ~]#
```

### Check the mail que, to see if the mail has flown out or not

```
#mailq
```

```
[root@myrhel73 ~]# mailq
Mail queue is empty
```

Note: If mailq is empty, that means the mail has been sent

### Check the mailbox on destination server, whether mail is received

```
#mail
```

```
[root@myrhel72 ~]# mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N 1 root Wed Oct 19 22:35 22/793 "TESTMAIL"
&
```

Read the mail number 1 and reply

```
& 1
From: root@myrhel73.my.com (root)
Status: R

HI THERE,
THIS IS A TEST MAIL
```

Reply it using “r” key

```
& r
To: root@myrhel72.my.com root@myrhel73.my.com
Subject: Re: TESTMAIL

root@myrhel73.my.com (root) wrote:

> HI THERE,
> THIS IS A TEST MAIL
Rec'd it successfully
EOT
```

Check back on first Machine whether it received a mail back or not

#mail

```
[root@myrhel73 ~]#
You have new mail in /var/spool/mail/root
[root@myrhel73 ~]# mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/spool/mail/root": 2 messages 1 new
  1 Mail Delivery System  Wed Oct 19 22:29  75/2394 "Undelivered Mail Returned to Sender"
>N  2 root           Wed Oct 19 22:38  27/996 "Re: TESTMAIL"
& [2]
Message 2:
From root@myrhel72.my.com  Wed Oct 19 22:38:14 2016

Return-Path: <root@myrhel72.my.com>
X-Original-To: root@myrhel73.my.com
Delivered-To: root@myrhel73.my.com
Date: Wed, 19 Oct 2016 22:38:15 +0530
To: root@myrhel73.my.com, root@myrhel72.my.com
Subject: Re: TESTMAIL
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
From: root@myrhel72.my.com (root)
Status: R

root@myrhel73.my.com (root) wrote:

> HI THERE,
> THIS IS A TEST MAIL
Rec'd it successfully

& [ ]
```

Also test mails from between two different clients to test Relay host functionality.

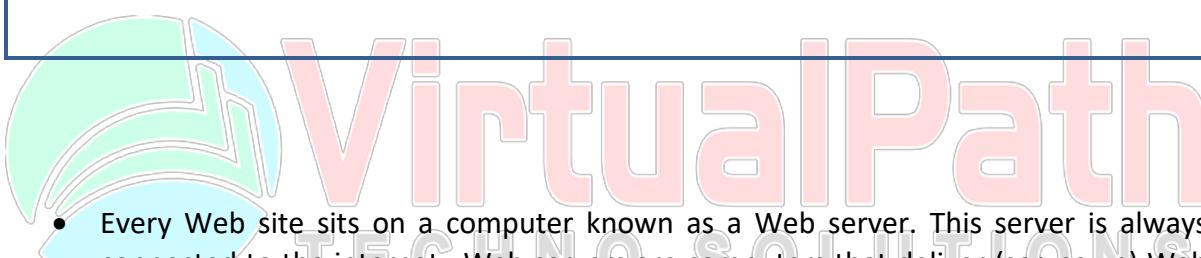
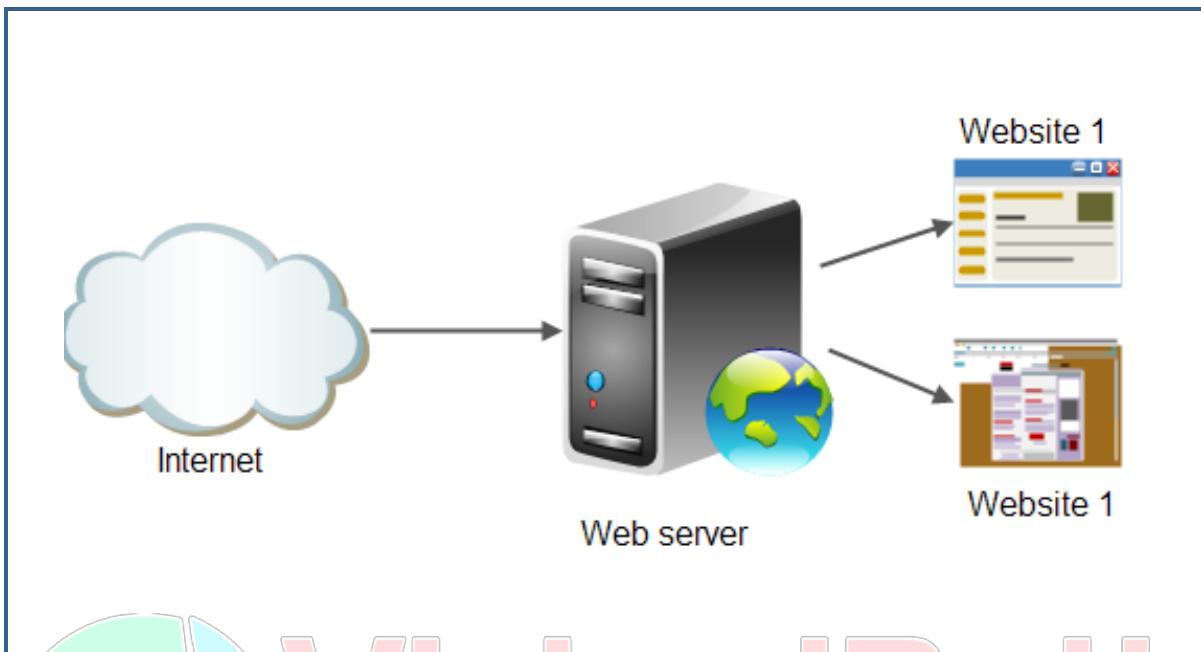
To delete the mails stucked in mailq use the following command

#postsuper -d ALL

**Note:** If internet connection is available and you have a registered domain on internet, then you can send the mails directly to any mail address in the world, but unregistered domains mail will be rejected..

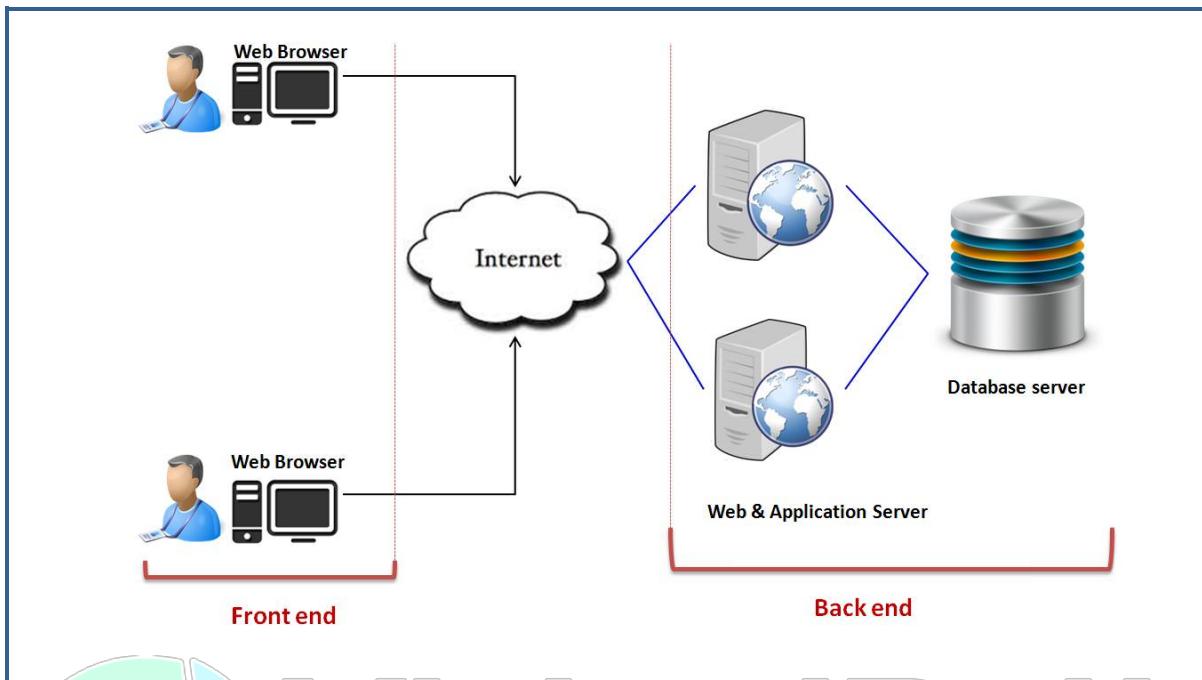
*That's it, keep working*

## WEB SERVER (APACHE)



- Every Web site sits on a computer known as a Web server. This server is always connected to the internet. Web servers are computers that deliver (serves up) Web pages. Every Web server has an IP address and possibly a domain name.
- A web server can mean two things - a computer on which a web site is hosted and a program that runs on such a computer. So the term web server refers to both hardware and software.
- A web server is what makes it possible to be able to access content like web pages or other data from anywhere as long as it is connected to the internet. The hardware houses the content, while the software makes the content accessible through the internet.
- The most common use of web servers is to host websites but there are other uses like data storage or for running enterprise applications. There are also different ways to request content from a web server. The most common request is the Hypertext Transfer Protocol (HTTP), but there are also other requests like the Internet Message Access Protocol (IMAP) or the File Transfer Protocol (FTP).

## How a Web Server Works



A simple exchange between the client machine and Web server goes like this:

1. The client's browser dissects the URL into a number of separate parts, including address, path name and protocol.
2. A Domain Name Server (DNS) translates the domain name the user has entered into its IP address, a numeric combination that represents the site's true address on the Internet (a domain name is merely a "front" to make site addresses easier to remember).
3. The browser now determines which protocol (the language client machines use to communicate with servers) should be used. Examples of protocols include FTP, or File Transfer Protocol, and HTTP, Hypertext Transfer Protocol.
4. The server sends a GET request to the Web server to retrieve the address it has been given. For example, when a user types <http://www.example.com/1.jpg>, the browser sends a GET 1.jpg command to example.com and waits for a response. The server now responds to the browser's requests. It verifies that the given address exists, finds the necessary files, runs the appropriate scripts, exchanges cookies if necessary, and returns the results back to the browser. If it cannot locate the file, the server sends an error message to the client.
5. The browser translates the data it has been given into HTML and displays the results to the user.

## Profile for Apache Server

|                         |   |                                        |
|-------------------------|---|----------------------------------------|
| Use                     | : | Hosting a web site.                    |
| Package                 | : | httpd                                  |
| Port                    | : | 80/http                                |
| Configuration File      | : | /etc/httpd/conf/httpd.conf             |
| Sample Config File      | : | /usr/share/doc/httpd/httpd-vhosts.conf |
| Configuration directory | : | /etc/httpd/conf.d                      |
| Document Root           | : | /var/www/html                          |
| Daemon                  | : | httpd                                  |

## Steps to configure a simple web server

### Step1: Install the package

- The package for apache web server is httpd.

```
#yum install httpd* -y
```

```
[root@ adm ~]# yum install httpd* -y
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
rhel   | 3.7 kB     00:00 ...
Setting up Install Process
Package httpd-2.2.15-5.el6.i686 already installed and latest version
Package httpd-tools-2.2.15-5.el6.i686 already installed and latest version
Resolving Dependencies
--> Running transaction check
--> Package httpd-devel.i686 0:2.2.15-5.el6 set to be updated
--> Processing Dependency: apr-util-devel for package: httpd-devel-2.2.15-5.el6
Installed:
  httpd-devel.i686 0:2.2.15-5.el6          httpd-manual.noarch 0:2.2.15-5.el6

Dependency Installed:
  apr-devel.i686 0:1.3.9-3.el6           apr-util-devel.i686 0:1.3.9-3.el6
  cyrus-sasl-devel.i686 0:2.1.23-8.el6    db4-cxx.i686 0:4.7.25-16.el6
  db4-devel.i686 0:4.7.25-16.el6         expat-devel.i686 0:2.0.1-9.1.el6
  openldap-devel.i686 0:2.4.19-15.el6

Complete!
```



### Step2: Copy a sample file to /etc/httpd/conf.d folder and edit it.

- #cp /usr/share/doc/httpd/httpd-vhosts.conf /etc/httpd/conf.d/msw.conf

```
[root@mlinux3 ~]# cp /usr/share/doc/httpd/httpd-vhosts.conf /etc/httpd/conf.d
```

- Edit the copied file
- #vim /etc/httpd/conf.d/msw.conf

```
<VirtualHost *:@@Port@@>
  ServerAdmin webmaster@dummy-host.example.com
  DocumentRoot "@@ServerRoot@@/docs/dummy-host.example.com"
  ServerName dummy-host.example.com
  ServerAlias www.dummy-host.example.com
  ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
  CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>
```

Keep these lines in file and edit it with your preferences.

- Edit the file as below

```
#####MY SIMPLE WEBSITE#####
<VirtualHost 192.168.106.81:80>
    ServerAdmin root@mlinux1.my.com
    DocumentRoot "/var/www/html"
    ServerName mlinux1.my.com
    ServerAlias mlinux1.my.com
    ErrorLog "/var/log/httpd/msw-error_log"
    CustomLog "/var/log/httpd/msw-access_log" common
</VirtualHost>
```

**Step2:** Navigate to the document root folder i.e. /var/www/html/ and create an index.html file which will be accessed through a web browser

- #vim /var/www/html/index.html

```
<h1>MY SIMPLE WEBSITE</h1>
<h2>WELCOME TO WEB SERVER </h2>
```

**Step3:** Start the service and enable it in boot configuration

```
#systemctl start httpd; systemctl enable httpd
```

```
[root@mlinux1 html]# systemctl start httpd; systemctl enable httpd
ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/m
```

**Step3:** Allow http in firewall trusted list

```
[root@mlinux1 html]# firewall-cmd --add-service=http --permanent
success
[root@mlinux1 html]# firewall-cmd --reload
success
[root@mlinux1 html]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpcv6-client dns ftp http mounted nfs rpc-bind samba ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

**Step4:** Access the Server via web browser like Firefox, etc.

- Open Firefox web browser and type the IP Address of the web server  
<http://192.168.106.81>



Same thing with hostname

- <http://mlinux1.my.com>

- (note: the client must be integrated with DNS, to get this thing work)



- To open the website from command line use the following command

```
#curl <IP/HOSTNAME of web server>
```

```
#curl 192.168.106.81
```

```
[root@mlinux2 Desktop]# curl 192.168.106.81
<h1>MY SIMPLE WEBSITE</h1>
<h2>WELCOME TO WEBSERVER </h2>
[root@mlinux2 Desktop]#
```

```
[root@mlinux2 Desktop]# curl mlinux1.my.com
<h1>MY SIMPLE WEBSITE</h1>
<h2>WELCOME TO WEBSERVER </h2>
[root@mlinux2 Desktop]#
```

*Also Try #elinks --dump 192.168.10.95 and check the output*

#### DNS configuration to get the feel of “www”

- Open the DNS configuration file and add the canonical name as “www”, so that we can use our domain as full fledged website.

```
#vim /var/named/my.rlz
```

```
[root@mlinux1 html]# vim /var/named/my.rlz
                                201610121      ; serial
                                1D              ; refresh
                                1H              ; retry
                                1W              ; expire
                                3H )            ; minimum
NS      mlinux1.my.com.
mlinux1 A      192.168.106.81
www    CNAME   mlinux1
mlinux2 A      192.168.106.82
mlinux3 A      192.168.106.83
mlinux4 A      192.168.106.84
```

- Restart the DNS services

```
#systemctl restart/reload named
```

```
[root@node1 ~]# systemctl reload named
```

- Okay! now we are ready, point the browser to the address as follows  
[www.my.com](http://www.my.com)



**Note:** This will only work in your DNS range, from others who are not in DNS use ip addresses

### To create an Alias/other page in the same website

- Navigate to the document root i.e. /var/www/html/ and create a folder

```
[root@mlinux1 ~]# cd /var/www/html
[root@mlinux1 html]# mkdir sec
[root@mlinux1 html]# cd sec
[root@mlinux1 sec]# vim index.html
```

- Create an index.html

```
<h1>MY SIMPLE WEBSITE SECOND PAGE</h1>
<h2>WELCOME TO WEB SERVER </h2>
```

- Open the configuration file /etc/httpd/conf.d/msw.conf and add a line as alias  
#vim /etc/httpd/conf/httpd.conf (RHEL6)  
#vim /etc/httpd/conf.d/httpd-vhosts.conf (RHEL7)

```
#####MY SIMPLE WEBSITE#####
<VirtualHost 192.168.106.81:80>
    ServerAdmin root@mlinux1.my.com
    DocumentRoot "/var/www/html"
    Alias /2 /var/www/html/sec
    ServerName mlinux1.my.com
    ServerAlias mlinux1.my.com
    ErrorLog "/var/log/httpd/msw-error_log"
    CustomLog "/var/log/httpd/msw-access_log" common
</VirtualHost>
```

- Reload the httpd service  
#systemctl reload httpd

```
[root@node1 ~]# systemctl reload httpd
```

- Open the Firefox web browser and type the following url  
<http://192.168.106.81/2> or <http://mlinux1.my.com/2>



MY SIMPLE WEBSITE SECOND PAGE

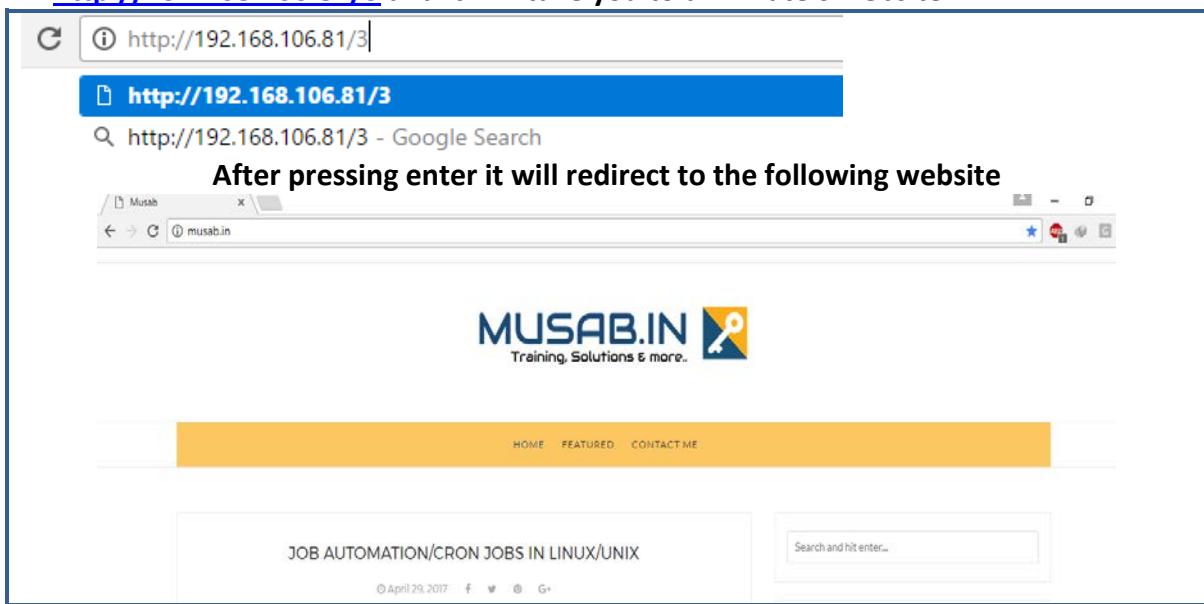
WELCOME TO WEB SERVER

### Redirecting the page to other website:

- Redirecting is to take the visitor to other website while they visit a particular page in our website
- To redirect a website, navigate and open the configuration file of http i.e. /etc/httpd/conf.d/msw.conf and add the following line in it at the end.  
`#vim /etc/httpd/conf.d/msw.conf`

```
#####MY SIMPLE WEBSITE#####
<VirtualHost 192.168.106.81:80>
    ServerAdmin root@mlinux1.my.com
    DocumentRoot "/var/www/html"
    Alias /2 /var/www/html/sec
    Redirect /3 "http://www.musab.in"
    ServerName mlinux1.my.com
    ServerAlias mlinux1.my.com
    ErrorLog "/var/log/httpd/msw-error_log"
    CustomLog "/var/log/httpd/msw-access_log" common
</VirtualHost>
```

- Reload the httpd service  
`#systemctl reload httpd`
- Take a browser and type the following website address  
<http://192.168.106.81/3> and it will take you to unixmate's website



After pressing enter it will redirect to the following website

<http://www.virtualpathtech.com>

VirtualPath Techno Solutions

## Virtual Web hosting

**Virtual hosting** is a method for hosting multiple domain names on a server using a single IP address. This allows one server to share its resources, such as memory and processor cycles, in order to use its resources more efficiently.

### Port based Hosting:

- The default port number for HTTP is 80. However, most web servers can be configured to operate on almost any port number, provided the port number is not in use by any other program on the server.
- For example, a server may host the website www.example.com. However, if the owner wishes to operate a second site, and does not have access to the domain name configuration for their domain name, and/or owns no other IP addresses which could be used to serve the site from, they could instead use another port number, for example, www.example.com:81 for port 81, www.example.com:8000 for port 8000, or www.example.com:8080 for port 8080.

## Steps to configure a port based web hosting

Step1: Make a directory for port based web hosting in document root i.e. /var/www/ say port.

```
#mkdir /var/www/port
```

```
[root@ktadm ~]# mkdir /var/www/port
[root@ktadm ~]# cd /var/www/
[root@ktadm www]# ls
cgi-bin  error  html  icons  manual  port
[root@ktadm www]#
```

Step2: Navigate to port directory and create an index.html file there

```
[root@ adm ~]# cd /var/www/port/
[root@ adm port]# vim index.html
<h1>MY PORT BASED WEBSITE</h1>
<h2>WELCOME TO WEB SERVER</h2>
```

Step3: Navigate to /etc/httpd/httpd.conf and copy the old file i.e., msw.conf with new name for new website

```
#cd /etc/httpd/conf.d/httpd
```

```
#cp msw.conf port.conf
```

```
[root@node1 ~]# cd /etc/httpd/conf.d/
[root@node1 conf.d]# ls
autoindex.conf  manual.conf  msw.conf  README  userdir.conf  welcome.conf
[root@node1 conf.d]# cp msw.conf port.conf
```

```
#vim /etc/httpd/conf.d/port.conf
```

```
#####
PORT BASED WEBSITE #####
Listen 8080
<VirtualHost 192.168.106.81:8080>
    ServerAdmin root@mlinux1.my.com
    DocumentRoot "/var/www/port"
    ServerName mlinux1.my.com
    ErrorLog "/var/log/httpd/port-error_log"
    CustomLog "/var/log/httpd/port-access_log" common
</VirtualHost>
```

#### Step4: Reload the httpd service

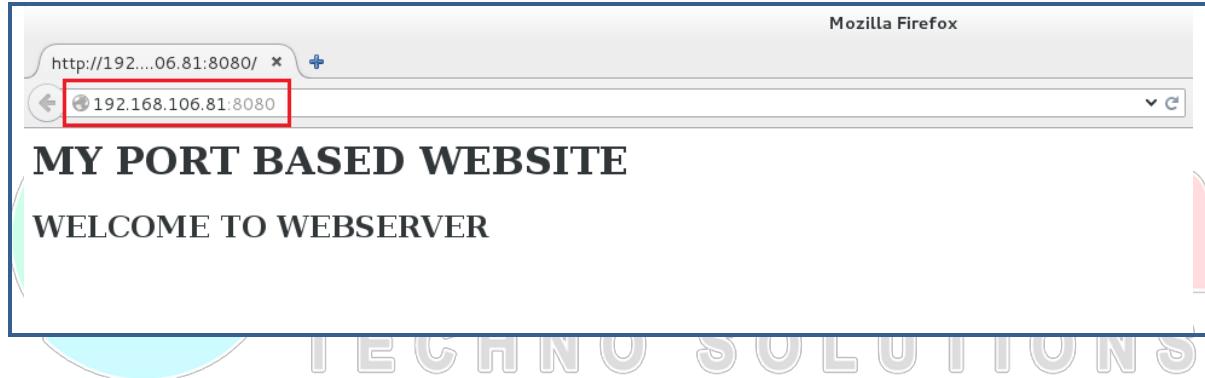
```
#systemctl reload httpd
```

```
[root@node1 ~]# systemctl reload httpd
```

- Allow the port no 8080 in firewall in RHEL7

```
[root@mlinux1 port]# firewall-cmd --add-port=8080/tcp --permanent
success
[root@mlinux1 port]# firewall-cmd --reload
success
[root@mlinux1 port]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpc6-client dns ftp http mountd nfs rpc-bind samba ssh
  ports: 8080/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Open Firefox and type <http://192.168.106.81:8080>



#### Name Based Virtual web Hosting

- Name-based virtual hosts use multiple host names for the same web server IP address.
- With web browsers that support HTTP/1.1 (as nearly all now do), upon connecting to a webserver, the browsers send the hostname from the address that the user typed into their browser's address bar along with the requested resource itself to the web server. The server can use the Host header field to determine which web site (or *virtual host*), as well as page, to show the user. The browser specifies the address by setting the Host HTTP header with the host specified by the user. The Host header is required in all HTTP/1.1 requests.
- For instance, a server could be receiving requests for two domains, www.example.com and www.example.net, both of which resolve to the same IP address. For www.example.com, the server would send the HTML file from the directory /var/www/user/Joe/site/, while requests for www.example.net would make the server serve pages from /var/www/user/Mary/site/.
- Example: A blog server can be hosted using Name base hosting. blog1.example.com and blog2.example.com

<http://www.virtualpathtech.com>

VirtualPath Techno Solutions

## Steps to configure name based web hosting:

### Step1: Make a directory in document root i.e. /var/www/ with some name say "ktname"

```
#mkdir /var/www/name
```

- Add an index page in it

```
#vim /etc/var/www/name/index.html
```

```
<h1>MY NAME BASED WEBSITE</h1>
<h2>WELCOME TO WEB SERVER</h2>
```

### Step2: Update the DNS zone configuration files with the new hostname of the web server

```
#vim /var/named/my.flz
```

```
$TTL 1D
@ IN SOA mlinux1.my.com. root.my.com. (
    201610121 ; serial
    1D ; refresh
    1H ; retry
    1W ; expire
    3H ) ; minimum

    NS mlinux1.my.com.
mlinux1 A 192.168.106.81
www CNAME mlinux1
prod A 192.168.106.81
mlinux2 A 192.168.106.82
mlinux3 A 192.168.106.83
mlinux4 A 192.168.106.84
```

```
#vim /var/named/my.rlz
```

```
$TTL 1D
@ IN SOA mlinux1.my.com. root.my.com. (
    201610121 ; serial
    1D ; refresh
    1H ; retry
    1W ; expire
    3H ) ; minimum

    NS mlinux1.my.com.
mlinux1 A 192.168.106.81
81 PTR mlinux1.my.com.
81 PTR prod.my.com.
82 PTR mlinux2.my.com.
83 PTR mlinux3.my.com.
84 PTR mlinux4.my.com.
```

- Reload the DNS services

```
#systemctl reload named
```

```
[root@node1 ~]# systemctl reload named
```

**Step5:** Navigate to /etc/httpd/httpd.conf and copy the old file i.e., port.conf with new name for new website

```
#cd /etc/httpd/conf.d/httpd
#cp port.conf name.conf
```

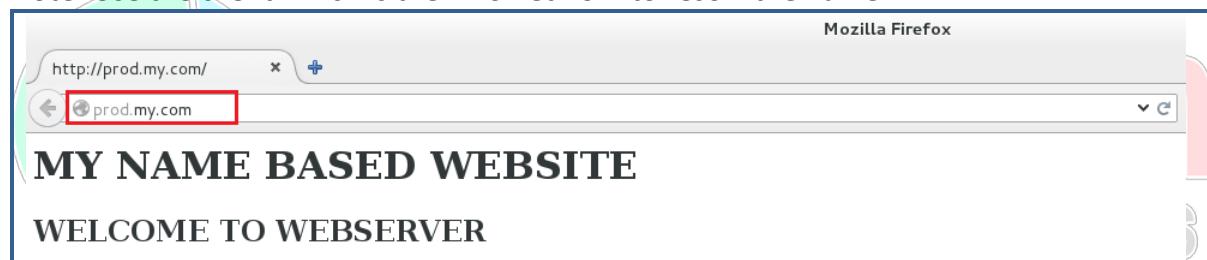
```
[root@node1 ~]# cd /etc/httpd/conf.d
[root@node1 conf.d]# ls
autoindex.conf manual.conf msw.conf port.conf README userdir.conf welcome.conf
[root@node1 conf.d]# cp port.conf name.conf
[root@node1 conf.d]# vim name.conf
#####
# NAME BASED WEBSITE #####
NameVirtualHost 192.168.106.81:80
<VirtualHost 192.168.106.81:80>
    ServerAdmin root@prod.my.com
    DocumentRoot "/var/www/name"
    ServerName prod.my.com
    ErrorLog "/var/log/httpd/name-error_log"
    CustomLog "/var/log/httpd/name-access_log" common
</VirtualHost>
```

**Step6:** Reload the httpd services

```
#systemctl reload httpd
```

http://prod.my.com now it will navigate to our name based web page

Note: Use the client which is the DNS network to resolv the name.



**Steps to configure vnc-server for graphical connectivity to any linux machine**

1. Install vncserver package

```
#yum install tigervnc-server -y
```

2. Set vnc passwd for access

```
#vncpasswd
```

Repeat the password twice

Set readonly passwd Y/N → No

3. Start vnc services

```
#vncserver
```

4. Allow vnc server in firewall.

```
#firewall-cmd --add-service=vnc-server --permanent; firewall-cmd --reload
```

5. Install vnc-viewer on client and try to connect with server address: session no.

*There are still many things to learn in webserver, try googling to learn more concepts*

## KICKSTART AND NETWORK INSTALLATIONS

- Many system administrators would prefer to use an automated installation method to install Red Hat Enterprise Linux on their machines. To answer this need, Red Hat created the kickstart installation method. Using kickstart, a system administrator can create a single file containing the answers to all the questions that would normally be asked during a typical installation.
- Kickstart files can be kept on a single server system and read by individual computers during the installation. This installation method can support the use of a single kickstart file to install Red Hat Enterprise Linux on multiple machines, making it ideal for network and system administrators.
- Kickstart installations can be performed using a local CD-ROM, a local hard drive, or via NFS, FTP, or HTTP

To use kickstart, you must:

1. Create a kickstart file.
2. Create a boot media with the kickstart file or make the kickstart file available on the network.
3. Make the installation tree available.
4. Start the kickstart installation.

Let's configure the kickstart installation by following above steps:

### 1. Create a kickstart file

*Kickstart* configuration files can be built manually. Additionally the standard Red Hat installation program Anaconda will produce a *kickstart* configuration file at the end of any manual installation process. This file can then be taken and either used to automatically reproduce the same installation or edited (either manually or with *system-config-kickstart*).

Copy anaconda-ks.cfg at ftp location or http location through which you want to share it.

```
#cp /root/anaconda-ks.cfg /var/ftp/pub/ks.cfg
```

```
[root@mlinux3 ~]# cp /root/anaconda-ks.cfg /var/ftp/pub/ks.cfg
[root@mlinux3 ~]# ls /var/ftp/pub/
ks.cfg  rhel8
[root@mlinux3 ~]#
```

- Make the changes in the file as required

```
[root@mlinux3 ~]# cat /var/ftp/pub/ks.cfg
#version=RHEL8
ignoredisk --only-use=sda
# Partition clearing information
clearpart --all --initlabel
# Use graphica/textl install
graphical
# Use CDROM installation media
cdrom
# Keyboard layouts
keyboard --vckeymap=in-eng --xlayouts='in (eng)'
# System language
lang en_IN.UTF-8

# Network information
network --bootproto=dhcp --device=enp0s3 --onboot=off --ipv6=auto --no-activate
network --hostname=localhost.localdomain
repo --name="AppStream" --baseurl=file:///run/install/repo/AppStream
# Root password
rootpw --iscrypted
$6$rvlt35FK1L7cdUUS$91te5TAy.Tmz9fqWZIraG3ARRmy.b4uUNGeAwpVVp5AUQeGAO
z5ncZ6mLQsj6Lvr1MfAFs34FN5Y2Pypzswpj/
# X Window System configuration information
xconfig --startxonboot
# Run the Setup Agent on first boot
firstboot --disable
# Reboot after installation
reboot
# System services
services --enabled="chronyd"
# Intended system purpose
syspurpose --role="Red Hat Enterprise Linux Server" --sla="Self-Support" --
usage="Development/Test"
# System timezone
timezone Asia/Kolkata --isUtc
user --name=myuser --
password=$6$Kh6/u743HK0TE4mG$HM0yoOhOSMFVvCeWO7IkjHobYTI1yi8EuL8fd/WK
1ESWQav3uvfDUdF71U604S/1SzZmDZBnGEI.y5zwXrbMY1 --iscrypted --gecos="myuser"
# Disk partitioning information
part / --fstype="xfs" --ondisk=sda --size=15360
part /boot --fstype="xfs" --ondisk=sda --size=1024
part swap --fstype="swap" --ondisk=sda --size=2048

%packages
@^graphical-server-environment
kexec-tools
```

```
%end

%addon com_redhat_kdump --enable --reserve-mb='auto'

%end

%anaconda
pwpolicy root --minlen=6 --minquality=1 --notstrict --nochanges --notempty
pwpolicy user --minlen=6 --minquality=1 --notstrict --nochanges --emptyok
pwpolicy luks --minlen=6 --minquality=1 --notstrict --nochanges --notempty
%end
```

If you are planning for LVM based partitioning scheme, use following parameters

```
# Disk partitioning information
part /boot --fstype="xfs" --ondisk=sda --size=1024
part swap --fstype="swap" --ondisk=sda --size=4096
part pv.263 --fstype="lvmpv" --ondisk=sda --size=12296
volgroup rhel --pesize=4096 pv.263
logvol / --fstype="xfs" --size=8192 --name=root --vgname=rhel
logvol /var --fstype="xfs" --size=2048 --name=var --vgname=rhel
logvol /home --fstype="xfs" --size=2048 --name=home --vgname=rhel
```

To use different language, keyboard lang and time zone use the following commands

#### Listing languages

```
[root@mlinux3 ~]# localectl list-locales
C.utf8
en_AG
en_AU
en_AU.utf8
en_BW
en_BW.utf8
```

#### Listing Keyboard layout names

```
[root@mlinux3 ~]# localectl list-keymaps |grep en
ca-eng
cm-french
en-latin9
gh-generic
in-eng
```

#### Listing different Time Zones name

```
[root@mlinux3 ~]# timedatectl list-timezones
```

Generating encrypted password for root or other users to use in kickstart file

```
[root@mlinux3 ~]# openssl passwd -1 redhat
$1$hM5ZPKj2$u5HhCD6PCBefJpDv65TjY0
```

Note: lastly start ftp services and add it in firewall

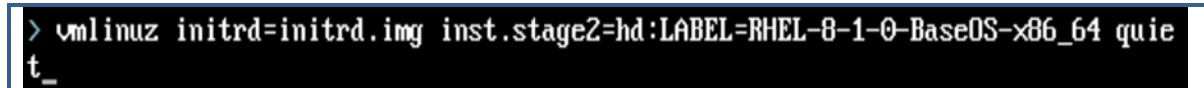
```
#systemctl start vsftpd.service; #systemctl enable vsftpd.service
#firewall-cmd --add-service=ftp --permanent
```

## Client side operation for kickstart

- Boot the system using RHEL 8 DVD and press “*Tab*” at splash screen



- After pressing “*tab*”, the following screen will be displayed

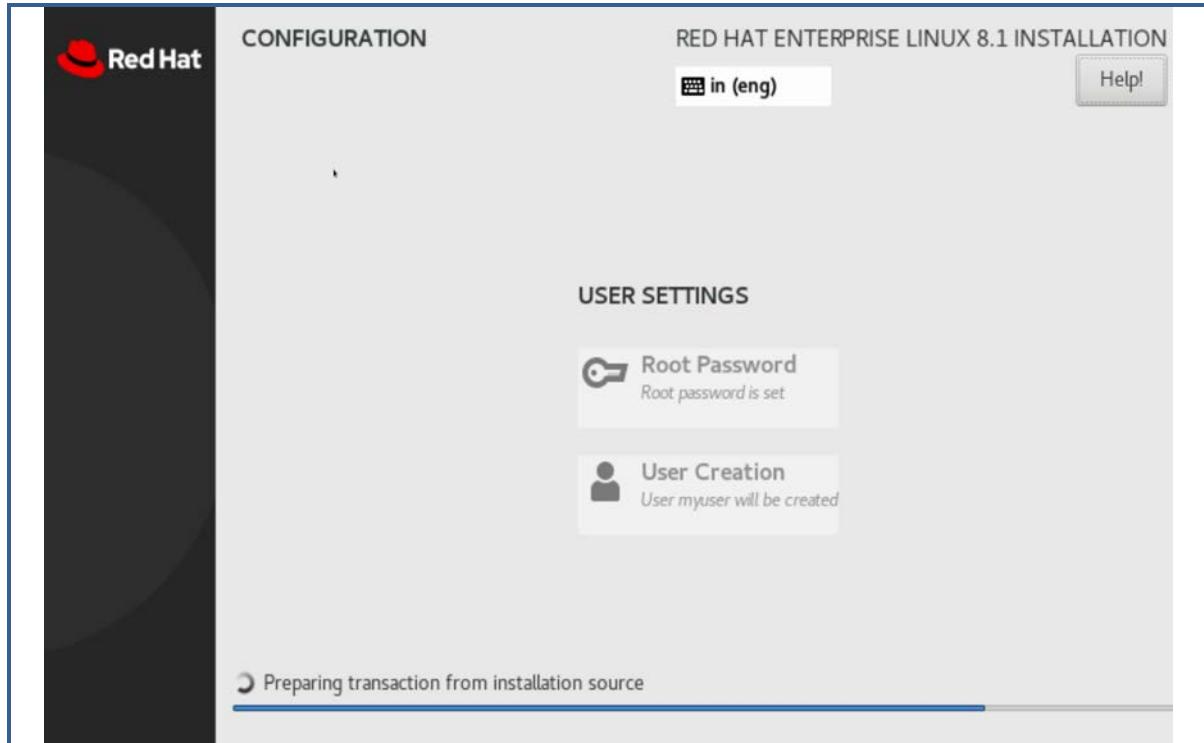


- Type the following information about the kickstart file and its server and also assign some IP address to the machine to communicate with kickstart server.



Note: If you have a dhcp server in your environment, no need to assign IP

- After entering above information, press enter to continue with your kickstart installation. Wait till installation is completed.



## NETWORK INSTALLATIONS

Network installations can be performed using following methods.

- FTP
- NFS
- HTTP

### Steps to perform installation using FTP:

- Copy the entire RHEL7 DVD on document root of ftp i.e. /var/ftp/pub/

```
[root@mlinux3 ~]# cd /var/ftp/pub/  
[root@mlinux3 pub]# ls  
ks.cfg  rhel8
```

### Start installation using FTP directory

- As we are trying to install RHEL7 from network, we still require a boot media to get the boot screen to type the required command.
- To get the boot screen we can have a media like CD/DVD or USB drive with *boot.iso* image copied in it.
- “*boot.iso*” image will produce a boot screen as follows



### **To make a DVD / Pen Drive bootable using boot.iso image**

- Download the boot.iso from redhat website.
- Copy the boot.iso in DVD or PENDRIVE using following command

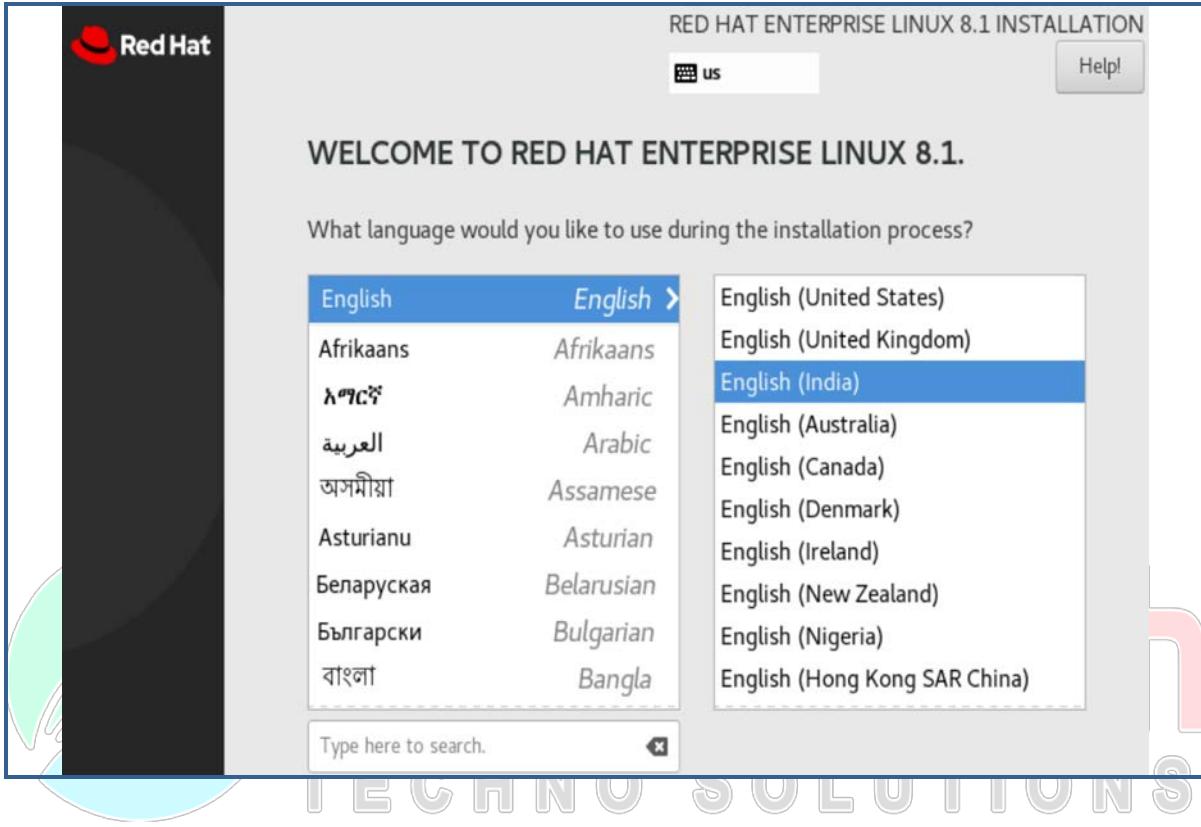
#### For DVD

```
# cdrecord /root/boot.iso (where "/root/boot.iso" is the path of boot.iso image)
```

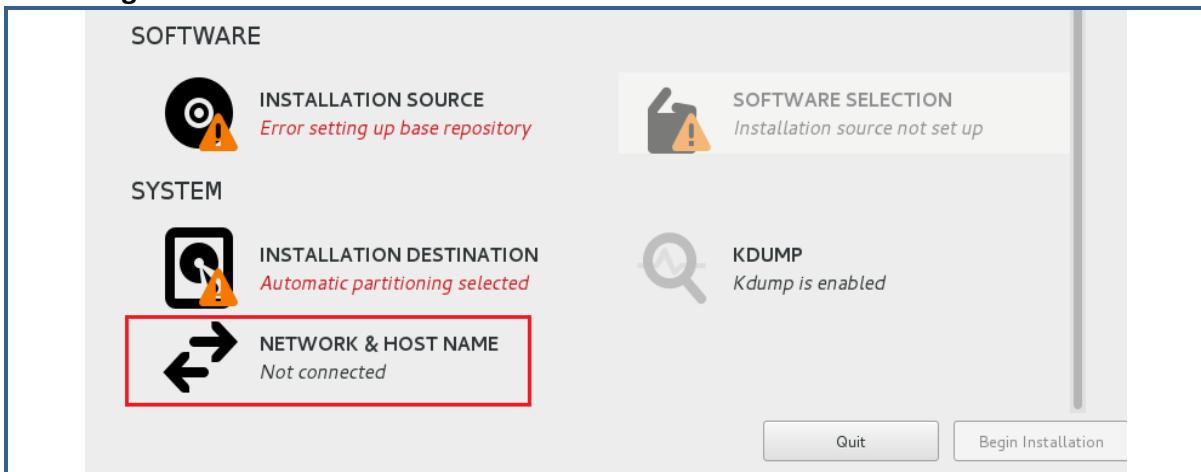
#### For USB Drive

```
#dd if=/root/boot.iso of=/dev/sdb1 (where /dev/sdb1 is the address of the USB drive)
```

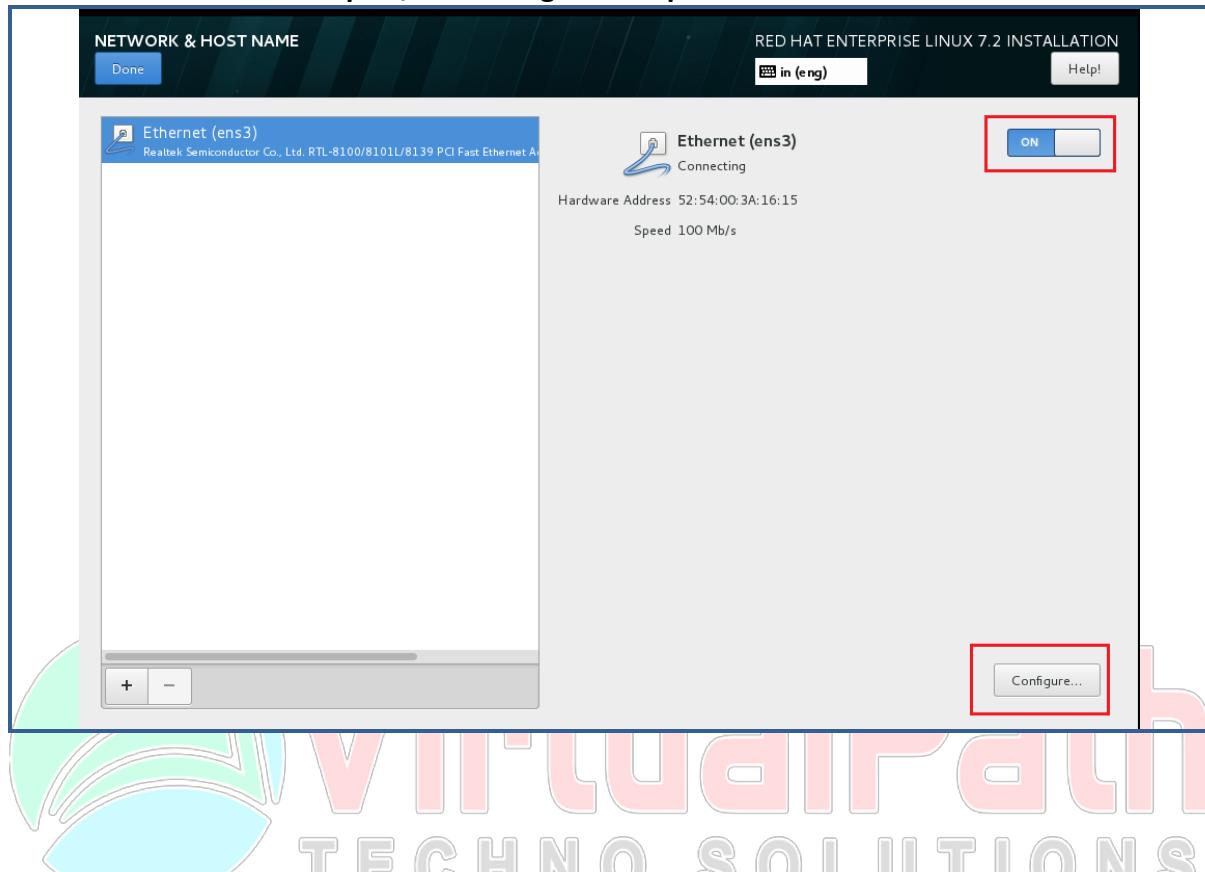
- After a while the following prompt will be display where you can select the preferred language



- Configure network in order to use network installation



- Activate ethernet adapter, and configure the ip



- Assign some IP address to your machine so that it can communicate with the server. Make sure that IP should be in the same range that of the server.

Address	Netmask	Gateway
192.168.10.90	24	

**Method:** Manual

**Addresses**

DNS servers: [ ]

Search domains: [ ]

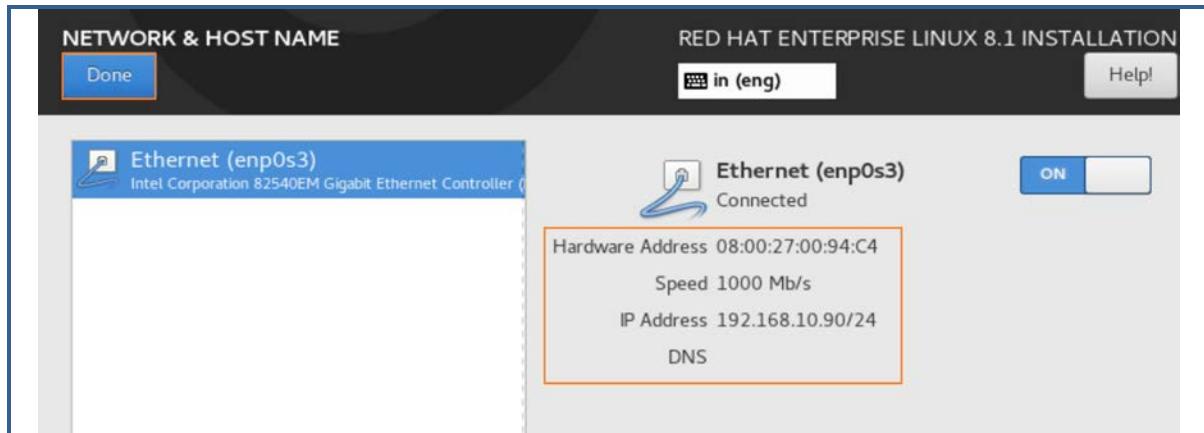
DHCP client ID: [ ]

Require IPv4 addressing for this connection to complete

Routes... [ ]

Cancel [ ] Save [ ]

- Once the network is configured, it would appear on screen, and click on done to continue



- Now, click on software tab to configure network server details



- Select ftp and provide the details of ftp server



## Configuring NFS and http servers for network installations

### NFS Configuration

- Make an entry in **/etc/exports** to export the RHEL7 media.
- Let us say my RHEL8 DVD is dumped in **/var/ftp/pub/rhel8** directory

<b>/var/ftp/pub/rhel8</b>	<b>* (rw, sync)</b>
---------------------------	---------------------

- Use the ***exportfs*** command to export the directory.

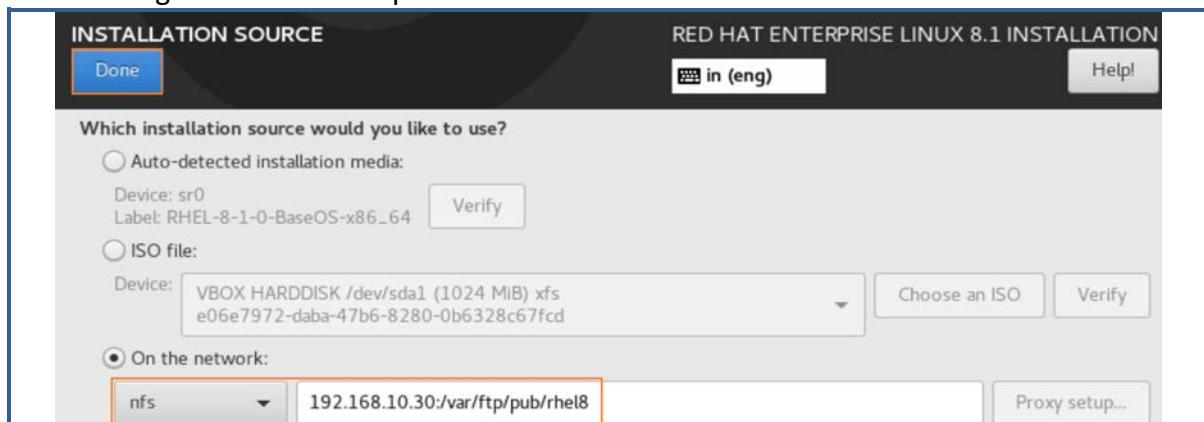
```
[root@mlinux3 ~]# exportfs -rv
exporting *:/var/ftp/pub/rhel8
```

Don't forget to restart the services of NFS

Your NFS server is now ready to host the media, Login to client and start the N/W installation.

### Client Side Setup

- Follow the same steps of what we have done in **ftp** method, the only change will be selecting NFS instead of ftp



*That's it; your installation will be started from NFS server*

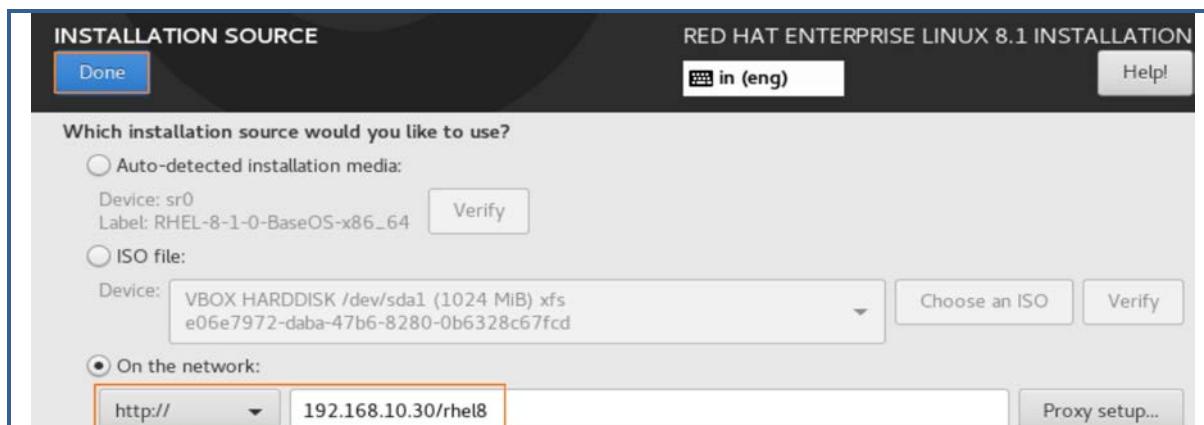
### HTTP Configuration for network installations

- Copy the **RHEL7 DVD** dump in the document root of http i.e., **/var/www/html**, or else create a soft link of the rhel7 directory in the document root of http.

```
[root@mlinux3 ~]# ln -s /var/ftp/pub/rhel8 /var/www/html/
[root@mlinux3 ~]# ls -l /var/www/html/
total 4
-rw-r--r--. 1 root root 27 Apr 10 21:17 index.html
drwxr-xr-x. 2 root root 24 Apr 10 21:19 pg2
lrwxrwxrwx. 1 root root 18 Apr 14 22:33 rhel8 -> /var/ftp/pub/rhel8
[root@mlinux3 ~]#
```

- start the services of http if needed, and you are done with the server side configuration**

### Client side setup



*Once continued with "Done" installation will be started through http*

## PERFORMING FULLY AUTOMATED INSTALLATION BY COMBINING KICKSTART AND NETWORK INSTALLATION

In such type of installation we will take the media from network and also use kickstart to answer all the queries asked during installation.

### Creating a Kickstart file with network enabled installation predefined

- Create a kickstart file as shown earlier, the only change we need to do is in method of installation. Type the url for network server either **FTP, NFS or HTTP**.

#vim /var/ftp/pub/ks.cfg

```
# Use graphical/text install
text
# Use network server installation media
url --url http://192.168.10.30/rhel8
# Keyboard layouts
```

- We can select any method of installation from the list and specify the details regarding the server.
- If using ftp to access the kickstart file save it in document root of ftp

### Client side setup

- Boot the system with boot.iso image and press *Esc* when blue screen is appear.
- Give the information for kickstart file as shown below



**Observe that an automated installation will be perform and the installation media will be taken from network.**

```
* if the graphical installation interface fails to start, try again with the
inst.text bootoption to start text installation
* when reporting a bug add logs from /tmp as separate text/plain attachments
17:47:01 Not asking for VNC because of an automated install
17:47:01 Not asking for VNC because text mode was explicitly asked for in kickstart
Starting automated install.....
Generating updated storage configuration
Checking storage configuration...

=====
=====
Installation

1) [x] Language settings
    (English (India))
3) [x] Installation source
    (http://192.168.10.30/rhel8)
5) [x] Installation Destination
    (Custom partitioning selected)
7) [x] Network configuration
    (Wired (enp0s3) connected)

2) [x] Time settings
    (Asia/Kolkata timezone)
4) [x] Software selection
    (Custom software selected)
6) [x] Kdump
    (Kdump is enabled)

=====
=====
Progress

.
Setting up the installation environment
.
Configuring storage
.
Creating disklabel on /dev/sda
Creating xfs on /dev/sda1
```

**TECHNO SOLUTIONS**

Isn't it amazing....!

My dear students, this sums up our course, but it does not end learning Linux. Linux is like an Ocean and there are many things which you can learn in it. So keep doing the good work and do R&D, read blogs and master your skills. Keep visiting my website

[www.musab.in](http://www.musab.in)

I wish you all “*best of luck*” for your future. I hope this small contribution of mine would be a good help for your career. God bless

## **BONUS STUFF FOR SELF-LEARNING**

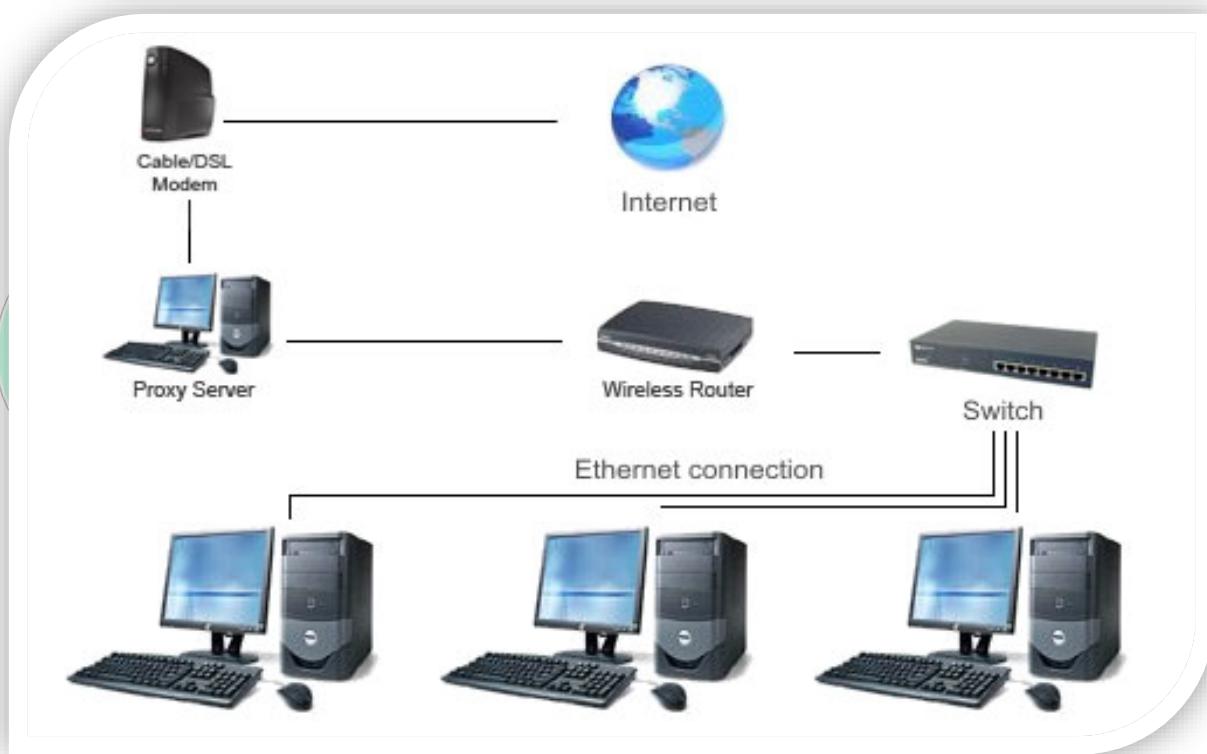


**For video tutorials of the following topics please visit my  
blog <http://www.musab.in>**

**<http://musab.in/2018/03/configuring-proxy-and-dhcp-on-centos-rhel7/>**

## SQUID PROXY SERVER

A proxy server is one that receives requests intended for another server and that acts on the behalf of the client (as the client proxy) to obtain the requested service. It is often used when the client and the server are incompatible for direct connection. For example, the client may be unable to meet the security authentication requirements of the server but may be required to access some services. It may also be used for screening purposes to enable the administrator to control access to undesirable sites. The proxy server may also be used for caching purposes, which enables faster access to frequently used websites. All the computers connected to the LAN access the Internet through a single IP address, resulting in improved security simply because the number of ports exposed is reduced.



### Profile for Squid proxy server

Use	:	To Share Internet, Restrict unwanted websites.
Package	:	squid
Port	:	3128 (default)
Configuration Files	:	/etc/squid/squid.conf
Daemon	:	squid

## Configuring a proxy server for internet sharing:

**Step1:** Check and Install the squid package

```
[root@ linux ~]# rpm -qa squid
[root@ linux ~]# yum install squid* -y
Loaded plugins: product-id, refresh-packagekit, security, subscription-manager
Updating certificate-based repositories.
ftp://192.168.10.96/pub/rhel6/repo/epel/repomd.xml: [Errno 14] PYCURL ERROR 7 - "
couldn't connect to host"
Trying other mirror.
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package squid.x86_64 7:3.1.10-1.el6_1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version            Repository      Size
=====
Installing:
squid            x86_64   7:3.1.10-1.el6_1.1  KTRERO          1.7 M

Transaction Summary
=====
[root@ adm ~]# rpm -qa squid
squid-3.1.10-1.el6_1.1.x86_64
```

**Step2:** Edit the configuration file for squid i.e. “/etc/squid/squid.conf”, Add the network range from where the clients can connect to proxy server.

#vim /etc/squid/squid.conf

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
acl mynet src 192.168.104.0/24
http_access allow mynet
```

**Step3:** Start the squid services and make it permanent

#systemctl start squid; systemctl enable squid (RHEL7)

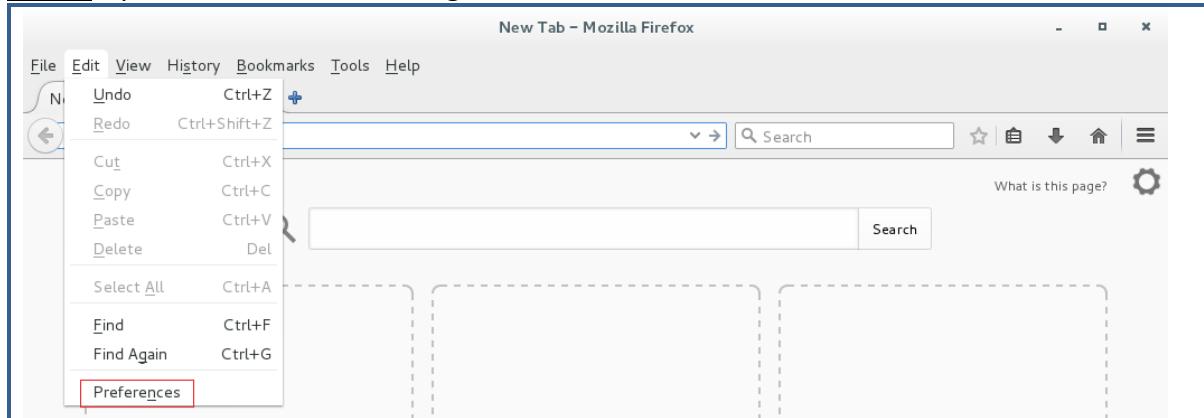
```
[root@myrhel73 ~]# systemctl start squid; systemctl enable squid
Created symlink from /etc/systemd/system/multi-user.target.wants/squid.service
[root@myrhel73 ~]#
```

**Step4:** Allow the squid service in firewall in RHEL7

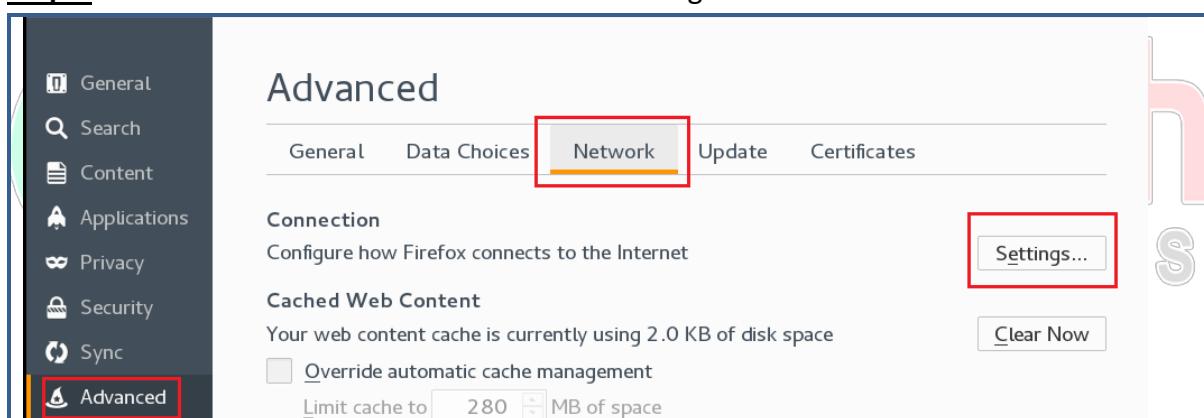
```
[root@myrhel73 ~]# firewall-cmd --add-service=squid --permanent
success
[root@myrhel73 ~]# firewall-cmd --reload
success
```

## Client side configuration for receiving internet:

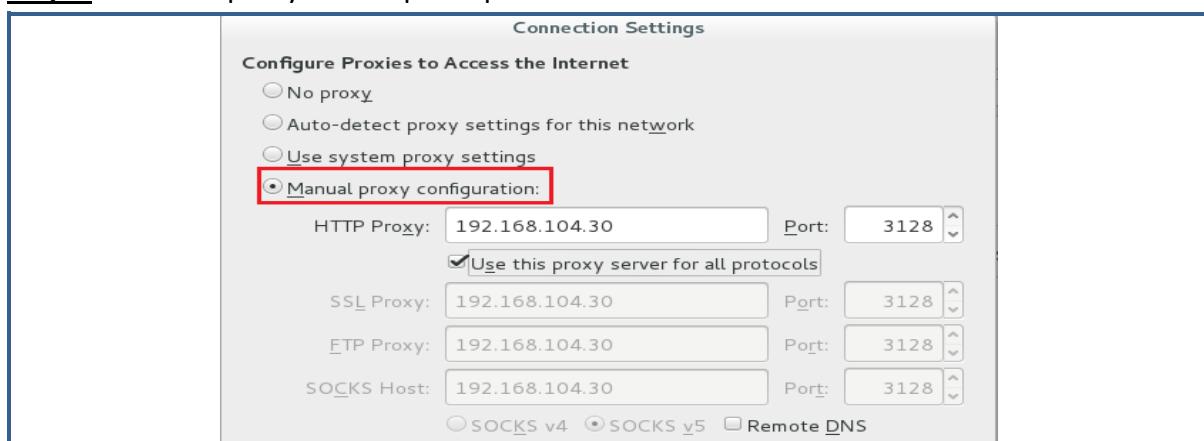
**Step1:** Open Browser, ex: firefox, go to edit/tool → Preferences



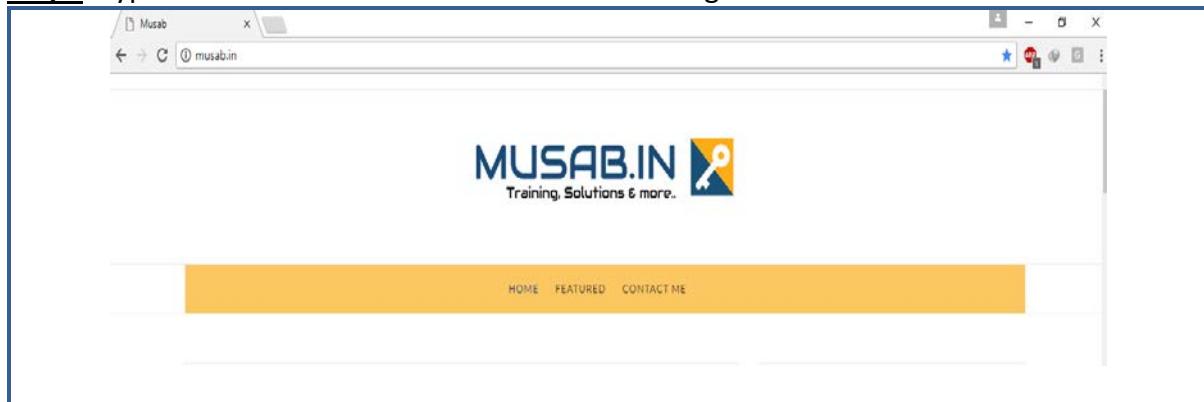
**Step2:** Go to Advanced → Network and select Settings



**Step3:** Enter the proxy server ip and port number as shown below



**Step4:** Type the website address to see if it connects



### Blocking websites through proxy:-

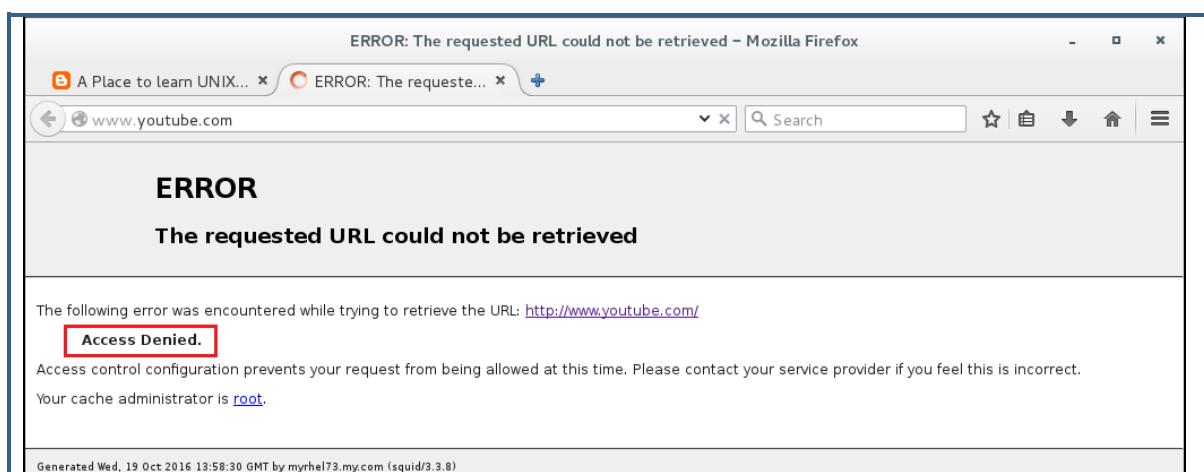
**Step1:** Go to the configuration file, **/etc/squid/squid.conf** and add the following parameters.

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
acl block url_regex youtube
http_access deny block
acl mynet src 192.168.104.0/24
http_access allow mynet
```

**Step2:** Restart/Reload the squid services (preferably reload)

```
[root@myrhel73 ~]# systemctl reload squid
[root@myrhel73 ~]#
```

**Step3:** Check with the browser can you access [www.hotmail.com](http://www.hotmail.com) through your browser



## Blocking multiple sites using proxy:

**Step1:** Create a file in /etc/squid with any name and add the phrase of the website, which you want to block

```
#vim /etc/squid/block
```

```
hotmail
youtube
facebook
twitter
yahoo
~
```

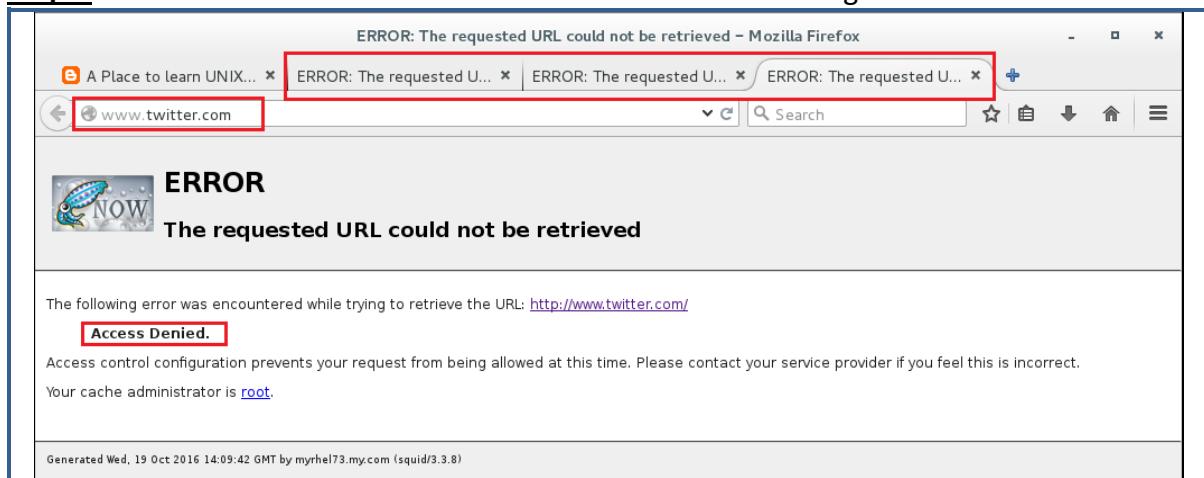
**Step2:** Add the same file info in configuration file, i.e., /etc/squid/squid.conf

```
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
acl block url_regex "/etc/squid/block"
http_access deny block
acl mynet src 192.168.104.0/24
http_access allow mynet
```

**Step3:** Restart/Reload the squid services (preferably reload)

```
[root@myrhel73 ~]# systemctl reload squid
[root@myrhel73 ~]#
```

**Step4:** Go to client browser and check whether the sites are being blocked



## Changing the default port of Proxy:

**Step1:** By default the port no. for proxy is 3128, which can be changed by making a small change in the configuration file as shown below and change it to 8000.

```
[root@ktlinux2 ~]# vim /etc/squid/squid.conf
```

```
# Squid normally listens to port 3128
http_port 8000
```

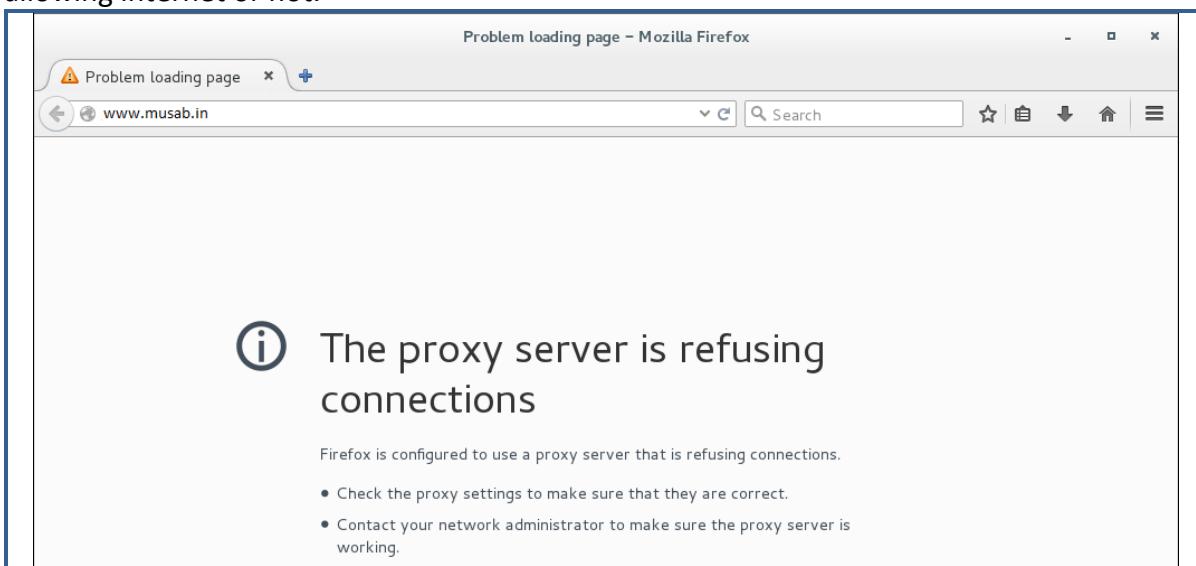
**Step2:** Restart/Reload the squid services (preferably reload)

```
[root@myrhe173 ~]# systemctl reload squid
[root@myrhe173 ~]#
```

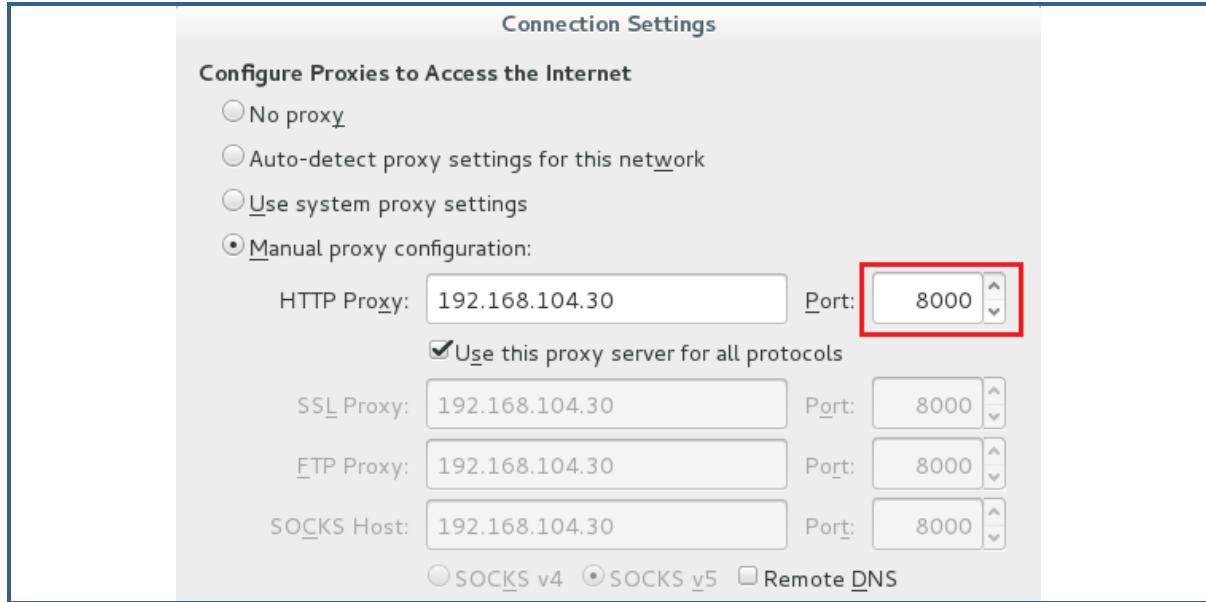
**Step3:** Allow the new port into firewall

```
[root@myrhe173 ~]# firewall-cmd --add-port=8000/tcp --permanent
success
[root@myrhe173 ~]# firewall-cmd --reload
success
[root@myrhe173 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: enp0s3 enp0s8
  sources:
  services: dhcp dhcpcv6-client squid ssh
  ports: 8000/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

**Step4:** Go to client's browser and check whether with default port, i.e. 3128, whether it is allowing internet or not.



**Step 4:** change the port to 8000 and check whether internet is allowed or not.



Connection Settings

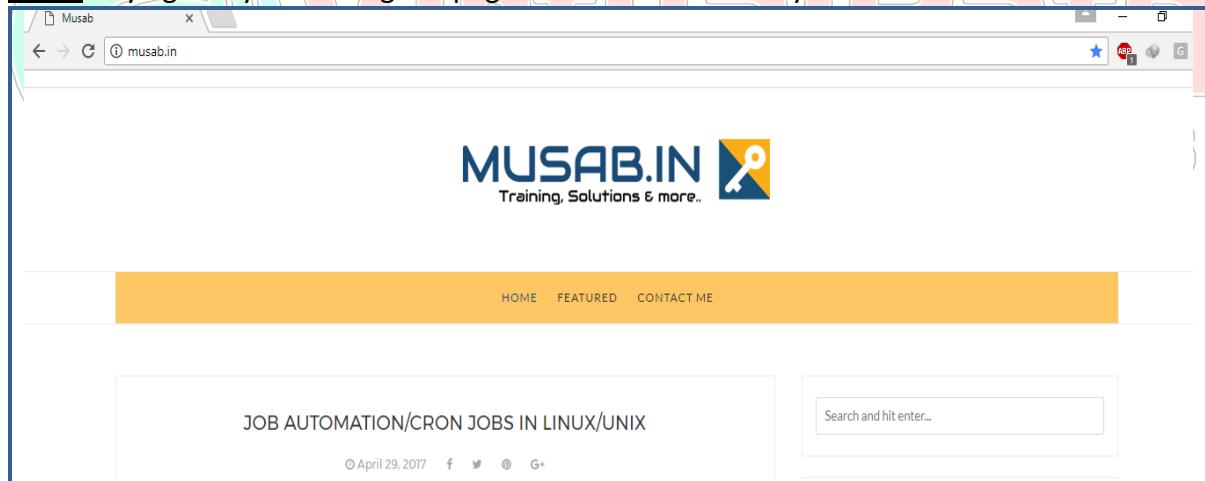
Configure Proxies to Access the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- Manual proxy configuration:

HTTP Proxy:	192.168.104.30	Port:	8000
<input checked="" type="checkbox"/> Use this proxy server for all protocols			
SSL Proxy:	192.168.104.30	Port:	8000
FTP Proxy:	192.168.104.30	Port:	8000
SOCKS Host:	192.168.104.30	Port:	8000

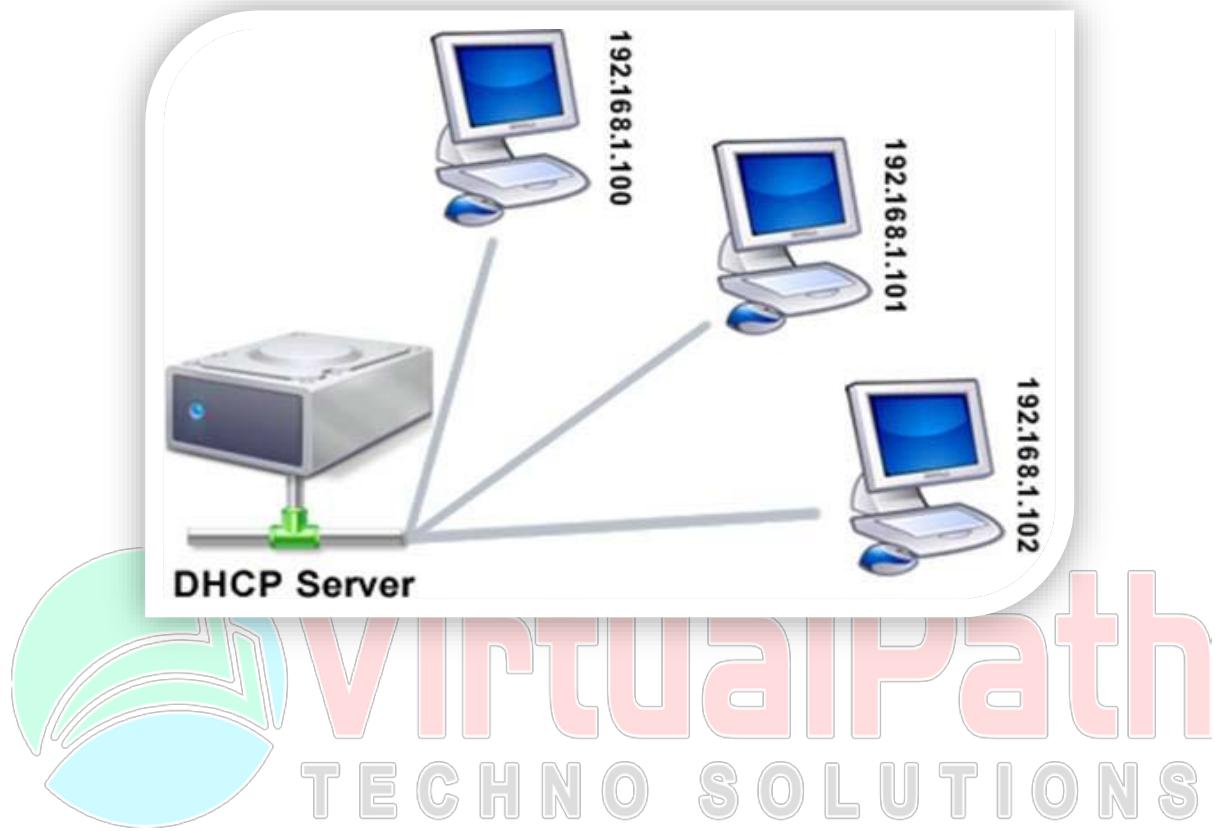
SOCKS v4    SOCKS v5    Remote DNS

**Step6:** Try again by refreshing the page and it would certainly work



**Note:** Squid Proxy is only the basic proxy, to learn more on proxy google for the third party tools like; **Squidguard, Untangle and Smoothwall**. There is lot to do with squid, try doing google and read the /etc/squid/squid.conf for more information.

## DHCP SERVER



### What is DHCP?

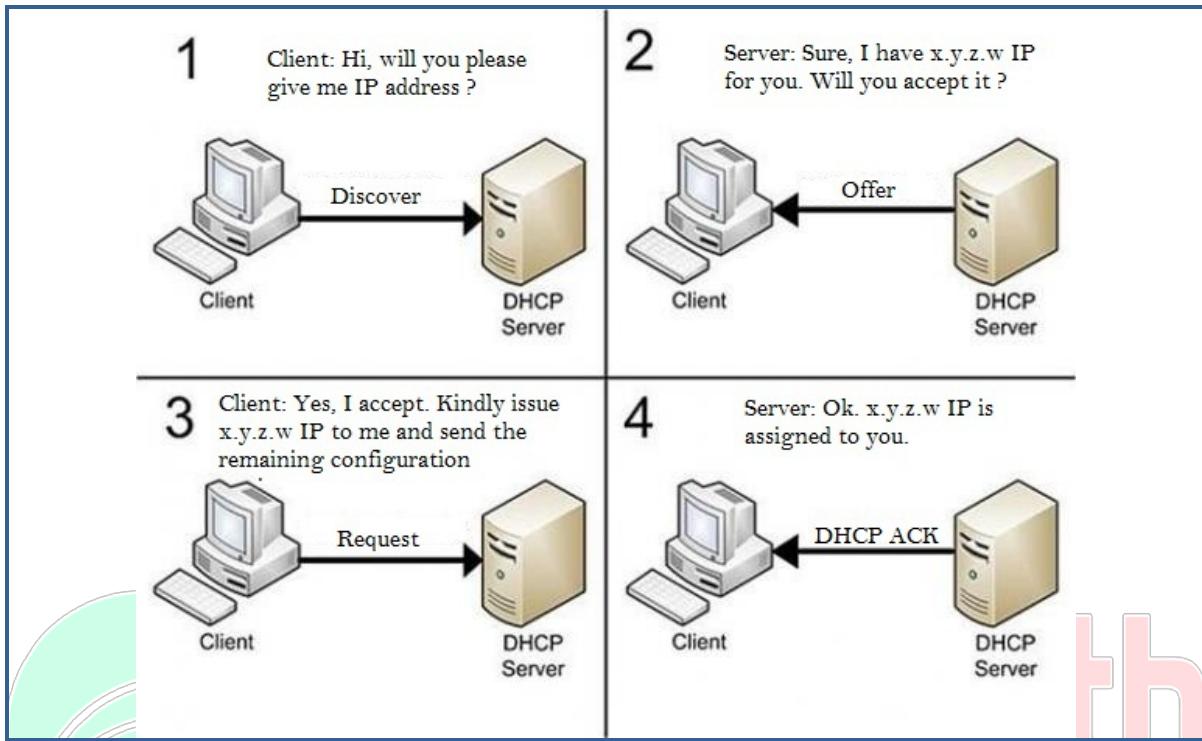
Dynamic Host Configuration Protocol (DHCP) is a network protocol that enables a server to automatically assign an IP address to a computer from a defined range of numbers (i.e., a scope) configured for a given network.

**DHCP** allows a computer to join an IP-based network without having a pre-configured IP address. DHCP is a protocol that assigns unique IP addresses to devices, then releases and renews these addresses as devices leave and re-join the network.

Internet service providers usually use DHCP to help customers join their networks with minimum setup effort required. Likewise, home network equipment like broadband routers offer DHCP support for added convenience in joining home computers to local area networks (LANs).

## How does DHCP works?

The “DORA” process in DHCP



DHCP assigns an IP address when a system is started, for example:

1. A user turns on a computer with a DHCP client.
2. The client computer sends a broadcast request (called a **DISCOVER** or **DHCPODISCOVER**), looking for a DHCP server to answer.
3. The router directs the DISCOVER packet to the correct DHCP server.
4. The server receives the DISCOVER packet. Based on availability and usage policies set on the server, the server determines an appropriate address (if any) to give to the client. The server then temporarily reserves that address for the client and sends back to the client an **OFFER** (or **DHCPOFFER**) packet, with that address information. The server also configures the client's DNS servers, WINS servers, NTP servers, and sometimes other services as well.
5. The client sends a **REQUEST** (or **DCHPREQUEST**) packet, letting the server know that it intends to use the address.
6. The server sends an **ACK** (or **DHCPACK**) packet, confirming that the client has been given a lease on the address for a server-specified period of time.

When a computer uses a static IP address, it means that the computer is manually configured to use a specific IP address. One problem with static assignment, which can result from user error or inattention to detail, occurs when two computers are configured with the same IP address. This creates a conflict that results in loss of service. Using DHCP to dynamically assign IP addresses minimizes these conflicts.

### Profile for DHCP server

Usage	:	To assign IP's to the computers in the network dynamically.
Package	:	Dhcp
Configuration file	:	/etc/dhcp/dhcpd.conf
Port no	:	67, 68
Daemon	:	dhcpd

Note: Pl watch the video tutorial to configure dhcp server on my website using following url

<http://musab.in/2018/03/configuring-proxy-and-dhcp-on-centos-rhel7/>

### Configuring a DHCP server:

Step1: Check whether the package is installed or not

```
[root@ cl5 ~]# rpm -q dhcp  
package dhcp is not installed
```

Step2: Install the package using yum,

```
[root@ cl5 ~]# yum install dhcp* -y  
Loaded plugins: product-id, refresh-packagekit, security, subscription-manager  
Updating certificate-based repositories.  
Setting up Install Process  
Package 12:dhcp-common-4.1.1-25.P1.el6.x86_64 already installed and latest version  
Resolving Dependencies  
--> Running transaction check  
--> Package dhcp.x86_64 12:4.1.1-25.P1.el6 will be installed  
--> Finished Dependency Resolution
```

Step3: Copy the example file for dhcp configuration over dhcp configuration file,  
i.e., </etc/dhcp/dhcpd.conf>

```
# cp -p /usr/share/doc/dhcp*/dhcpd.conf.sample /etc/dhcp/dhcpd.conf
```

Step4: Open the configuration file and edit it as per the requirement.

```
[root@ cl5 ~]# vim /etc/dhcp/dhcpd.conf

# A slightly different configuration for an internal subnet.
subnet 192.168.106.0 netmask 255.255.255.0 {
    range 192.168.106.10 192.168.106.30;
    option routers 192.168.106.1;
    default-lease-time 600;
    max-lease-time 7200;
}
```

Subnet	: The subnet of the network
Netmask	: The netmask of the network
Range	: The range of IP address to be assigned to the clients, in short "Scope"
Option routers	: gateway address (optional)
Default-lease-time	: The minimum lease time of the ip assigned to the clients
Max-lease-time	: The maximum lease time of the IP assigned to the clients

Step5: Make sure the dhcp server contains same range static IP as follows

```
[root@mlinux1 ~]# ifconfig ens3
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.106.81 netmask 255.255.255.0 broadcast 192.168.106.255
inet6 fe80::5054:ff:fe92:fd24 prefixlen 64 scopeid 0x20<link>
ether 52:54:00:92:fd:24 txqueuelen 1000 (Ethernet)
RX packets 6888 bytes 809163 (790.1 KiB)
RX errors 0 dropped 65 overruns 0 frame 0
TX packets 1950 bytes 283016 (276.3 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Step6: Start/Restart the dhcp services and make it permanent

```
[root@mlinux1 ~]# systemctl start dhcpcd; systemctl enable dhcpcd
```

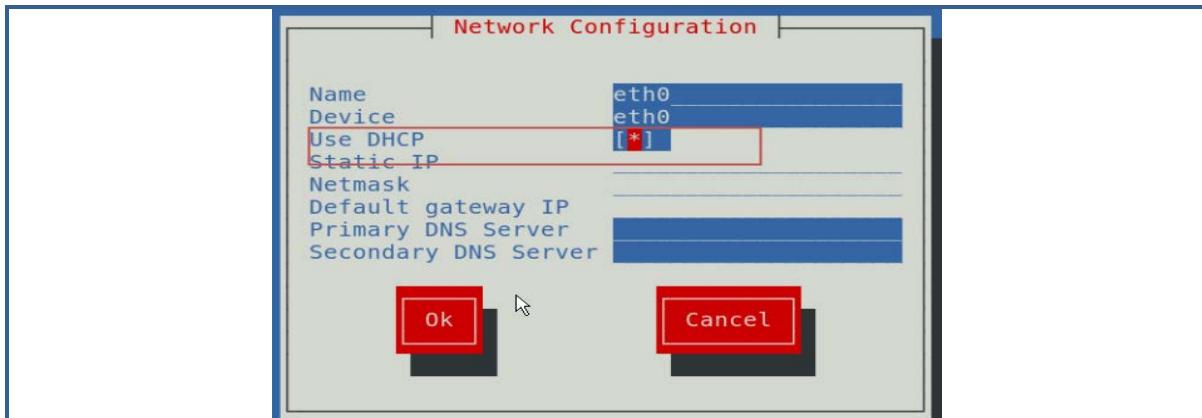
Step 7: Allow dhcp in firewall

```
[root@mlinux1 ~]# firewall-cmd --add-service=dhcp --permanent
success
[root@mlinux1 ~]# firewall-cmd --reload
success
```

## Client side configuration for DHCP:

### RHEL6 as a client

Step1: Make the dhcp option enabled in network configuration using #setup command.



Step2: Restart the network services and check the IP address is in dhcp scope.

```
[root@ cl6 Desktop]# setup
[root@ cl6 Desktop]# service network restart
Shutting down interface eth0: Device state: 3 (disconnected)
[ OK ]
Shutting down loopback interface: [ OK ]
Bringing up loopback interface: [ OK ]
```

Step3: Check the IP address using #ifconfig command

```
[root@ cl6 Desktop]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 52:54:00:5A:40:95
          inet addr:192.168.100.10 Bcast:192.168.100.255 Mask:255
                      inet6 addr: fe80::5054:ff:fe5a:4095/64 Scope:Link
                        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                        RX packets:141 errors:0 dropped:0 overruns:0 frame:0
                        TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
                        collisions:0 txqueuelen:1000
                        RX bytes:8914 (8.7 KiB) TX bytes:1730 (1.6 KiB)
                        Interrupt:11 Base address:0x4000
```

## RHEL7 as a client

Step1: Make the dhcp option enabled in network configuration using *nmcli* command

```
#nmcli con mod ens#(connection name) ipv4.method auto (dhcp)
```

```
[root@mlinux1 ~]# nmcli con mod ens8 ipv4.method auto
```

Step2: activate the connection

```
#nmcli con up <con-name>
```

```
[root@mlinux1 ~]# nmcli con up ens8
Connection successfully activated (D-Bus active path:
[root@mlinux1 ~]#
```

Step3: Check the IP address using *#ifconfig* command

```
[root@mlinux1 ~]# ifconfig ens8
ens8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.106.14 netmask 255.255.255.0 broadcast 192.168.106.255
inet6 fe80::5054:ff:fe12:8546 prefixlen 64 scopeid 0x20<link>
ether 52:54:00:12:85:46 txqueuelen 1000 (Ethernet)
RX packets 6554 bytes 697510 (681.1 KiB)
RX errors 0 dropped 65 overruns 0 frame 0
TX packets 131 bytes 23863 (23.3 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```





# SHELL SCRIPTING REFERENCE CUM LAB MANUAL

*Version 3.0*

*By*

*Musabuddin Syed (Redhat Certified)*

VirtualPath Techno Solutions  
#101, Namdev Block, Balaji Towers  
Prime Hospital Lane, Ameerpet Hyd-TS  
Contact: +91 799 309 6092, 040-6666 6092  
Email: [info@virtualpathtech.com](mailto:info@virtualpathtech.com)  
Website: [www.virtualpathtech.com](http://www.virtualpathtech.com), [www.musab.in](http://www.musab.in)

# Table of Contents

1. Introduction to Shell Scripting.....	01-04
2. Getting Started .....	05-06
3. Echo Command.....	06-08
4. Colorizing the Output .....	09-11
5. Miscellaneous Commands .....	12-16
6. Quotes and Filter commands .....	17-27
7. AWK Command .....	28-32
8. SED Command .....	33-44
9. Variables, Command Substitutions and Arrays.....	45-53
10. Read Command for User Input .....	54-55
11. Tests, if Conditions and Compound Expressions.....	56-74
12. LOOPS (while, until, for and Select loops).....	75-87
13. Functions.....	88-92
14. Option Parsing & GETOPTS .....	93-101
15. Sample classroom scripts .....	102-117
16. Additional Stuff and Scripts .....	118-122

## INTRODUCTION

### **What's Kernel**

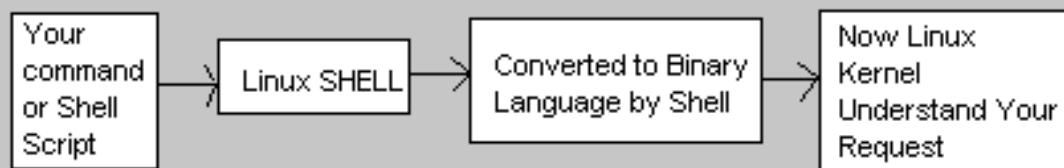
Kernel is the heart of Linux O/S. It manages resource of Linux O/S. Resources means facilities available in Linux. For e.g. Facility to store data, print data on printer, memory, file management etc. Kernel decides who will use this resource, for how long and when. It runs your programs (or set up to execute binary files) its Memory resident portion of Linux. It performs following task:-

- I/O management
- Process management
- Device management
- File management
- Memory management

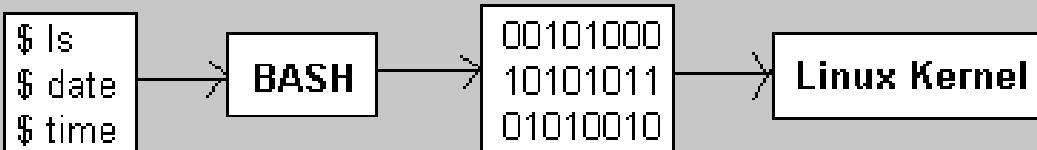
### **What's Linux Shell?**

Computer understand the language of 0's and 1's called binary language, In early days of computing, instruction are provided using binary language, which is difficult for all of us, to read and write. So in O/S there is special program called Shell. Shell accepts your instruction or commands in English and translate it into computer's native binary language.

#### This is what Shell Does for Us



You type Your command and shell convert it as



- Its environment provided for user interaction.
- Shell is a command language interpreter that executes commands read from the standard input device (keyboard) or from a file.

### Types of Shells in Linux:

- Shell is not part of the system kernel, but it uses system kernel to execute the programs.
- There are 13 types of shells widely used across, Among them SH, BASH, KSH are more popular

TYPES OF SHELLS	DEVELOPED BY	WHERE
BSH/ SH (Bourne SHell)	Stephen Bourne	AT & T Bell Labs
BASH ( Bourne-Again SHeLL )	Brian Fox and Chet Ramey	Free Software Foundation
CSH (C SHeLL)	Bill Joy	University of California (For BSD)
KSH (Korn SHeLL)	David Korn	AT & T Bell Labs

### Accessing a shell in Linux:

As soon as we login into the system successfully, we would be landing into the home directory and by default we would be given a program that would be running in the background call shell. To use a shell you just need to login into the system and you would be able to access it.

### What is a Shell Script?

Shell scripting is writing a series of command for the shell to execute. It can combine lengthy and repetitive sequences of commands into a single and simple script, which can be stored and executed anytime. This reduces the effort required by the end user.

### Why Shell Scripting?

- Shell script can take input from user, file and output them on screen.
- Useful to create our own commands.
- Save lots of time.
- To automate some task of day today life.
- System Administration part can be also automated.

### Limitation of Shell Script:

- Compatibility problems between different platforms.
- Slow execution speed.
- A new process launched for almost every shell command executed.

### **Creating a Shell Script:**

- To write shell script we can use any of the Linux's text editor such as vi, vim, nano, emacs etc., even you can use cat command.
- It is preferable to use vi or vim editor as it is widely used and known by many Linux admins.

### **Basics of Unix/Linux:**

- Before we can go to shell scripting, let's review some important basics of Unix/Linux administration.

### **Getting help in Unix/Linux:**

To get help in Unix/Linux command line, we can make use of following commands after command name.

- **help**
- **man**
- **info**

### **File Types supported in Unix/Linux:**

- UNIX Supports variety of files among them the basic types of files are.

Character	File Type
-	Regular file.
l	Symbolic Link
c	Character Special
b	Block Special
p	Named Pipe
s	Socket file.
d	Directory

- To determine the type of the file we can get it from ls -l command. The first character defines the type.

```
[root@mlinux7 ~]# ls -l myfile
-rw-r--r--. 1 root root 12 Oct  1 19:00 myfile
[root@mlinux7 ~]#
```

## File Permissions:

### Permissions are applied on three levels:

- Owner or User level
- Group level
- Others level

### Access modes are of three types:

- **r** read only
- **w** write/edit/delete/append
- **x** execute/run a command

### Access modes are different on file and directory:

Permissions	Files	Directory
R	Open the file	'ls'/list the contents of directory
W	Write, edit, append, delete file	Add/Del/Rename contents of directory
X	To run a command/shell script	To enter into directory using 'cd'

```
[root@musab1 ~]# ls -l myfile
-rw-r--r--. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]# ls -ld mydir
drwxr-xr-x. 2 root root 4096 Feb 13 16:43 mydir
```

Filetype+permission, links, owner, group name of owner, size in bytes, date of modification, file name

- To change the ownership permissions of file using **chown** command and this command can only be executed by root user.
- To change the file actions permissions of a file using **chmod** command and this can be given by the owner of the file.

## Let's Get Started:

### Shebang:

The **#!** Syntax used in scripts to indicate an interpreter for execution under UNIX / Linux operating systems. Most Linux shell and perl / python script starts with the following line:

```
#!/bin/bash
OR
#!/usr/bin/perl
OR
#!/usr/bin/python
```

- It is called a shebang or a "bang" line.
- It is nothing but the absolute path to the Bash interpreter (shell).
- It consists of a number sign and an exclamation point character (**#!**), followed by the full path to the interpreter such as `/bin/bash`.
- All scripts under Linux execute using the interpreter specified on a first line.
- Almost all bash scripts often begin with `#!/bin/bash` (assuming that Bash has been installed in `/bin`)
- This ensures that Bash will be used to interpret the script, even if it is executed under another shell.
- The shebang was introduced by Dennis Ritchie between Version 7 Unix and 8 at Bell Laboratories. It was then also added to the BSD line at Berkeley
- If no shebang interpreter line is added, then it would consider `/bin/sh` as default. It is recommended to give `#!/bin/bash` for best results.

#### **Do you know?**

- In musical notation, a **"#"** is called a **sharp** and an exclamation point - **"!"** - is sometimes referred to as a **bang**. Thus, **shebang** becomes a shortening of **sharp-bang**. The term is mentioned in Elizabeth Castro's *Perl and CGI for the World Wide Web*.

### Comment:

- You should be aware of the fact that you might not be the only person reading your code. A lot of users and system administrators run scripts that were written by other people. If they want to see how you did it, comments are useful to enlighten the reader.
- Comments also make your own life easier. Say that you had to read a lot of man pages in order to achieve a particular result with some command that you used in your script. You won't remember how it worked if you need to change your script after a few weeks or months, unless you have commented what you did, how you did it and/or why you did it.

```
#!/bin/bash
echo "Hello World" #printing a message --> this is a comment
```

### Executing a Shell Script:

→ The most simplest and the easiest way to run a script in Unix/Linux is as follows

```
[root@mlinux71 ~]# sh script1.sh
```

Or

```
[root@mlinux71 ~]# bash script1.sh
```

→ Another widely used method is;

```
[root@mlinux71 ~]# ./script1.sh
```

→ Do you know that almost 90% of first time writers of shell script will get below error

```
[root@mlinux71 ~]# ./script1.sh
-bash: ./script1.sh: Permission denied
[root@mlinux71 ~]#
```

This error is because every script needs execute permission which might be missing

So, let's first apply the execute "x" permission on the file

```
[root@mlinux71 ~]# chmod +x script1.sh
[root@mlinux71 ~]# ls -l script1.sh
-rwxr-xr-x. 1 root root 73 Oct  3 18:51 script1.sh
[root@mlinux71 ~]#
```



The built-in **echo** command is an older form of **printf** command in Linux/Unix systems. It used to display the text or variables on the output screen. The following example narrates the usage of echo command.

```
[root@mlinux71 ~]# cat script1.sh
#!/bin/bash
echo WELCOME TO SHELL SCRIPTING
```

### Output

```
[root@mlinux71 ~]# ./script1.sh
WELCOME TO SHELL SCRIPTING
```

### Options:

- -n: (No New Line) a line field is automatically added after the string is displayed, It can be suppressed with the - n option.
- -e : (enable escape sequence) If the - e option is enabled then echo command will enable escape sequences

\\n - New line escape sequence.

\\t - New horizontal tab escape sequence.

### ECHO -n Option:

- ECHO Command is used to print the text on the output screen and by default it creates a new line feed automatically after the string is displayed.
- We can suppress \t using -n option and the following commands will show the difference in using -n option.

#### *Without -n Option:*

```
[root@mlinux71 ~]# echo hello world
hello world
[root@mlinux71 ~]#
```

#### *With -n Option:*

```
[root@mlinux71 ~]# echo -n hello world
hello world[root@mlinux71 ~]#
```

In the above example you can see the command prompt is in the same line instead of new line if we are using -n option.

### Echo -e option:

- Option -e stands for enabling escape sequence in **echo** command. In some cases you may require to print multiple lines of output using **echo** command. Usually we have to use multiple **echo** commands to do so, but we can do the same in **echo** command by enabling -e option in it.
- The following are the most used option in echo escape sequence.
  - \n - New line escape sequence.
  - \t - New horizontal tab escape sequence.

#### *Without -e Option:*

```
[root@mlinux7 ~]# echo Hello; echo Welcome to VPTS
Hello
Welcome to VPTS
```

In this output to get two lines we have to use two echo commands separated by a ";" semicolon.

#### *With -e Option:*

<p style="text-align: right;"><i>With \n</i></p> <pre>[root@mlinux7 ~]# echo -e "Hello\nWelcome to VPTS" Hello Welcome to VPTS</pre>	<p style="text-align: right;"><i>With \t</i></p> <pre>[root@mlinux7 ~]# echo -e "Hello\tWelcome to VPTS" Hello    Welcome to VPTS</pre>
--------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------

If the escape sequence like \n \t is used without -e it won't be treated as special characters.

### Lab Exercises:

1. Script to print the message on the output screen.

```
$vim script1.sh
#!/bin/bash
echo Hello World ### echo command is used print
### the message on the screen
```

#### Output:

```
[root@mlinux7 scripts]# ./script1.sh
-bash: ./script1.sh: Permission denied
[root@mlinux7 scripts]# chmod +x script1.sh
[root@mlinux7 scripts]# ./script1.sh
Hello World
```

2. Script to print the message on the screen using escape sequence

```
$vim script2.sh
#!/bin/bash
echo -e "Hello world\nWelcome to VPTS." ### echo command will print new line
### \n is mentioned in the message.
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script2.sh
[root@mlinux7 scripts]# ./script2.sh
Hello world
Welcome to VPTS.
```

3. Script to print the output and execute the commands using Double quotes.

```
$vim script3.sh
#!/bin/bash
### To execute commands in message of echo command
### Calling date command
echo "Date Command Output is = `date`"
### Calling who am i command
echo "Logged in as = `who am i`"
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script3.sh
[root@mlinux7 scripts]# ./script3.sh
Date Command Output is = Mon Oct 23 17:07:18 IST 2017
Logged in as = root          pts/0                2017-10-23 16:51
```

### Colorizing the Output:

Shell scripts commonly used ANSI escape codes for color output. Following table shows Numbers representing colors in Escape Sequences.

Color	Foreground	Background
Black	30	40
Red	31	41
Green	32	42
Yellow	33	43
Blue	34	44
Magenta	35	45
Cyan	36	46
White	37	47

The following is the syntax for getting colored text

```
#echo -e "\033[COLOR(CODE)m TEXT"
```

The "\033[" begins the escape sequence. You can also use "\e[" instead of "\033[", COLOR specifies a foreground color, according to table above. The "m" terminates escape sequence, and text begins immediately after that.

To print Magenta colored text:

```
[root@mlinux71 ~]# echo -e "\033[35m sample text"
sample text
[root@mlinux71 ~]#
[root@mlinux71 ~]#
```

Or

```
[root@mlinux71 ~]# echo -e "\e[35m sample text"
sample text
[root@mlinux71 ~]#
[root@mlinux71 ~]#
```

The problem with above statement is that the magenta color that starts with the 35 color code is never switched back to the regular color, so any text you type after the prompt and even prompt also is still in the Magenta color.

To return to the plain, normal mode, we have to use another sequence.

```
[root@mlinux71 ~]# echo -e "\033[0m"

[root@mlinux71 ~]#
[root@mlinux71 ~]#
```

Now you won't see anything new on the screen, as this echo statement was not passed any string to display. But it has done its job, which was to restore the normal viewing mode.

- Escape sequence also allow you to control the manner in which characters are displayed on the screen.
- The following table summarizes numbers representing text attributes in Escape Sequences.

ANSI CODE	Meaning
0	Normal Characters
1	Bold Characters
4	Underlined Characters
5	Blinking Characters
7	Reverse video Characters

Combining all these Escape Sequences, you can get more fancy effect. Use the following template for writing colored text on a colored background.

```
echo -e "\033[COLOR1;COLOR2m sample text\033[0m"
```

Example for yellow color text on blue background

```
[root@mlinux71 ~]# echo -e "\033[33;44m Yellow text on blue background\033[0m"
Yellow text on blue background
```

Bold Yellow Text on Blue background

```
[root@mlinux71 ~]# echo -e "\033[1;33;44m Bold Yellow text on blue background\033[0m"
Bold Yellow text on blue background
```

Bold Yellow Underlined text on Blue Background

```
[root@mlinux71 ~]# echo -e "\033[1;4;33;44m Bold Yellow underline text on blue background\033[0m"
Bold Yellow underline text on blue background
[root@mlinux71 ~]#
```

### Lab Exercises:

#### 1. Using Colors in Shell scripting.

```
$vim script66.sh
#!/bin/bash
# This script echoes colors and codes
echo -e "\n\033[1;4;31mLight Colors\033[0m \t\t\033[4;31mDark Colors\033[0m"

echo -e "\e[0;30;47m Black \e[0m 0 ;30m \t\e[1;30;40m Dark Gray \e[0m 1 ;30m"
echo -e "\e[0;31;47m Red \e[0m 0 ;31m \t\e[1;31;40m Dark Red \e[0m 1 ;31m"
echo -e "\e[0;32;47m Green \e[0m 0 ;32m \t\e[1;32;40m Dark Green \e[0m 1 ;32m"
echo -e "\e[0;33;47m Yellow \e[0m 0 ;33m \t\e[1;33;40m Dark Yellow\033[0m 1 ;33m"
echo -e "\e[0;34;47m Blue \e[0m 0 ;34m \t\e[1;34;40m Dark Blue \e[0m 1 ;34m"
echo -e "\e[0;35;47m Magenta \e[0m 0 ;35m \t\e[1;35;40m DarkMagenta\033[0m 1 ;35m"
echo -e "\e[0;36;47m Cyan \e[0m 0 ;36m \t\e[1;36;40m Dark Cyan \e[0m 1 ;36m"
echo -e "\e[0;37;47m LightGray\033[0m 0 ;37m \t\e[1;37;40m White \e[0m 1 ;37m"
```

Output:

```
[root@mlinux7 scripts]# chmod +x script66.sh  
[root@mlinux7 scripts]# ./script66.sh
```

Light Colors

Black	0 ;30m
Red	0 ;31m
Green	0 ;32m
Yellow	0 ;33m
Blue	0 ;34m
Magenta	0 ;35m
Cyan	0 ;36m
	0 ;37m

Dark Colors

Dark Gray	1 ;30m
Dark Red	1 ;31m
Dark Green	1 ;32m
Dark Yellow	1 ;33m
Dark Blue	1 ;34m
DarkMagenta	1 ;35m
Dark Cyan	1 ;36m
White	1 ;37m



## MISCELLANEOUS COMMANDS

There are few commands which can be used only in scripting and which may make our job easy in certain situations.

### EVAL Command:

The ***eval*** command line, performs all shell substitutions, and then executes the command line and returns the exit status of the executed command.

The following example narrates the usage of ***eval*** command.

Situation of executing the commands without ***eval*** command.

```
[root@mlinux7 scripts]# a=100
[root@mlinux7 scripts]# b='$a'
[root@mlinux7 scripts]# echo $b
$a
```

In the above example if we are expecting a value of 10 as the output but we are not getting the exact value why because the value is in single quote which is making our variable to treat-a normal character. In such situation if we need to exactly substitute the values and then the command is to be executed we have to use ***eval*** command.

The same example if you see using the ***eval*** command.

```
[root@mlinux7 scripts]# a=100
[root@mlinux7 scripts]# b='$a'
[root@mlinux7 scripts]# eval echo $b
100
```

### Colon Command (:):

The ***colon*** command is a shell initialized command and it will have only single command that exit with a status zero. That is nothing but if you execute ***colon*** command it simply returns a status zero and does nothing other than that.

```
[root@mlinux7 ~]# :
[root@mlinux7 ~]# echo $?
0
```

The colon command is usually used in ***if*** statements which requires at least one statement.

### Type Command:

The ***type*** shell built-in command will make you know whether the command you are executing is an alias, function, shell built-in or the command installed on the server. The syntax of ***type*** command is as follows

```
type <Command>
```

The following examples describe the usage of type command

```
[root@mlinux7 ~]# type ls
ls is aliased to `ls --color=auto'
[root@mlinux7 ~]# type cat
cat is /usr/bin/cat
[root@mlinux7 ~]# type vim
vim is /usr/bin/vim
[root@mlinux7 ~]# type history
history is a shell builtin
```

### Sleep Command:

The ***sleep*** command pauses for a given number of seconds. The basic syntax is

**sleep n**

Where n is the number of seconds to sleep or pause. Some types of UNIX enable other time units to be specified. It is usually recommended that n not exceed 65,534.

***Sleep*** can be used to give a user time to read an output message before clearing the screen

### Trap Command:

The trap command is used to catch a signal that is sent to a process created due to execution of script. An action is taken based on the signal by using the action which is define in the trap command instead of taking the default effect on the process. The basic syntax is as follows.

**trap '' <Signal Number>**

In the single quotes we can give the commands when appropriate signal is given trap command will execute those commands. We can get all the signals available in Linux using ***Kill -l*** command.

```
[root@mlinux7 ~]# kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
 11) SIGSEGV    12) SIGUSR2     13) SIGPIPE     14) SIGNALRM   15) SIGTERM
 16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
 21) SIGTTIN    22) SIGTTOU     23) SIGURG      24) SIGXCPU    25) SIGXFSZ
 26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
 31) SIGSYS     34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
 38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
 43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
 53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
 58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
 63) SIGRTMAX-1  64) SIGRTMAX
```

Among all these signals we usually trap SIGINT, SIGKILL signals.

**Note:** In single quotes if we don't give any commands then ***trap*** command will not allow the shell to pass those signals to that particular process.

### Lab Exercises:

1. Script for the usage of **eval** command.

```
$vim script63.sh
#!/bin/bash
a='10'
b='$a'
echo "Without using eval command"
echo '$b='$b
echo "With eval command"
eval echo '$b='$b
```

#### Output

```
[root@mlinux7 scripts]# chmod +x script63.sh
[root@mlinux7 scripts]# ./script63.sh
Without using eval command
$b=$a
With eval command
$a=10
```

2. Usage of xargs command.

```
$vim script64.sh
#!/bin/bash
### Usage of xargs command
$echo -e "1\n2\n3\n4\n5"
$echo -e "1\n2\n3\n4\n5"
Using xargs command we can convert this column into row
$echo -e "1\n2\n3\n4\n5" |xargs
$echo -e "1 \n2\n3\n4\n5" |xargs
We can specify number of columns also in xargs using -n option
$echo -e "1\n2\n3\n4\n5" |xargs -n 2
$echo -e "1\n2\n3\n4\n5" |xargs -n 2
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script64.sh
[root@mlinux7 scripts]# ./script64.sh
$echo -e "1\n2\n3\n4\n5"
1
2
3
4
5
Using xargs command we can convert this column into row
$echo -e "1\n2\n3\n4\n5" |xargs
1 2 3 4 5
We can specify number of columns also in xargs using -n option
$echo -e "1\n2\n3\n4\n5" |xargs -n 2
1 2
3 4
5
```

### 3. Usage of *expr* command

```
$vim script65.sh
#!/bin/bash
### expr command explained.
echo "To compare two words and get a count that how many characters are matched"
echo '$ expr Techno : Tech'
expr Techno : Tech
echo "We can compare two strings and get the status 0= Failure, 1=Success"
echo '$ expr 10 = 20'
expr 10 = 20
echo '$ expr 10 != 20'
expr 10 != 20
echo "Doing a numerical operations"
echo '$ expr 10 / 2'
echo 10 / 5
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script65.sh
[root@mlinux7 scripts]# ./script65.sh
To compare two words and get a count that how many characters are matched
$ expr Techno : Tech
4
We can compare two strings and get the status 0= Failure, 1=Success
$ expr 10 = 20
0
$ expr 10 != 20
1
Doing a numerical operations
$ expr 10 / 2
10 / 5
```

TECHNO SOLUTIONS

### 4. Basic usage of the trap command.

```
$ vim script67.sh
#!/bin/bash
### Script to use trap command and disable the CTRL + C keys.
echo "Try stopping the script using ctrl+c"
trap '' 2
sleep 10
echo "!!!Task Completed, now exiting!!!"
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script67.sh
[root@mlinux7 scripts]# ./script67.sh
Try stopping the script using ctrl+c
^C^C^C
!!!Task Completed, now exiting!!!
```

## 5. Usage of trap command using functions.

\$vim script68.sh

```
#!/bin/bash
## trap using functions.
trapf() {
    echo "You pressed CTRL-C"
    echo "Removing temporary files if used"
    sleep 1
    ### Execute any command(s) if you want to do anything before exiting
    exit
}
trap 'trapf' 2
sleep 10
echo "Task Completed Exiting"
```

### Output:

```
[root@mlinux7 scripts]# chmod +x script68.sh
[root@mlinux7 scripts]# ./script68.sh
^CYou pressed CTRL-C
Removing temporary files if used
```



## QUOTES

### Wild Cards:

Wildcards are a shell feature that makes the command line much more powerful than any GUI file managers. If you want to select a big group of files in a graphical file manager, you usually have to select them with your mouse. This may seem simple, but in some cases it can be very frustrating. For, example, you have a directory with a huge amount of all kinds of files and subdirectories, and you have decided to move all the LOG files, which have the word "Linux" somewhere in the middle of their names, from that big directory into another directory. What's a simple way to do this? If the directory contains a huge amount of differently named LOG files, your task is everything but simple!

In the Linux CLI that task is just as simple to perform as moving only one LOG file, and it's so easy because of the shell wildcards. Wildcards are special characters that allow you to select filenames that match certain patterns of characters. This helps you to select even a big group of files with typing just a few characters, and in most cases it's easier than selecting the files with a mouse.

Here's a list of the most commonly used wildcards in bash:

Wild Card	Description	Example
*	Zero or More Characters	ls *.sh, ls log*.log
?	Exactly only one character	ls file? .txt
-	Range of Characters	ls file[1-3].txt
[]	Enclose a set of characters	ls file[1,2,3].txt
~	Denotes user home path	cd ~/Desktop
{}	exactly one entire word in the options given	ls {file* .txt,log* .log}

In wild cards we looked at shell substitution, which occurs automatically whenever you enter a command containing a wildcard character or a \$ parameter. The way the shell interprets these and other special characters is generally useful, but sometimes it is necessary to turn off shell substitution and let each character stand for itself. Usually we use echo command to quote the text and to turn off the special meaning of a character is called quoting, and it can be done in three ways:

- Using Backslash
- Using Single Quotes
- Using Double Quotes

### Meta Characters:

Here is a list of most of the shell special characters also called Meta characters.

\* ? [ ] ' " \ \$ ; & ( ) | ^ < > new-line space tab

### Using Backslash:

By using, backslash we can quote or nullify the specialty of only one character at a time. We can prevent the **echo** command or shell from interpreting a character by placing a backslash ("\\") in front of a special character.

```
[root@mlinux7 ~]# echo Hello ; World
Hello
bash: World: command not found...
[root@mlinux7 ~]# echo Hello \; World
Hello ; World
```

In the above example to nullify the specialty of the **semicolon (;**) character we used backslash.

```
[root@mlinux7 ~]# echo You owe me $1500
You owe me 500
[root@mlinux7 ~]# echo You owe me \$1500
You owe me $1500
```

In the above example we expected the output as **\$1500** which shown as **500** because of having dollar (**\$1**) special character initialized by shell as a variable. So after quoting it using backslash the output is as expected.

### Using Single Quotes:

In single quotes all the characters including the special characters are treated as normal characters. So characters within single quotes are quoted just as if a backslash is in front of each character.

```
[root@mlinux7 ~]# echo <-$1500.**>; (update?) [y/n]
-bash: syntax error near unexpected token `;'
[root@mlinux7 ~]# echo \<-$1500.\*\*\>; \ (update\?\) \ [y\|n\]
<-$1500.**>; (update?) [y\|n]
[root@mlinux7 ~]# echo '<-$1500.**>; (update?) [y/n]' 
<-$1500.**>; (update?) [y/n]
```

### Using Double Quotes:

The double quote ( “ ” ) protects everything enclosed between two double quote marks except \$, `, " and \. Use the double quotes when you want only variables and command substitution.

```
[root@mlinux7 ~]# echo "$SHELL"
/bin/bash
[root@mlinux7 ~]# echo "/etc/*.*conf"
/etc/*.*conf
[root@mlinux7 ~]# echo "Today is $(date)"
Today is Thu Oct  5 19:46:14 IST 2017
[root@mlinux7 ~]# echo "$1=Rs 60"
=Rs 60
[root@mlinux7 ~]# echo "\$1=Rs 60"
$1=Rs 60
```

The comparisons table for all the three quotes is as follows.

S.No	Parameter	Backslash	Single Quote	Double Quote
1.	Variable	Use Backslash to change the meaning of the characters to escape special characters.	No	Yes
2.	Wild Card		No	No
3.	Command Substitute		No	Yes

### Quoting Rules and Situations:

Following are the additional options that help in quoting for various scenario.

S.No	Situation	Example	
1.	Quoting Ignores Word Boundaries	echo Hello ; World	
2.	Combining Quoting in Commands	echo "\$USER" owe me '\$10'	
3.	Quoting Newlines to Continue on the Next Line	echo 'Line1 >Line2'	echo Line1 \ >Line2
4.	Embedding Spaces in a Single Argument	Echo 'Hi Bye'	
5.	Quoting to Access Filenames Containing Special Characters	rmdir New\ Folder	
6.	Quoting the Backslash to Enable echo Escape Sequences	echo -e "Line1\nline2"	

### Lab Exercises:



- Script to read password from the users.

```
$vim script14.sh
#!/bin/bash
# -s option is used to read a password without displaying on the screen.
read -p 'Pl Enter your Name: ' name
read -s -p 'Pl Enter your Password: ' pass
echo -e "\nYour Name= $name\nYour Password= $pass"
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script14.sh
[root@mlinux7 scripts]# ./script14.sh
Pl Enter your Name: Musab
Pl Enter your Password:
Your Name= Musab
Your Password= abcd
```

2. Script to define two values to do basic numerical operations.

```
$vim script15.sh
#!/bin/bash
## Script to do numerical operation using variables .
a=20
b=10
ADD=$((a+b))
SUB=$((a-b))
MUL=$((a*b))
DIV=$((a/b))
echo -e "Addition=$ADD\nSubtraction=$SUB\nMultiply=$MUL\nDivide=$DIV"
```

Output:

```
[root@mlinux7 scripts]# chmod +x script15.sh
[root@mlinux7 scripts]# ./script15.sh
Addition=30
Subtraction=10
Multiply=200
Divide=2
```

3. Script to read the input values from the user and do numerical operations for the given values.

```
$vim script16.sh
#!/bin/bash
##Script to do numerical operations using input from user
echo "Please Enter the inputs to continue."
read -p 'Enter the 1st value: ' a
read -p 'Enter the 2nd value: ' b
ADD=$((a+b))
SUB=$((a-b))
MUL=$((a*b))
DIV=$((a/b))
echo -e "Addition=$ADD\nSubtraction=$SUB\nMultiply=$MUL\nDivide=$DIV"
```

Output:

```
[root@mlinux7 scripts]# chmod +x script16.sh
[root@mlinux7 scripts]# ./script16.sh
Please Enter the inputs to continue.
Enter the 1st value: 10
Enter the 2nd value: 5
Addition=15
Subtraction=5
Multiply=50
Divide=2
```

4. Script for the basic usage of pipes to get one command output and the same given as input to another command.

```
$vim script17.sh
#!/bin/bash
### Using Pipes instead of STDIN
who |wc -l ### To get how many users logged into server
echo "Body Message" |mail -s "Hey!!" admin@vpts.com
## Send an email using the echo command message as body message.
```

Output:

```
[root@mlinux7 scripts]# chmod +x script17.sh
[root@mlinux7 scripts]# ./script17.sh
2
```

5. Script to show the usage of basic calculator.

```
$vim script18.sh
#!/bin/bash
### bc is basic calculator in Linux
echo 1 + 2 |bc
echo 1/2 |bc
echo "scale=3; 1/2" |bc ## Using scale to print decimal values
```

Output:

```
[root@mlinux7 scripts]# chmod +x script18.sh
[root@mlinux7 scripts]# ./script18.sh
3
0
.500
```

6. Script to do numerical operations using **bc** calculator and also read the input from the user.

```
#!/bin/bash
##Script to do numerical operations using bc calculator.
echo "Please Enter the inputs to continue."
read -p 'Enter the 1st value: ' a
read -p 'Enter the 2nd value: ' b
ADD=`echo $a+$b|bc`
SUB=`echo $a-$b|bc`
MUL=`echo $a*$b|bc`
DIV=`echo "scale=5;$a/$b" |bc`
echo -e "Addition=$ADD\nSubtraction=$SUB\nMultiply=$MUL\nDivide=$DIV"
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script19.sh
[root@mlinux7 scripts]# ./script19.sh
Please Enter the inputs to continue.
Enter the 1st value: 10
Enter the 2nd value: 20
Addition=30
Subtraction=-10
Multiply=200
Divide=.50000
```

## INPUT/ OUTPUT:

### Redirectors:

- The shell and many UNIX commands take their input from standard input (STDIN), write output to standard output (STDOUT), and write error output to standard error (STDERR). By default, standard input is connected to the terminal keyboard and standard output and error to the terminal screen.
- Redirection of I/O to a file, is accomplished by specifying the destination on the command line using a redirection Meta characters followed by the desired destination.
- The BASH uses a format for redirection which includes numbers. The numbers refers to the file descriptor numbers (0 standard input, 1 standard output, 2 standard error).

Characters	Actions
>	Redirect standard output
2>	Redirect standard error
2>&1	Redirect standard error to standard output
<	Redirect standard input
	Pipe standard output to another command
>>	Appends to standard output
2>&1	Pipe standard output and standard error to another command

Example	Details
\$ls >list.out	Redirects the output of ls command to a file list.out Note: If there is a file list.out this particular command removes the old data and add the out of the latest command.
\$ls 2>list.err	Redirects the STDERR to list.err file. Whereas the normal output will be printed on the screen.
\$ls 2>&1 >list.out	Redirects both STDOUT and STDERR > to list.out file
\$ls >>list.out	Appends output of ls command to list.out file. It will append the latest data to the file and wont erases the earlier data.
mail -s TEST myuser@vpts.com <list.out	This command will Send an email to the user but whereas the body message will be taken from a file list.out instead of STDIN from keyboard.

### Pipes:

A pipe is used to connect output of one program or command to the input of other command/program. It is symbolized by “|”

```
[root@mlinux7 ~]# cat list.err |wc -l
[root@mlinux7 ~]# cat list.err |sort
```

### Filters:

Shell scripts are often called on to manipulate and reformat the output from commands that they execute. Sometimes this task is as simple as displaying only part of the output by filtering out certain lines. In most instances, the processing required is much more sophisticated. Now we see few basic text filtering commands with example.

### Head:

**head** is a program on UNIX and Unix-like systems used to display the first few lines of a text file or piped data. By default it will show the first 10 lines of a file.

Syntax: **head [-number | -n number] filename**

```
$ head list.out
$ head -n 20 list.out
$ head -20 list.out
```

### Tail:

**Tail** is a program on UNIX and Unix-like systems used to display the last few lines of a text file or piped data. By default it will show the last 10 lines of a file.

Syntax: **tail [options] filename**

```
$ tail -n 5 /etc/passwd      ### Prints the last five lines of the file
$ tail -f /var/log/messages  ### To see the continuous output of appended lines
```

### Grep:

The word grep stands for globally regular expression print. The **grep** command allows you to search one file or multiple files for lines that contain a pattern.

Syntax: **grep [options] pattern [files]**

The most regular options we use in grep command are as follows

Option	Description
<b>-b</b>	Display the block number at the beginning of each line.
<b>-c</b>	Display the number of matched lines.
<b>-h</b>	Display the matched lines, but do not display the filenames.
<b>-i</b>	Ignore case sensitivity.
<b>-l</b>	Display the filenames, but do not display the matched lines.
<b>-n</b>	Display the matched lines and their line numbers.
<b>-s</b>	Silent mode.
<b>-v</b>	Display all lines that do NOT match.
<b>-w</b>	Match whole word.

The following are the examples for *grep* command with different options.

To search for a word **root** in */etc/passwd* file

```
[root@mlinux7 ~]# grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

To find the block number of search result pattern, it will list the lines of pattern found along with the block number of the line.

```
[root@mlinux7 ~]# grep -b root /etc/passwd
0:root:x:0:0:root:/root:/bin/bash
340:operator:x:11:0:operator:/root:/sbin/nologin
```

To count the number of lines pattern was found in search result

```
[root@mlinux7 ~]# grep -c root /etc/passwd
2
```

While searching for the multiple files for the same pattern it will display the lines along with the file names. For example we are searching for a pattern **root** from files */etc/passwd* & */etc/shadow*.

```
# grep root /etc/passwd /etc/shadow
/etc/passwd:root:x:0:0:root:/root:/bin/bash
/etc/passwd:operator:x:11:0:operator:/root:/sbin/nologin
/etc/shadow:root:$6$AvTUk/9qKoMKOWin$hdxz5DZ5vGBvdLsedA0Bvb41b8tU7EAuqmWvY5uXLfsbRrSGxcJWwF0xCA7h1xA:::0:99999:7:::
```

Same thing without displaying file names.

```
[root@mlinux7 ~]# grep -h root /etc/passwd /etc/shadow
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
root:$6$AvTUk/9qKoMKOWin$hdxz5DZ5vGBvdLsedA0BvR6BNaaIDHDmWvY5uXLfsbRrSGxcJWwF0xCA7h1xA:::0:99999:7:::
```

Displaying only filenames without the search strings.

```
[root@mlinux7 ~]# grep -l root /etc/passwd /etc/shadow
/etc/passwd
/etc/shadow
```

Displaying all the lines except a particular word/string in the files.

```
[root@mlinux7 ~]# grep -v root /etc/passwd
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

Highlighting the words searched by displaying in colors in the output

```
[root@mlinux7 ~]# grep --color root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

Grep can be combined with other commands by taking input from them.

```
#cat /etc/passwd |grep root
#lvdisplay |grep mylv
```

### Cut Command:

The cut command in UNIX (or Linux) is used to select sections of text from each line of files. You can use the cut command to select fields or columns from a line by specifying a delimiter or you can select a portion of the text by specifying the range or characters. Basically the cut command slices a line and extracts the text.

The most commonly used options in cut command are as follows.

Option	Description
-c	Cuts the input file using list of characters specified by this option
-f	Cuts the input file using list of field. The default field to be used TAB.
-d	Specifies a delimiter to use as a field. As mentioned previously default field is TAB and this option overwrites this default behavior.

In the following examples we would be using following file

```
[root@mlinux7 ~]# cat cut.txt
Welcome to VPTS
```

Example for using -c option

```
[root@mlinux7 ~]# cut -c 2 cut.txt
e
[root@mlinux7 ~]# cut -c -3 cut.txt
Wel
[root@mlinux7 ~]# cut -c 1-3 cut.txt
Wel
[root@mlinux7 ~]# cut -c 4- cut.txt
come to VPTS
```

Example to use -d option

```
[root@mlinux7 ~]# cat cut.txt
Welcome:to:VPTS
[root@mlinux7 ~]# cut -d ":" -f2 cut.txt
to
[root@mlinux7 ~]# cut -d ":" -f-2 cut.txt
Welcome:to
```

Example for **-f** option with **-d**

```
[root@mlinux7 ~]# cat cut.txt
Welcome to VPTS
[root@mlinux7 ~]# cut -d " " -f1 cut.txt
Welcome
[root@mlinux7 ~]# cut -d " " -f3 cut.txt
VPTS
[root@mlinux7 ~]# cut -d " " -f1,3 cut.txt
Welcome VPTS
[root@mlinux7 ~]# cut -d " " -f-2 cut.txt
Welcome to
[root@mlinux7 ~]# cut -d " " -f2- cut.txt
to VPTS
```

### Exit Status:

Any program completes execution under the UNIX or Linux system, it returns a status back to the system. This status is a number that usually indicates whether the program successfully ran or not, and hence called as exit status. By convention, an exit status of zero indicates that a program succeeded, and non-zero (1-255) indicates that it failed. Failures can be caused by invalid arguments passed to the program, or by an error condition detected by the program.

The shell variable **\$?** is automatically set by the shell to the exit status of the last command executed. Naturally, you can use **echo** to display its value at the terminal.

```
[root@mlinux7 ~]# cp myfile new_myfile
[root@mlinux7 ~]# echo $?
0
[root@mlinux7 ~]# cp vpts new_vpts
cp: cannot stat 'vpts': No such file or directory
[root@mlinux7 ~]# echo $?
1
```

Note that the numeric result of a "failure" for some commands can vary from one UNIX version to the next, but success is always signified by a zero exit status.

The following are the few exit status which will be generated regularly while executing the scripts or commands.

Exit Code	Description
<b>1</b>	General Input Error
<b>2</b>	Misuse of Command and very rare
<b>126</b>	Cannot invoke the requested command, For example executing a script which doesn't have execute command
<b>127</b>	Command not found error
<b>128 + n</b>	Fatal error signal "n" (for example-'kill -9 = 137).
<b>130</b>	Script terminated by Ctrl-C

In shell scripting we can make our scripts generic using the exit codes. We can exit our script using exit command followed by a number.

Following examples narrates Exit States:

```
#!/bin/bash
exit 1 ##Mention the value to exit with the same status.
### To exit with the value 1.

#!/bin/bash
echo VPTS
exit 0 ### To exit with the exit value 0.
```



## AWK COMMAND

**Awk** is a powerful language to manipulate and process text files. It is especially helpful when the lines in a text files are in a record format, i.e., when each line (record) contains multiple fields separated by a delimiter.

Even when the input file is not in a record format, you can still use awk to do some basic file and data processing. You can also write programming logic using awk even when there are no input files that needs to be processed. In short, AWK is a powerful language that can come in handy to do daily routine Job.

**The *employee.txt* sample file:**

The *employee.txt* is a comma delimited file that contains 5 employee records in the following format

**employee-number, employee-name, employee-title**

We will be using the following file in below examples.

```
[root@mlinux7 ~]# cat employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```



**Basic Awk Syntax:**

```
awk -Fs '/pattern/ {action}' input-file
(or)
awk -Fs '{action}' intput-file
```

In the above syntax:

- -Fs is the field separator. If you don't specify it will use an empty space as field delimiter.
- The /pattern/ and the {action} should be enclosed inside single quotes.
- /pattern/ is optional. If you don't provide it, awk will process all the records from the input-file. If you specify a pattern, it will process only those records from the input-file that match the given pattern.
- {action} - These are the awk programming commands, which can be one or multiple awk commands. The whole action block (including all the awk commands together) should be closed between {and}
- Input-file - The input file that needs to be processed.

## Awk program structure (BEGIN, body, END block).

### 1. BEGIN Block

Syntax for BEGIN block:

**BEGIN { awk commands }**

The begin Block gets executed only once at the beginning, before awk starts executing the body block for all the lines in the input file.

- The begin block is a good place to print report headers, and initialize variables.
- You can have one or more awk commands in the begin block.
- The keyword BEGIN should be specified in upper case.
- Begin block is optional.

### 2. Body Block

Syntax for body block:

**/pattern/ {action}**

The body block gets executed once for every line in the input file.

- If the input file has 10 records, the commands in the body block will be executed 10 times (once for each record in the input file).
- There is no keyword for the body block. We discussed pattern and action previously.

### 3. END Block

Syntax of end block:

**END { awk-commands }**

The end block gets executed only once at the end, after awk completes executing the body block for all the lines in the input-file.

- The end block is a good place to print report footer and do any clean-up activities.
- You can have one or more awk commands in the end block.
- The keyword END should be specified in upper case.

### Examples of Awk Command.

#### 1. To filter the columns in the employee.txt file using a delimiter.

```
[root@mlinux7 ~]# awk -F, '{print $1}' employee.txt
101
102
103
104
105
[root@mlinux7 ~]# awk -F, '{print $2}' employee.txt
John Doe
Jason Smith
Raj Reddy
Anand Ram
Jane Miller
```

**Note:** By default the delimiter taken by awk is either **tab space or a single space**, To specify custom delimiter we have to use **-F** option.

2. To filter the columns along with enabling search as well.

```
[root@mlinux7 ~]# awk -F, '/Manager/ {print $2}' employee.txt
Jason Smith
Jane Miller
```

In the above example along with the filtering column we are also filtering based on text

3. To use the **BEGIN** Command to print the headers.

```
[root@mlinux7 ~]# awk 'BEGIN { FS=",";print "--Employee Name--"} /Manager/ {print $2}' employee.txt
--Employee Name--
Jason Smith
Jane Miller
```

4. To use **END** Command to print the footers.

```
[root@mlinux7 ~]# awk -F, '/Manager/ {print $2} END { print "---EOF---"}' employee.txt
Jason Smith
Jane Miller
---EOF---
```

5. To use **BEGIN** and **END** blocks in a single awk command.

```
[root@mlinux7 ~]# awk -F, 'BEGIN { print "EMP_ID    EMP_NAME    EMP_DESG" } {print $1" "$2" "$3}
END { print "---EOF---"}' employee.txt
EMP_ID    EMP_NAME    EMP_DESG
101      John Doe    CEO
102      Jason Smith  IT Manager
103      Raj Reddy   Sysadmin
104      Anand Ram   Developer
105      Jane Miller  Sales Manager
---EOF---
```

6. To print or read a file using awk.

```
[root@mlinux7 ~]# awk '{print}' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

7. To see the usage of AWK Built-In Filed Separator (FS).

```
[root@mlinux7 ~]# awk -F, 'BEGIN {FS=","}{ print $1,$2}' employee.txt
101 John Doe
102 Jason Smith
103 Raj Reddy
104 Anand Ram
105 Jane Miller
```

**8. To see the usage of Output Field Separator (OFS).**

OFS is for input field separator. OFS is for output field separator. OFS is printed between consecutive fields in the output. By default, awk prints the output fields with space between the fields.

```
[root@mlinux7 ~]# awk -F ',' '{print $2, ":" $3}' employee.txt
John Doe :CEO
Jason Smith :IT Manager
Raj Reddy :Sysadmin
Anand Ram :Developer
Jane Miller :Sales Manager
[root@mlinux7 ~]#
[root@mlinux7 ~]# awk -F ',' 'BEGIN { OFS=":"}{ print $2, $3}' employee.txt
John Doe:CEO
Jason Smith:IT Manager
Raj Reddy:Sysadmin
Anand Ram:Developer
Jane Miller:Sales Manager
```

**9. To see the Record Separator (RS) usage.**

```
[root@mlinux7 ~]# cat employee-one-line.txt
101,John Doe:102,Jason Smith:103,Raj Reddy:104,Aslam khan:105,David Miller
[root@mlinux7 ~]#
[root@mlinux7 ~]# awk -F , 'BEGIN { RS=":" } { print $0 }' employee-one-line.txt
101,John Doe
102,Jason Smith
103,Raj Reddy
104,Aslam khan
105,David Miller
```

**10. Combining all IFS, OFS, RS usage.**

```
[root@mlinux7 ~]# cat employee-change-fs-ofs.txt
101
John Doe
CEO
-
102
Jason Smith
IT Manager
-
103
Raj Reddy
Sysadmin
104
Aslam khan
Developer
-
105
David Miller
Sales Manager
[root@mlinux7 ~]# awk 'BEGIN { FS="\n"; RS="-\n"; OFS=":" }{print $2, $3}' employee-change-fs-ofs.txt
John Doe:CEO
Jason Smith:IT Manager
Raj Reddy:Sysadmin
David Miller:Sales Manager
```

**11.** To check the number of records

- **NR** is very helpful. When used inside the loop, this gives the line number. When used in the **END block**, this gives the total number of records in the file.
- Even though **NR** stands for "**Number of Records**", it might be appropriate to call this as "**Number of the Record**", as it actually gives you the line number of the current record.

```
[root@mlinux7 ~]# awk 'BEGIN {FS=",")} {print "Emp Id of record number",NR,"is",$1;} END {print "Total number of records:",NR}' employee.txt
Emp Id of record number 1 is 101
Emp Id of record number 2 is 102
Emp Id of record number 3 is 103
Emp Id of record number 4 is 104
Emp Id of record number 5 is 105
Total number of records: 5
```

**12.** To see the Auto-Increment and Auto-Decrement using Unary Operators.

```
[root@mlinux7 ~]# cat employee-sal.txt
101,John Doe,CEO,85000
102,Jason Smith,IT Manager,50000
103,Raj Reddy,Sysadmin,30000
104,Anand Ram,Developer,30000
105,Jane Miller,Sales Manager,25000
[root@mlinux7 ~]# awk -F, '{print ++$4}' employee-sal.txt
85001
50001
30001
30001
25001
[root@mlinux7 ~]# awk -F, '{print --$4}' employee-sal.txt
84999
49999
29999
29999
24999
```



**13.** To print the last column of a file or last second column.

```
[root@mlinux7 ~]# awk -F, '{print $NF}' employee.txt
CEO
IT Manager
Sysadmin
Developer
Sales Manager
[root@mlinux7 ~]#
[root@mlinux7 ~]# awk -F, '{print $(NF-1)}' employee.txt
John Doe
Jason Smith
Raj Reddy
Anand Ram
Jane Miller
```

## SED COMMAND

**SED** stands for **Stream Editor**. It is very powerful tool to manipulate, filter, and transform text. Sed can take input from a file, or from a pipe. You might even have several sed one line commands in your bash startup file that you use for various scenarios without exactly understanding the sed scripts. For beginners, sed script might look cryptic. Once you understand the sed commands in detail, you'll be able to solve a lot of complex text manipulation problems by writing a quick sed script.

### **SED Command Syntax:**

The following is the basic SED command syntax.

```
sed {options} {sed-commands} {input-file}
```

- Sed reads one line at a time from the {input-file} and executes the {sed-commands} on that particular line.
- It reads the 1st line from the {input-file} and executes the {sed-commands} on the 1st line. Then it reads the 2nd line from the {input-file} and executes the {sed-commands} on the 2nd line. Sed repeats this process until it reaches the end of the {input-file}.
- There are also a few optional command line options that can be passed to sed as indicated by [options].

For all sed examples, we will be using the following employee.txt file. Please create this text file to try out the commands given in this book.

```
[root@mlinux7 ~]# cat employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

The above employee database contains the following fields for every record:

- Employee Id
- Employee Name
- Title

Let's see the different ways of editing the file using SED editor with different options.

### **SED Print Command (p):**

The syntax for printing lines in sed command is as follows.

```
sed 'p' file
```

7. To print a file we use 'p' command in sed.

```
[root@mlinux7 ~]# sed p employee.txt
101,John Doe,CEO
101,John Doe,CEO
102,Jason Smith,IT Manager
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
107,Tedd Stones,Practice Director
```

In the above you can see the double lines of output. That is due to the sed work behavior the first entry is due to the backup of the line and whereas the other line is the exact output.

8. To print a file without any duplicate lines we use '-n' option in print command of sed.

```
[root@mlinux7 ~]# sed -n 'p' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

9. To search a particular string in the file. Ex., jane in employee.txt file

```
[root@mlinux7 ~]# sed -n '/Jane/p' employee.txt
105,Jane Miller,Sales Manager
```

10. To search multiple expressions and print the matched lines we have to use '-e' option in sed command.

```
[root@mlinux7 ~]# sed -n -e '/Jane/ p' -e '/Stuart/ p' employee.txt
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
```

11. To execute multiple commands from a file we have to use -f option.

```
[root@mlinux7 ~]# cat mycomm.sed
/Jane/ p
/Stuart/ p
[root@mlinux7 ~]# sed -n -f mycomm.sed employee.txt
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
```

In the above example we are passing multiple commands which are listed in mycomm.sed file. Usually we use this option in such an environment where we need commonly used options each and every time, then we can go with creating a file with all the option and use that particular file in the sed command.

12. To print only 2<sup>nd</sup> line of a file.

```
[root@mlinux7 ~]#
[root@mlinux7 ~]# sed -n '2 p' employee.txt
102,Jason Smith,IT Manager
```

In the above example we are specifying before print that to print only the 2<sup>nd</sup> line

13. To print from line 1 through line 4.

```
[root@mlinux7 ~]# sed -n '1,4 p' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

14. To print from line 3 through the last line.

```
[root@mlinux7 ~]# sed -n '3,$ p' employee.txt
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

15. To print only odd numbered lines

```
[root@mlinux7 ~]# sed -n '1~2 p' employee.txt
101,John Doe,CEO
103,Raj Reddy,Sysadmin
105,Jane Miller,Sales Manager
107,Tedd Stones,Practice Director
```

16. To print lines starting from the 1st match of "Jason" until the 4th line.

```
[root@mlinux7 ~]# sed -n '/Jason/,4 p' employee.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

17. To print lines starting from the 1st match of "Raj" until the last line.

```
[root@mlinux7 ~]# sed -n '/Raj/, $ p' employee.txt
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

18. To print lines starting from the line matching "Raj" until the line matching "Jane"

```
[root@mlinux7 ~]# sed -n '/Raj/, /Jane/ p' employee.txt
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
```

19. To print the line matching "Jason" and 2 lines immediately after that.

```
[root@mlinux7 ~]# sed -n '/Jason/, +2 p' employee.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

#### SED Delete Command (d):

The syntax for deleting lines in sed command is as follows.

```
sed 'd' file
```

The following are the different ways of using the delete (d) command.

1. To delete only the 2nd line from a file.

```
[root@mlinux7 ~]# sed '2 d' employee.txt
101,John Doe,CEO
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

2. To delete from line 1 to 4

```
[root@mlinux7 ~]# sed '1,4 d' employee.txt
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

3. To delete from line 2 through last line.

```
[root@mlinux7 ~]# sed '2,$ d' employee.txt
101,John Doe,CEO
```

4. To delete only odd numbered lines.

```
[root@mlinux7 ~]# sed '1~2 d' employee.txt
102,Jason Smith,IT Manager
104,Anand Ram,Developer
106,Stuart Gant,HR Manager
```

5. To delete lines matching the pattern Manager.

```
[root@mlinux7 ~]# sed '/Manager/ d' employee.txt
101,John Doe,CEO
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
107,Tedd Stones,Practice Director
```

6. To delete lines starting from the 1st match of Jason until the 4th line.

```
[root@mlinux7 ~]# sed '/Jason/,4 d' employee.txt
101,John Doe,CEO
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

7. To delete lines starting from the 1st match of "Raj" until the last line.

```
[root@mlinux7 ~]# sed '/Raj/,$ d' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
```

8. To delete lines starting from the line matching "Raj" until the line matching.

```
[root@mlinux7 ~]# sed '/Raj/,/Jane/ d' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

9. To delete lines starting from the line matching "Jason" and 2 lines immediately after that.

```
[root@mlinux7 ~]# sed '/Jason/,+2 d' employee.txt
101,John Doe,CEO
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

**10. To delete all empty lines in a file**

```
[root@mlinux7 ~]# sed '/^$/ d' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

**Write Command (W):**

**1. To Write the content of employee.txt file to file output.txt**

```
[root@mlinux7 ~]# sed 'w output.txt' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
[root@mlinux7 ~]#
[root@mlinux7 ~]# cat output.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```



In the above example you can observe that the content on employee.txt file is copied to a new file output.txt and also you can see the duplicate lines while writing to other file. We can use -n option to not print the original lines on the screen. Usually in general people use Print command to print the content and use redirectors to write in a new file.

**2. To Write the content of sed.txt file to output.txt-file without displaying the output**

```
# sed -n w output.txt sed .txt
```

**3. To write only the 2<sup>nd</sup> line to output.txt file**

```
[root@mlinux7 ~]# sed -n '2 w output.txt' employee.txt
[root@mlinux7 ~]#
[root@mlinux7 ~]# cat output.txt
102,Jason Smith,IT Manager
```

4. To write lines from 1 to 4

```
[root@mlinux7 ~]# sed -n '1,4 w output.txt' employee.txt
[root@mlinux7 ~]#
[root@mlinux7 ~]# cat output.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

5. To write lines from 2 till the last line

```
[root@mlinux7 ~]# sed -n '2,$ w output.txt' employee.txt
[root@mlinux7 ~]#
[root@mlinux7 ~]# cat output.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

6. To Write only odd numbered lines.

```
[root@mlinux7 ~]# sed -n '1~2 w output.txt' employee.txt
[root@mlinux7 ~]#
[root@mlinux7 ~]# cat output.txt
101,John Doe,CEO
103,Raj Reddy,Sysadmin
105,Jane Miller,Sales Manager
107,Tedd Stones,Practice Director
```

7. Write lines matching the pattern "Jane"

```
[root@mlinux7 ~]# sed -n '/Jane/ w output.txt' employee.txt
[root@mlinux7 ~]#
[root@mlinux7 ~]# cat output.txt
105,Jane Miller,Sales Manager
```

8. To write lines starting from the 1st match of "Jason" until the 4<sup>th</sup> line.

```
[root@mlinux7 ~]# sed -n '/Jason/,4 w output.txt' employee.txt
[root@mlinux7 ~]#
[root@mlinux7 ~]# cat output.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

9. To write the line matching “Jason” and the next 2 lines immediately after that.

```
[root@mlinux7 ~]# sed -n '/Jason/,+2 w output.txt' employee.txt
[root@mlinux7 ~]#
[root@mlinux7 ~]# cat output.txt
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
```

#### Substitute Command (s):

The following is the **sed** substitute command syntax.

```
sed '[address-range|pattern]    s/originalstring/replacement-string/(substitute-flags)'
infile
```

In the above sed substitute command syntax:

- Address-range or pattern -range is optional. If you don't specify one, sed will execute the substitute command on all lines.
- s -tells sed to execute the substitute command
- Original-string - this is the string to be searched for in the input file. The original-string can also be a regular expression.
- Replacement-string- Sed will replace original-string with this string.
- Substitute-flags are optional. More on this in the next section.

1. To replace all occurrences of Manager with Director

```
[root@mlinux7 ~]# sed 's/Manager/Director/' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Director
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Director
106,Stuart Gant,HR Director
107,Tedd Stones,Practice Director
```

2. To Replace Manager with Director only on lines that contains the keyword “Sales”

```
[root@mlinux7 ~]# sed '/Sales/s/Manager/Director/' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Director
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

3. To Replace the 1st occurrence of lower case a with upper case A.

```
[root@mlinux7 ~]# echo apple ant |sed -e 's/a/A/'  
Apple ant  
[root@mlinux7 ~]# echo apple ant |sed -e 's/a/A/g'  
Apple Ant
```

4. To Replace the 2nd occurrence of lower case *a* to upper case *A*.

```
[root@mlinux7 ~]# echo apple ant|sed -e 's/a/A/2'  
apple Ant
```

5. To write only the line that was changed by the substitute command to output.txt.

```
[root@mlinux7 ~]# sed -n 's/John/Johnny/w output.txt' employee.txt  
[root@mlinux7 ~]# cat output.txt  
101,Johnny Doe,CEO
```

6. To replace John with Johnny using *i* command to enable case insensitivity.

```
[root@mlinux7 ~]# sed 's/john/Johnny/' employee.txt  
101,John Doe,CEO  
102,Jason Smith,IT Manager  
103,Raj Reddy,Sysadmin  
104,Anand Ram,Developer  
105,Jane Miller,Sales Manager  
106,Stuart Gant,HR Manager  
107,Tedd Stones,Practice Director  
[root@mlinux7 ~]# sed 's/john/Johnny/i' employee.txt  
101,Johnny Doe,CEO  
102,Jason Smith,IT Manager  
103,Raj Reddy,Sysadmin  
104,Anand Ram,Developer  
105,Jane Miller,Sales Manager  
106,Stuart Gant,HR Manager  
107,Tedd Stones,Practice Director
```



#### Append line after {a command} :

You can insert a new line after specific locations by using the sed append command (a). The following is the syntax.

```
sed '[address] a the -line-to-append' input-file
```

7. To add a new record to the employee.txt file after line number

```
[root@mlinux7 ~]# sed '2 a 203,Rock Johnson,Engineer' employee.txt  
101,John Doe,CEO  
102,Jason Smith,IT Manager  
203,Rock Johnson,Engineer  
103,Raj Reddy,Sysadmin  
104,Anand Ram,Developer  
105,Jane Miller,Sales Manager  
106,Stuart Gant,HR Manager  
107,Tedd Stones,Practice Director
```

8. To add a new record to the end of the employee.txt file.

```
[root@mlinux7 ~]# sed '$ a 108,Rock Johnson,Engineer' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
108,Rock Johnson,Engineer
[root@mlinux7 ~]#
```

#### Insert Line Before (i) Command:

The sed insert command (i) works just like the append command except that it inserts a line before a specific location instead of after the location. The following is the syntax.

```
sed '[address] i the-line-to-insert' input-file
```

1. To insert a new record before line number 2 of the employee.txt file.

```
[root@mlinux7 ~]# sed '2 i 203,Rock Johnson,Engineer' employee.txt
101,John Doe,CEO
203,Rock Johnson,Engineer
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

2. To insert a new record before the last line of the employee.txt file.

```
[root@mlinux7 ~]# sed '$ i 203,Rock Johnson,Engineer' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
203,Rock Johnson,Engineer
107,Tedd Stones,Practice Director
```

- To insert two lines before the line that matches 'Jason'.

```
[root@mlinux7 ~]# sed '/Jason/i\
> 203,Rock Johnson,Engineer\
> 204,Steve Austin,Sales Engineer' employee.txt
101,John Doe,CEO
203,Rock Johnson,Engineer
204,Steve Austin,Sales Engineer
102,Jason Smith,IT Manager
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

### Change Command (c):

The sed change command (c) lets you replace an existing line with new text. The syntax for change command is as follows.

```
sed '[address] c the-line-to-insert' input-file
```

- To delete the record at line number 2 and replace it with a new record

```
[root@mlinux7 ~]# sed '2 c 202,Rock Johnson,Engineer' employee.txt
101,John Doe,CEO
202,Rock Johnson,Engineer
103,Raj Reddy,Sysadmin
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

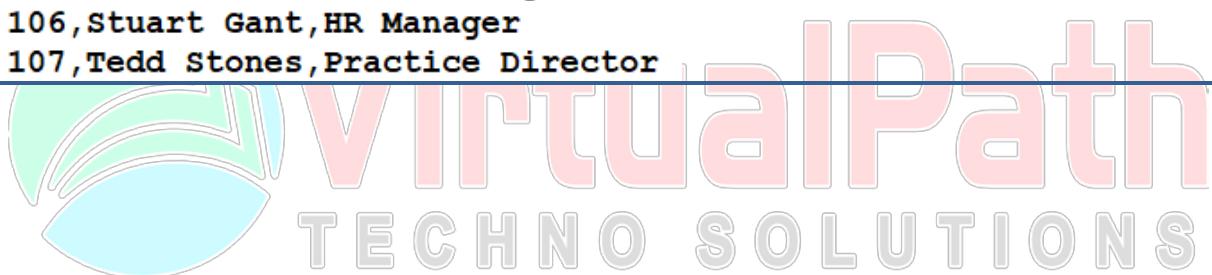
- To delete the line that matches 'Raj' and replaces it with two new lines.

```
[root@mlinux7 ~]# sed '/Raj/c\
> 203,Rock Johnson,Engineer\
> 204,Mark Wahlberg,Sales Engineer' employee.txt
101,John Doe,CEO
102,Jason Smith,IT Manager
203,Rock Johnson,Engineer
204,Mark Wahlberg,Sales Engineer
104,Anand Ram,Developer
105,Jane Miller,Sales Manager
106,Stuart Gant,HR Manager
107,Tedd Stones,Practice Director
```

You can also combine the “a, i, and c” commands. The following sed example does all these three things:

- a - Append 'Rock Johnson' after 'Jason'
- i - Insert 'Mark Wahlberg' before 'Jason'
- c - Change 'Jason' to 'Joe Mason'

```
[root@mlinux7 ~]# sed '/Jason/ {  
> a\  
> 204,Rock Johnson,Engineer  
> i\  
> 202,Mark Wahlberg,Sales Engineer  
> c\  
> 203,Joe Mason, Sysadmin  
> }' employee.txt  
101,John Doe,CEO  
202,Mark Wahlberg,Sales Engineer  
203,Joe Mason, Sysadmin  
204,Rock Johnson,Engineer  
103,Raj Reddy,Sysadmin  
104,Anand Ram,Developer  
105,Jane Miller,Sales Manager  
106,Stuart Gant,HR Manager  
107,Tedd Stones,Practice Director
```



## VARIABLE

A variable in a shell script is a means of referencing a numeric or character value. And unlike formal programming languages, a shell script doesn't require you to declare a type for your variables. Thus, you could assign a number to the variable stuff and then use it again in the same script to hold a string of characters. To access the value (contents) of a variable, prefix it with a dollar (\$) sign. Usually the variables are used to store the path of the files & output of commands.

### **Setting a Variable:**

The shell enables us to set and unset a variable

**Syntax: name=value Ex: NAME=Musab**

- Here 'name' is the name of the variable and 'value' is the data that variable name hold.
- In the above example, **NAME** is the variable and **Musab** is the data named to the variable.
- Variables of this type are called scalar variables, which mean it can hold only one value at a time.

### **Accessing the Variable:**

To access a variable, prefix its name with the dollar sign.

Ex: echo \$NAME

```
[root@mlinux71 ~]# NAME=Musab
[root@mlinux71 ~]#
[root@mlinux71 ~]# echo $NAME
Musab
```

In order to store multiple words or multiple lines in a variable you need to use either single quote or double quote.

```
[root@mlinux71 ~]# NAME="Musab Syed"
[root@mlinux71 ~]# echo $NAME
Musab Syed
[root@mlinux71 ~]# NAME='Musab Syed'
[root@mlinux71 ~]# echo $NAME
Musab Syed
[root@mlinux71 ~]#
```

### **Rules to define a Variable:**

There are certain limitation for defining the name of the variable.

- The name of the variable can contain letters (a to z & A to Z)
- It can have numbers ( 0 to 9 )
- No special characters can be used in the name of a variable other than underscore character “\_”.
- A variable name can either start with letter or underscore and it cannot be started with a number.
- Variable names starting with numbers are used for shell which will be discussed in coming topics.

## Command Substitution:

The Bourne shell can redirect a command's standard output back to the shell's own command line. That is, you can use a command's output as an argument to another command, or you can store the command output in a shell variable. We can do this by enclosing a command in back quotes (`) or in braces ({}).

Syntax: VAR=`COMMAND` OR VAR=\$((COMMAND))

```
[root@mlinux71 ~]# STATUS=`passwd -S myuser`
[root@mlinux71 ~]# echo $STATUS
myuser PS 1969-12-31 0 99999 7 -1 (Password set, SHA512 crypt.)
```

## Arithmetic Substitution:

In ksh and bash, the shell enables integer arithmetic to be performed. This avoids having to run an extra program such as expr or bc to do math in a shell script. This feature is not available in sh.

Syntax: VAR=\$((expression))

```
[root@mlinux71 ~]#
[root@mlinux71 ~]# cal=$(( ((10 + 5*2) -4) /2 ))
[root@mlinux71 ~]# echo $cal
8
```

The following are the operators could be used in the arithmetic substitution.

OPERATOR	DESCRIPTION
+	Addition operator, Add two Numbers
-	Subtraction Operator, Subtract two Numbers
*	Multiplication Operator, Multiply two Numbers
/	Division Operator, Division
()	The parentheses clarify which expressions should be evaluated before others.

Note: Arithmetic Substitution usually won't give the value in decimals, to get the values in decimal way we have to go with other command "bc".

### **Lab Exercises:**

4. Script to
  - a. Declare a variable
  - b. Access the variable
  - c. Unset the variable
  - d. Include the comment for every command.

```
$vim script4.sh
#!/bin/bash
###Script to set and unset a variable.
a=10      #Declaring a variable.
echo $a #Accessing the variable.
unset a #Unsettling the variable.
```

### **Output:**

```
[root@mlinux7 scripts]# chmod +x script4.sh
[root@mlinux7 scripts]# ./script4.sh
10
```

## **ARRAY VARIABLES**

The variable in Bash is a scalar. From Bash 2.0 it can support a type of variable called Array Variable which can handle multiple values at a same time. All the naming rules discussed for Shell Variables would be applicable while naming arrays. Shell does not create a bunch of blank array items to fill the space between indexes. It will keep track of an array index that contains values. In KSH numerical indices for arrays must be between 0 and 1023. In bash there is no limitation. The index supports only integers and we cannot use floating and decimal values. If an array variable with the same name as scalar variable is defined, the value of scalar variable becomes the value of the element of the array at index 0.

### **Syntax:**

```
ArrayName[index]=Value Ex: VAR[x]=value
```

Here, VAR is the variable name

[x] Refer the index value

Value is the data that the array variable would be holding.

### **Example:**

```
Fruits[0]="Apple"
Fruits[1]="Mango"
Fruits[2]="Orange"
Fruits[3]="Banana"
```

Another way with which you can initialize the Array variables are as follows:

**Syntax:**

**ArrayName=(Value1 Value2 Value3 .. ValueN)**

**Example:**

**Fruits=(Apple Mango Orange Banana)**

The Following examples shows us how to use array:

```
#!/bin/bash
#####
# Demo Script for array variables

### Defining an array individually
fruits[0]="Apple"
fruits[1]="Mango"
fruits[2]="Orange"
fruits[3]="Banana"

### Accessing the variable

echo 'echo ${fruits[0]}='${fruits[0]}
echo 'echo ${fruits[1]}='${fruits[1]}
echo 'echo ${fruits[2]}='${fruits[2]}
echo 'echo ${fruits[3]}='${fruits[3]}

### Accessing all values at a time

echo 'echo ${fruits[*]}='${fruits[*]}
echo 'echo ${fruits[@]}='${fruits[@]}

### To get the array size

echo 'echo ${#fruits[*]}='${#fruits[*]}
echo 'echo ${#fruits[@]}='${#fruits[@]}

Output
[root@mlinux7 ~]# ./array.sh
echo ${fruits[0]}=Apple
echo ${fruits[1]}=Mango
echo ${fruits[2]}=Orange
echo ${fruits[3]}=Banana
echo ${fruits[*]}=Apple Mango Orange Banana
echo ${fruits[@]}=Apple Mango Orange Banana
echo ${#fruits[*]}=4
echo ${#fruits[@]}=4
```



**Unset Variable:**

The shell built -in unset command is used for unsetting the data that an initialized variable is holding.

**Syntax: unset <Variable>    Ex: unset name**

The above example will unset the data for the variable name is holding.

### Read-only Variables:

The Shell allows us to make or mark a variable as read-only; once the value is marked as read-only it cannot be changed or manipulated.

**Syntax: readonly <Variable Name> Ex: readonly name**

The above example will set the variable "name" as readonly.

```
[root@mlinux7 ~]# name="VPTS"
[root@mlinux7 ~]# readonly name
[root@mlinux7 ~]# echo $name
VPTS
[root@mlinux7 ~]# unset name
-bash: unset: name: cannot unset: readonly variable
[root@mlinux7 ~]# name=hello
-bash: name: readonly variable
[root@mlinux7 ~]#
```

**Note:** Once the variable is marked as “readonly”, it can't be reversed or undone.

### Environment Variables:

Generally when you create a variable it is only valid on your shell, it is not available for others on different terminal/shell. Environmental variables are such variables which are available globally for everyone who are using the shell, although they may be connected to different session of terminal.

- Some programs need environment variables in order to function correctly.
- Usually a shell script defines only those environment variables that are needed by the programs that it runs.

Normally all our variables are local. Local variable can be used in same shell, if you load another copy of shell (by typing the /bin/bash at the \$ prompt) then new shell will ignore all old shell's variable.

```
[root@mlinux7 ~]# name=Musab
[root@mlinux7 ~]# echo $name
Musab
[root@mlinux7 ~]# /bin/bash
[root@mlinux7 ~]# echo $name
[root@mlinux7 ~]# exit
exit
[root@mlinux7 ~]# echo $name
Musab
[root@mlinux7 ~]#
```

→ *An empty line is printed*

To set an environment variable we have to use the **export** command to use across the environment.

Syntax: `export <Variable Name>`    `export name`

```
[root@mlinux7 ~]# name=Musab
[root@mlinux7 ~]# echo $name
Musab
[root@mlinux7 ~]# export name
[root@mlinux7 ~]# /bin/bash
[root@mlinux7 ~]# echo $name
Musab
[root@mlinux7 ~]# exit
exit
[root@mlinux7 ~]# echo $name
Musab
```

In Bash Shell, with the `set` command we can check whether the variable is set to some value or not. If we want to check only the variables which are exported then `export -p` is the command.

```
[root@mlinux7 ~]# set
[root@mlinux7 ~]# export -p
```

### Lab Exercises:

5. Script to declare an environment variable and try to access them in local shell.

```
$vim script5.sh
#!/bin/bash
## Script to declare environment variables.
MYNAME='VPTS'
export MYNAME
### OR ####
export MYNAME='VPTS'
```

### Output:

```
[root@mlinux7 scripts]# chmod +x script5.sh
[root@mlinux7 scripts]# ./script5.sh
[root@mlinux7 scripts]# echo $MYNAME

[root@mlinux7 scripts]# . script5.sh
[root@mlinux7 scripts]# echo $MYNAME
VPTS
```

**Note:** If you are declaring any environment variables those scripts has to be executed with '`. <Space>`' not with '`. /`'.

6. Define an environment variable in shell and write a script to access the same variable.

```
$vim script6.sh
#!/bin/bash
### Accessing an environmental variable
### defined in a shell
echo $MYNAME1
```

Output:

```
[root@mlinux7 scripts]# chmod +x script6.sh
[root@mlinux7 scripts]# ./script6.sh

[root@mlinux7 scripts]# export MYNAME1=MUSAB
[root@mlinux7 scripts]# ./script6.sh
MUSAB
```

7. Script to print the variable defined in 5<sup>th</sup> example.

```
$vim script7.sh
#!/bin/bash
### Accessing the variable of other script
. script5.sh
echo $MYNAME
```

Output:

```
[root@mlinux7 scripts]# chmod +x script7.sh
[root@mlinux7 scripts]# ./script7.sh
VPTS
```

8. Script to

- a) Declare an array variable.
- b) Access the values of an array individually.
- c) Access all the values at a time.
- d) Get the size of an array.

```
$vim script8.sh
```

```
#!/bin/bash
## Declaring array variables individually.

FRUIT[0]=Banana
FRUIT[1]=Mango
FRUIT[2]=Apple

##Accessing the array variables individually.
echo "###Accessing the array values individually"
echo ${FRUIT[0]}
echo ${FRUIT[1]}
echo ${FRUIT[2]}

## Declaring an array values all at a time.

FRUIT=(Banana Mango Apple)

### Accessing all values in an array at a time.
echo "###Accessing all values in array at a time."
echo ${FRUIT[*]}
### OR ###
echo ${FRUIT[@]}
### Get the size of an array
echo "###Size of an array"
echo ${#FRUIT[@]}
```

Output:

```
[root@mlinux7 scripts]# chmod +x script8.sh
[root@mlinux7 scripts]# ./script8.sh
###Accessing the array values individually
Banana
Mango
Apple
###Accessing all values in array at a time.
Banana Mango Apple
Banana Mango Apple
###Size of an array
3
```

9. Script to declare a readonly variable.

```
$vim script9.sh
```

```
#!/bin/bash
MYVAR="READONLY"
readonly MYVAR ### making a variable as readonly
echo $MYVAR
MYVAR="CHANGE"
echo $MYVAR
```



Output:

```
[root@mlinux7 scripts]# chmod +x script9.sh
[root@mlinux7 scripts]# ./script9.sh
READONLY
./script9.sh: line 5: MYVAR: readonly variable
READONLY
```

10. Script to define a variable from command output.

```
$vim script10.sh
#!/bin/bash
## Defining a variable from Command Output
DATE=`date +%F` ##Date command with custom option
echo "Today's Date is --> $DATE"
```

Output:

```
[root@mlinux7 scripts]# chmod +x script10.sh
[root@mlinux7 scripts]# ./script10.sh
Today's Date is --> 2017-10-24
```

11. Script to define a variable using arithmetic substitution for Addition, Subtraction, Multiplication, Division and printing the output values.

```
$vim script11.sh
#!/bin/bash
### Declaring Arithmetic Substitution of variables.
ADD=$((2+1))
SUB=$((2-1))
MUL=$((2*1))
DIV=$((2/1))
echo -e "Addition(2+1)=$ADD\nSubtraction(2-1)=$SUB
Multiply(2*1)=$MUL\nDivide(2/1)=$DIV"
```

Output:

```
[root@mlinux7 scripts]# chmod +x script11.sh
[root@mlinux7 scripts]# ./script11.sh
Addition(2+1)=3
Subtraction(2-1)=1
Multiply(2*1)=2
Divide(2/1)=2
```

## READ COMMAND

To get input from the keyboard, you use the **read** command. The **read** command takes input from the keyboard and assigns it to a variable. It is mainly used at the time taking confirmation from the user to do the jobs that seems to be little dangerous.

Syntax:      **read <Variable Name>**      Ex: **read a**

It reads the input from the user and stores in a variable. In the above example it will store in variable named "a".

### **Read - p Option:**

If you would like to print the output while reading a variable then you have use **- p** option. This is helpful to reduce one echo command to display the output.

Syntax: **read -p "Message to be printed" variable**

Ex: **read -p "Enter You Name" name**

### **Read - s Option:**

If you would like to go with silent read we have to go with **- s** option in read command. Usually for reading passwords you have to mention the **- s option**; **read -s** option disables the feature of showing the text that you enter on the screen.

Syntax: **read -p -s "Message to be displayed" variable**

Ex: **read -p -s "Enter your password" pass**

The following is the script narrates the usage of Read Command.

```
#!/bin/bash

## Basic read command
echo -n "Enter your name: "
read name
echo "Your name is $name"

## Using -p option
read -p 'Enter your name: ' name
echo "Your name is $name"

## Using -s option to read
read -s -p 'Enter Password: ' pass
echo "Password is $pass"
```

### **Output:**

```
[root@mlinux7 ~]# ./read.sh
Enter your name: Musab
Your name is Musab
Enter your name: Musab Syed
Your name is Musab Syed
Enter Password: Password is vpts
```

**Lab Exercises:**

12. Script to read the input from the user using read command and print the output of the given input.

```
$ vim script12.sh
#!/bin/bash
### Script to read the input from the user.
echo "Please Enter Your Name:"
read name
echo "Please Enter your course:"
read course
echo -e "Your Name: $name\nYour Course: $course"
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script12.sh
[root@mlinux7 scripts]# ./script12.sh
Please Enter Your Name:
Musab
Please Enter your course:
Shell Scripting
Your Name: Musab
Your Course: Shell Scripting
```



13. Script to read the input from the user and use the same read command to print the output while reading.

```
$vim script13.sh
#!/bin/bash
### -p option in read helps you to print the message while reading.
read -p 'Pl Enter you Name:' name
read -p 'Pl Enter your course:' course
echo -e "Your Name: $name\nYour Course: $course"
```

**Output:**

```
[root@mlinux7 scripts]# vim script13.sh
[root@mlinux7 scripts]# chmod +x script13.sh
[root@mlinux7 scripts]# ./script13.sh
Pl Enter you Name: Musab
Pl Enter your course:Shell Scripting
Your Name: Musab
Your Course: Shell Scripting
```

## CONDITIONS

The order in which commands execute in a shell script is called the *flow* of the script. In the scripts that you have looked at so far, the flow is always the same because the same set of commands executes every time.

In most scripts, you need to change the commands that execute depending on some condition provided by the user or detected by the script itself. When you change the commands that execute based on a condition, you change the flow of the script

Two powerful flow control mechanics are available in the shell:

- The **case** statement
- The **if** statement

The case statement is the other major form of flow control available in the shell. Let's see a syntax to use it.

```
case WORD in
  pattern1)
    list1
    ;;
  pattern2)
    list2
    ;;
esac
```

The case statement can execute commands based on a pattern matching decision. The word <WORD> is matched against every pattern <PATTERN n> and on a match, the associated list <LIST n> is executed. Every command list is terminated by ;;, this rule is optional for the very last command list (i.e. you can omit the ;; before the esac). In the place of PATTERN if we give "\*" then if the WORD is not matched with any of the PATTERN then the LIST will be executed.

Basic Example of case statement:

```
#!/bin/bash
read -p 'Enter the OS: ' name
case $name in
    linux) cat /etc/redhat-release ;;
    unix) uname -a ;;
esac
Output:
[root@mlinux7 ~]# ./case1.sh
Enter the OS: linux
Red Hat Enterprise Linux Server release 7.2 (Maipo)
[root@mlinux7 ~]# ./case1.sh
Enter the OS: unix
Linux mlinux7.mbl.com 3.10.0-327.el7.x86_64 #1 SMP Thu
86_64 x86_64 x86_64 GNU/Linux
```

Same thing with multiple patterns, each pattern must be delimit using pipe symbol

```
case <word> in
    <Pattern1>|<Pattern2>
        List1
        ;;
    <Pattern3>
        List2
        ;;
esac
```

An example for multiple pattern

```
#!/bin/bash
read -p 'Enter the OS:' name
case $name in
    linux|centos|redhat|oel) cat /etc/redhat-release ;;
    aix|hpx) uname -a ;;
esac
```

#### Output

```
[root@mlinux7 ~]# sh case2.sh
Enter the OS:redhat
Red Hat Enterprise Linux Server release 7.2 (Maipo)
[root@mlinux7 ~]# sh case2.sh
Enter the OS:aix
Linux mlinux7.mb.com 3.10.0-327.el7.x86_64 #1 SMP Thu
86_64 x86_64 x86_64 GNU/Linux
```



#### Lab Exercises:

20. Script for simple case statements.

```
#!/bin/bash
### Simple case statement
read -p 'Pl Enter your Choice[L/U]: ' choice
case $choice in
    L) echo "You Have Selected Linux" ;;
    U) echo "You Have Selected Unix" ;;
    *) echo "Invalid option selected" ;;
esac
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script20.sh
[root@mlinux7 scripts]# ./script20.sh
Pl Enter your Choice[L/U]: L
You Have Selected Linux
[root@mlinux7 scripts]# ./script20.sh
Pl Enter your Choice[L/U]: U
You Have Selected Unix
[root@mlinux7 scripts]# ./script20.sh
Pl Enter your Choice[L/U]: R
Invalid option selected
```

## 21. Script for numerical operation using case statements.

```
$vim script21.sh
#!/bin/bash
## Numerical operation using case statements.
read -p 'Pl Enter 1st Value: ' a
read -p 'Pl Enter 2nd Value: ' b
read -p 'Enter the operator[ADD|SUB|MUL|DIV]: ' op
case $op in
    ADD) ADD=$((a+b))
          echo "Addition = $ADD" ;;
    SUB) SUB=$((a-b))
          echo "Subtract = $SUB" ;;
    MUL) MUL=$((a*b))
          echo "Multiply = $MUL" ;;
    DIV) DIV=$((a/b))
          echo "Divide = $DIV" ;;
    *) echo "Invalid Operator !!!" ;;
esac
```

### Output:

```
[root@mlinux7 scripts]# vim script21.sh
[root@mlinux7 scripts]# chmod +x script21.sh
[root@mlinux7 scripts]# ./script21.sh
Pl Enter 1st Value: 24
Pl Enter 2nd Value: 26
Enter the operator[ADD|SUB|MUL|DIV]: ADD
Addition = 50
```



### Test Command:

`test` command or [ expr ] is used to see if an expression is true, and if it is true it return zero(0), otherwise returns nonzero for false.

### Syntax:

**test expression**

Here expression is constructed using one of the special options to the `test` command. The `test` command returns either 0 (true) or a 1 (false) after evaluating an expression.

Shorthand for the `test` command is placing the expression in [ ].

**[ expression ]**

Here *expression* is any valid expression that the `test` command understands. This shorthand form is the most common form of test that you can encounter.

The types of expressions understood by `test` can be broken into three types:

- File tests
- String comparisons
- Numerical comparisons

### File Tests:

File test expressions test whether a file fits some particular criteria. The general syntax for a file test is

<b>test option file</b>	<b>or</b>	<b>[ option file ]</b>
-------------------------	-----------	------------------------

The following are the options available in file tests.

Option	Description
<b>-b</b>	True if file exists and is a block special file.
<b>-c</b>	True if file exists and is a character special file.
<b>-d</b>	True if file exists and is a directory.
<b>-e</b>	True if file exists.
<b>-f</b>	True if file exists and is a regular file.
<b>-g</b>	True if file exists and has its SGID bit set.
<b>-h</b>	True if file exists and is a symbolic link.
<b>-k</b>	True if file exists and has its "sticky" bit set.
<b>-p</b>	True if file exists and is a named pipe.
<b>-r</b>	True if file exists and is readable.
<b>-s</b>	True if file exists and has a size greater than zero.
<b>-u</b>	True if file exists and has its SUID bit set.
<b>-w</b>	True if file exists and is writable.
<b>-x</b>	True if file exists and is executable.
<b>-O</b>	True if file exists and is owned by the effective user ID.

### Example for file test

<code>[root@mlinux7 ~]# test -f case2.sh</code>
<code>[root@mlinux7 ~]# echo \$?</code>
<code>0</code>
<code>[root@mlinux7 ~]# test -b case2.sh</code>
<code>[root@mlinux7 ~]# echo \$?</code>
<code>1</code>

In the above example we are checking a file **case2.sh** as regular file or 0 using **-f** option and it returned a value 0 (true) which means it is true that particular file is a regular file. Then we are checking the same file whether it is block special file or not using **-b** option and it returned a value 1 (false) which means that particular file is not a block special file. Likewise we can check the files with all the options mentioned in the above table.

The **test** command also supports string comparisons. We can perform the following operations in string comparisons.

- Checking whether a string is empty or not.
- Checking whether two strings are equal or not.

The syntax for string comparison is

**test option string      or      [ option string ]**

The following are the available options in String Comparisons.

Option	Description
<b>-z string</b>	True if string has zero length.
<b>-n string</b>	True if string has nonzero length. It can also be defined as “!-z” string.
<b>string1 = string2</b>	True if the strings are equal.
<b>string1 != string2</b>	True if the strings are not equal.

Let's see an example how to use a test string comparison options

For checking whether a field is empty or not

```
[root@mlinux7 ~]# VAR=''
[root@mlinux7 ~]# [ -z "$VAR" ]
[root@mlinux7 ~]# echo $?
0
[root@mlinux7 ~]# [ -n "$VAR" ]
[root@mlinux7 ~]# echo $?
1
```

In the above-example we are defining a variable with an empty data and we are using **-z** option to check the variable has zero length and it returned a **zero** value as the string is empty. The same variable we checked with **-n** option and it returned a **non-zero** value as it doesn't have data.

**Note:** Make sure to use double quotes for strings while using the options.

For checking whether the strings are equal or not.

```
[root@mlinux7 ~]# VAL1='xyz'
[root@mlinux7 ~]# VAL2='abc'
[root@mlinux7 ~]# test "$VAL1" = "$VAL2"
[root@mlinux7 ~]# echo $?
1
[root@mlinux7 ~]# test "$VAL1" != "$VAL2"
[root@mlinux7 ~]# echo $?
0
```

In the above example we initialized two variables with different data in it and we are checking them whether those are equal or not. Where first returned 1 and other 0.

### Numerical Comparisons:

The test command enables us to compare two integers. The Syntax is as follows.

<b>test integer1 option integer2</b>	<b>or</b>	<b>[ integer1 operator integer2 ]</b>
--------------------------------------	-----------	---------------------------------------

Option	Description
<b>int1 -eq int2</b>	True if int1 is equals to int2.
<b>int1 -ne int2</b>	True if int1 is not equal to int2.
<b>int1 -lt int2</b>	True if int1 is less than int2.
<b>int1 -le int2</b>	True if int1 is less than or equal to int2.
<b>int1 -gt int2</b>	True if int1 is greater than int2.
<b>int1 -ge int2</b>	True if int1 is greater than or equal to int2.

### The IF Conditions:

The “if” conditional statements perform different computations or actions depending on whether a programmer-specified Boolean condition evaluates to **true** or **false**. These statements are used to execute different parts of your shell program depending on whether certain conditions are true. The ability to branch makes shell scripts powerful.

We have the following **if** conditional statements.

- If..then..fi statement (Simple if)
- If..then else..fi statement (If-Else)
- If..elif..else..fi statement (Else if ladder)
- If..then..else..if..then..fi..fi..(Nested if)

#### Simple if:

This statement is also called as simple if statement, the example is as follows.

<b>If [ conditional expression ]</b>
--------------------------------------

**Then**

**Statement 1**

**Statement 2**

...

**fi**

If the given conditional expression is true or returns zero, it enters and executes the statements enclosed between the keywords "then" and "fi". If the given expression is false or returns a non-zero, then consequent statement list is executed.

The following the example for Simple-If conditional statement.

<pre>#!/bin/bash val=10 if [ \$val -eq 10 ] then     echo "Value is 10" fi</pre>
----------------------------------------------------------------------------------

### If Else:

If then else statement are usually called as if else statement and the syntax is as follows.

```
if [ conditional expression ]
then
    Statement1
    Statement2
    ...
else
    Statement3
    Statement4
    ...
fi
```

If the conditional expression is true or returns a zero value, it executes the statement1 and 2. If the conditional expression is false or returns a nonzero, it jumps to else part, and executes the statement3 and 4. After the execution of if/else part, execution resume with the consequent statements.

```
#!/bin/bash
val=99
if [ $val -eq 100 ]
then
echo "Val is 100"
else
echo "Val is not 100"
fi
```

### Else If Ladder:

If..elif..else..fi statements are called as Else if ladder statements. The syntax is as follows

```
if [ conditional expression1 ]
then
    Statement1
    Statement2
elif [ conditional expression2 ]
then
    Statement3
    Statement4
else
    Statement5
fi
```

If you want to check more than one conditional expression then you have to use Else If conditional statements. It checks expression 1, if it is true executes statement 1, 2. If expression1 is false, it checks expression2, and if all the expression is false, then it enters into else block and executes the statements in the else block.

The following example narrates the usage of Else-If Statements.

```
#!/bin/bash
val=99
if [ $val -eq 100 ]
then
echo "Val is 100"
elif [ $val -gt 100 ]
then
echo "Val is greater than 100"
else
echo "Val is less than 100"
fi
```

#### Nested If:

If..then..if..then..fi..fi Statements are called as Nested If Statements. The Syntax is as follows.

```
if [ conditional expression1 ]
then
    statement1
    statement2
else
    if [ conditional expression2 ]
    then
        statement3
    fi
fi
```

An example for Nested if statements.

```
#!/bin/bash
val=99
if [ $val -le 100 ]
    then
        echo "Val is less than 100"
        if [ $val -eq 100 ]
            then
                echo "Val is equal to 100"
            else
                echo "Val is Not equal to 100"
            fi
    fi
```

### Lab Exercises:

22. Script with numerical operators in IF statements.

```
$vim script22.sh
#!/bin/bash
### Numerical Operators in If Statement
a=10
b=11
if [ $a -eq $b ]; then
    echo "$a is equal to $b"
fi
if [ $a -ne $b ]; then
    echo "$a is not equal to $b"
fi
if [ $a -gt $b ]; then
    echo "$a is greater than $b"
fi
if [ $a -ge $b ]; then
    echo "$a is greater than or equal to $b"
fi
if [ $a -lt $b ]; then
    echo "$a is lesser than $b"
fi
if [ $a -le $b ]; then
    echo "$a is less than or equal to $b"
fi
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script22.sh
[root@mlinux7 scripts]# ./script22.sh
10 is not equal to 11
10 is lesser than 11
10 is less than or equal to 11
```

23. Script for File Tests in IF Statements.

```
$vim script23.sh
#!/bin/bash
#### File Tests using IF Statements.
FILE='/dev/sda'
if [ -c $FILE ]; then
    echo "$FILE is a character special file"
else
    echo "$FILE is not a character special file"
```

```

fi
if [ -b $FILE ]; then
    echo "$FILE is a block special file"
else
    echo "$FILE is not a block special file"
fi
if [ -d $FILE ]; then
    echo "$FILE is a directory"
else
    echo "$FILE is not a directory"
fi

```

Output:

```

[root@mlinux7 scripts]# vim script23.sh
[root@mlinux7 scripts]# chmod +x script23.sh
[root@mlinux7 scripts]# ./script23.sh
/dev/sda is not a character special file
/dev/sda is a block special file
/dev/sda is not a directory

```

**24. Script for String comparisons in IF Statements.**

```

$vim script24.sh
#!/bin/bash
# String Comparisons
VAL1='VPT'
VAL2='vpt'

if [ $VAL1 = $VAL2 ]; then
    echo "The given values are equal"
else
    echo "The given values are not equal"
fi

```

Output:

```

[root@mlinux7 scripts]# chmod +x script24.sh
[root@mlinux7 scripts]# ./script24.sh
The given values are not equal

```

25. Script to check whether the given input have data or not.

```
$ vim script25.sh
#!/bin/bash
### Use -z and -n option to check the variable is empty or not.
read -p 'Please Enter the value: ' A
if [ -z "$A" ]; then
echo "Variable A is empty"
fi
if [ -n "$A" ]; then
echo "Variable A is not empty"
fi
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script25.sh
[root@mlinux7 scripts]# ./script25.sh
Please Enter the value: Hi
Variable A is not empty
[root@mlinux7 scripts]# ./script25.sh
Please Enter the value:
Variable A is empty
```

**Note:** Always remember that while you are comparing strings go with Quotes or else you will some errors in some cases, so it is a best practice to start writing the scripts using quotes.

26. Script for Numerical Operations using IF Statements and also check whether the given input from the user is valid or no.

```
$vim script26.sh
#!/bin/bash
## Numerical Operations including If Statements and String Checks.
read -p 'Enter value1: ' a
if [ -z $a ]; then
echo "Invalid input!!"
exit
fi
read -p 'Enter value2: ' b
if [ -z $b ]; then
echo "Invalid input!!"
exit
fi
read -p 'Enter the Operator[ADD|SUB|MUL|DIV]: ' op
if [ -z $op ]; then
echo "Invalid input!!"
exit
fi
if [ "$op" = "ADD" ]; then
echo "Addition = $((a+b))"
fi
```

```

if [ "$op" = "SUB" ]; then
    echo "Subtract = $((a-$b))"
fi
if [ "$op" = "MUL" ]; then
    echo "Multiply = $((a*$b))"
fi
if [ "$op" = "DIV" ]; then
    echo "Divide = $((a/$b))"
fi

```

**Output:**

```

[root@mlinux7 scripts]# chmod +x script26.sh
[root@mlinux7 scripts]# ./script26.sh
Enter value1:
Invalid input!!
[root@mlinux7 scripts]# ./script26.sh
Enter value1: 10
Enter value2:
Invalid input!!
[root@mlinux7 scripts]# ./script26.sh
Enter value1: 20
Enter value2: 2
Enter the Operator[ADD|SUB|MUL|DIV]: DIV
Divide = 10

```

**27. Script for Numerical Operations using ELSE-IF statement.**

```

$vim script27.sh
#!/bin/bash
## Numerical operations including ELSE-If Statements and String Checks.
read -p 'Pl Enter 1st Value: ' a
if [ -z $a ];then
    echo "Invalid input!!"
    exit
fi
read -p 'Pl Enter 2nd Value: ' b
if [ -z $b ]; then
    echo "Invalid input!!"
    exit
fi
read -p 'Enter the Operator[ADD|SUB|MUL|DIV]: ' op
if [ "$op" = "ADD" ]; then
    echo "Addition= $((a+b))"
elif [ "$op" = "SUB" ]; then
    echo "Subtract = $((a-b))"
elif [ "$op" = "MUL" ]; then
    echo "Multiply= $((a*b))"
elif [ "$op" = "DIV" ]; then
    echo "Divide = $((a/b))"

```

```

else
echo "Invalid Operator!!"
exit
fi

```

**Output:**

```

[root@mlinux7 scripts]# vim script27.sh
[root@mlinux7 scripts]# chmod +x script27.sh
[root@mlinux7 scripts]# ./script27.sh
P1 Enter 1st Value: 200
P1 Enter 2nd Value: 10
Enter the Operator[ADD|SUB|MUL|DIV]: DIV
Divide = 20

```

## 28. Script for Numerical Operations using Nested ELSE-IF.

```

$vim script28.sh
#!/bin/bash
### Nested IF-ELSE
read -p 'P1 Enter the 1st Value: ' a
read -p 'P1 Enter the 2nd Value: ' b
if [ -n "$a" ]; then
    if [ -z "$b" ]; then
        echo "Invalid input 1st Value!!"
        exit
    fi
else
    echo "Invalid input 2nd Value!!"
    exit
fi
read -p 'P1 Chose the Operator [ADD|SUB|MUL|DIV]: ' op
if [ -n "$op" ]; then
    if [ $op = ADD ]; then
        echo "Addition = $((a+b))"
    else
        if [ $op = SUB ]; then
            echo "Subtract = $((a-b))"
        else
            if [ $op = MUL ]; then
                echo "Multiply = $((a*b))"
            else
                if [ $op = DIV ]; then
                    echo "Division = $((a/b))"
                else
                    echo "Invalid Operator"
                    exit
                fi
            fi
        fi
    fi
fi

```

```

        fi
    fi
else
    echo "Invalid Operator Input"
    exit
fi

```

Output:

```

[root@mlinux7 scripts]# vim script28.sh
[root@mlinux7 scripts]# chmod +x script28.sh
[root@mlinux7 scripts]# ./script28.sh
P1 Enter the 1st Value: 25
P1 Enter the 2nd Value: 17
P1 Chose the Operator [ADD|SUB|MUL|DIV]: SUB
Subtract = 8

```

29. Script for Numerical operations using ELSE-IF and also exit with specific value if any invalid data is given.

```

$vim script29.sh
#!/bin/bash
## Numerical Operations including ELSE-If Statement
### We can exit with the custom exit
read -p 'Enter 1st Value: ' a
if [ -z $a ]; then
    echo "Invalid input!!"
    exit 1
fi
read -p 'Enter 2nd Value: ' b
if [ -z $b ]; then
    echo "Invalid input!!"
    exit 1
fi
read -p 'Enter the Operator[ADD|SUB|MUL|DIV]: ' op
if [ "$op" = "ADD" ]; then
    echo "Addition = $((a+b))"
elif [ "$op" = "SUB" ]; then
    echo "Subtract = $((a-b))"
elif [ "$op" = "MUL" ]; then
    echo "Multiply = $((a*b))"
elif [ "$op" = "DIV" ]; then
    echo "Divide = $((a/b))"
else
    echo "Invalid Operator!!"
    exit 1
fi

```



Output:

```
[root@mlinux7 scripts]# vim script29.sh
[root@mlinux7 scripts]# chmod +x script29.sh
[root@mlinux7 scripts]# ./script29.sh
Enter 1st Value:
Invalid input!!
[root@mlinux7 scripts]# echo $?
1
[root@mlinux7 scripts]# ./script29.sh
Enter 1st Value: 10
Enter 2nd Value: 2
Enter the Operator[ADD|SUB|MUL|DIV]: MUL
Multiply = 20
[root@mlinux7 scripts]# echo $?
0
```

### Compound Expressions

- So far you have seen individual expressions, but many times you need to combine expressions in order to satisfy a particular expression. When two or more expressions are combined, the result is called a *compound* expression.
- You can create compound expressions using the test command's built in operators, or you can use the conditional execution operators, && and ||.
- Also you can create a compound expression that is the negation of another expression by using the! Operator.

The following are the available options.

Option	Description
<b>! expr</b>	True if expr is false.
<b>expr1 -a expr2</b>	True if both the expressions are true.
<b>expr1 -o expr2</b>	True if either of expression is true.

The following is the example that narrates the usage of compound expressions of commands.

```
[root@mlinux7 ~]# test -f /etc/passwd && echo "File Exists"
File Exists
[root@mlinux7 ~]# test -d /tmp/dir1 && echo "Directory Exists"
[root@mlinux7 ~]# echo $?
1
[root@mlinux7 ~]# test -d /tmp/dir1 || mkdir /tmp/dir1
[root@mlinux7 ~]# echo $?
0
```

An example for the usage of compound expressions of if statement

```
#!/bin/bash
read -p 'Please Enter your name: ' name
read -p 'Please Enter your Course: ' course
if [ -z "$name" -o -z "$course" ]; then
    echo "Invalid Data !!. Please enter again"
    exit 1
fi
echo "WELCOME TO VPTS"
```

*Output:*

```
[root@mlinux7 ~]# sh compound.sh
Please Enter your name: Musab
Please Enter your Course: Shell Scripting
WELCOME TO VPTS
```

```
[root@mlinux7 ~]# sh compound.sh
Please Enter your name:
Please Enter your Course:
Invalid Data !!. Please enter again
```

### 30. Script for compound expressions in IF Statements.

```
$vim script30.sh
#!/bin/bash
###Compound expression in IF statements using -a (Logical AND)
### and -o (Logical OR)

###Logical AND
if [ -f /etc/passwd -a 10 -eq 10 ]; then
    echo "The file /etc/passwd exists and 10 is equal to 10"
fi
###Logical OR
if [ -f /etc/passwd -o 10 -eq 11 ]; then
    echo "The file /etc/passwd exists but 10 is not equal to 11"
fi
```

*Output:*

```
[root@mlinux7 scripts]# vim script30.sh
[root@mlinux7 scripts]# chmod +x script30.sh
[root@mlinux7 scripts]# ./script30.sh
The file /etc/passwd exists and 10 is equal to 10
The file /etc/passwd exists but 10 is not equal to 11
```

**31.** Script for Numerical Operations using IF Statements, check input using compound expressions, and exit with specific exit status.

\$vim script31.sh

```
#!/bin/bash
### Numerical Operations Using Compound If, ELSE-IF and specific exit status
read -p 'Enter value1: ' a
read -p 'Enter value2: ' b
### Using Logical OR in IF Statement .
if [ -z "$a" -o -z "$b" ]; then
    echo "Invalid Input."
    exit 1
fi
read -p 'Enter the operator(ADD|SUB|MUL|DIV): ' op
if [ $op = "ADD" ]; then
    OUT=$((a+b))
elif [ $op = "SUB" ]; then
    OUT=$((a-b))
elif [ $op = "MUL" ]; then
    OUT=$((a*b))
elif [ $op = "DIV" ]; then
    OUT=$((a/b))
else
    echo "Invalid Operator!!"
    exit 1
fi
echo "OUTPUT = $OUT"
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script31.sh
[root@mlinux7 scripts]# ./script31.sh
Enter value1:
Enter value2:
Invalid Input.
[root@mlinux7 scripts]# ./script31.sh
Enter value1: 10
Enter value2:
Invalid Input.
[root@mlinux7 scripts]# ./script31.sh
Enter value1: 20
Enter value2: 5
Enter the operator(ADD|SUB|MUL|DIV): DIV
OUTPUT = 4
```

**32. Script to nullify the output and error.**

```
$vim script32.sh
#!/bin/bash
### Nullifying the command output and checking whether command is
## executed successfully or not.
ls >/dev/null 2>&1
no-command >/dev/null 2>&1
```

**Output**

```
[root@mlinux7 scripts]# chmod +x script32.sh
[root@mlinux7 scripts]# ./script32.sh
[root@mlinux7 scripts]#
```

**33. Script to nullify unwanted output and check the command executed successfully or not.**

```
$ vim script33.sh
#!/bin/bash
### Check the exit status of previous executed command using $? variable
ls >/dev/null 2>&1
if [ $? -eq 0 ]; then
    echo "ls Command executed Successfully"
else
    echo "ls Command Not executed Successfully"
fi
lss >/dev/null 2>&1
STATUS=$?
if [ $STATUS -eq 0 ]; then
    echo "lss Command executed Successfully"
else
    echo "lss command not executed Successfully"
fi
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script33.sh
[root@mlinux7 scripts]# ./script33.sh
ls Command executed Successfully
lss command not executed Successfully
```

34. Script to check the given number is a number or a string with characters.

```
$vim script34.sh
#!/bin/bash
## Script to check the given number is a number or a string
read -p 'Pl Enter a value: ' a
test $a -eq 0 >/dev/null 2>&1
if [ $? -eq 2 ]; then
    echo "The value you have entered is not a integer"
else
    echo "The value you have entered is a integer"
fi
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script34.sh
[root@mlinux7 scripts]# ./script34.sh
Pl Enter a value: 10
The value you have entered is a integer
[root@mlinux7 scripts]# ./script34.sh
Pl Enter a value: hello
The value you have entered is not a integer
[root@mlinux7 scripts]# ./script34.sh
Pl Enter a value:
The value you have entered is not a integer
```

## LOOPS

Loops enable you to execute a series of commands multiple times. There are two basic types of loops

- The while loop
- The for loop

### **While loop.**

The *while* loop is a control flow statement that allows code or commands to be executed repeatedly based on a given condition or expression is true or not. The following is the syntax for basic while loop.

```
while [ condition ]
do
    command1
    command2
    command3
done
```

If the given condition is true then the given commands will be executed continuously one by one till the condition get fails. So it is all our responsibility to control the condition otherwise the loop may get into infinite loop.

**Note:** The conditions can be either numerical or string or file check conditions.

The following is the example that narrates the usage of **while** loop.

```
#!/bin/bash
x=1
while [ $x -le 5 ]
do
    echo "Welcome $x times"
    x=$(( $x +1 ))
done
```

### **Lab Exercises:**

35. Script for basic while loop.

```
$ vim script35.sh
#!/bin/bash
### while Loop
a=1
while [ $a -le 5 ]
do
    echo "This is $a iteration"
    a=`echo $a+1|bc`
done
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script35.sh
[root@mlinux7 scripts]# ./script35.sh
This is 1 iteration
This is 2 iteration
This is 3 iteration
This is 4 iteration
This is 5 iteration
```

36. Script for Numerical Operations using while loop. Also check the given value for loop iteration is integer or not, also validate the data given and exit with the specific exit status.

```
#!/bin/bash
### Numerical operations using while loop.
read -p 'Pl Enter Number for iterations: ' i
if [ -n $i ]; then
    test $i -eq 0 >/dev/null 2>1
    if [ $? -eq 2 ]; then
        echo "Pl enter only integers"
        exit 2
    fi
else
    echo "Pl enter a value."
    exit 1
fi
while [ $i -gt 0 ]
do
    read -p 'Enter the 1st value: ' a
    read -p 'Enter the 2nd value : ' b
    if [ -z $a -o -z $b ]; then
        echo "Invalid Input"
        exit 1
    fi
    read -p 'Enter the Operator[ADD|SUB|MUL|DIV]: ' op
    case $op in
        ADD) OUT=$((a+b));;
        SUB) OUT=$((a-b));;
        MUL) OUT=$((a*b));;
        DIV) OUT=$((a/b));;
        *) echo "Invalid Operator!!"
    esac
    echo "OUTPUT =$OUT"
    i=`echo $i - 1|bc`
done
```



**Output:**

```
[root@mlinux7 scripts]# chmod +x script36.sh
[root@mlinux7 scripts]# ./script36.sh
P1 Enter Number for iterations: Hi
P1 enter only integers
[root@mlinux7 scripts]# ./script36.sh
P1 Enter Number for iterations: 1
Enter the 1st value: 10
Enter the 2nd value : 20
Enter the Operator[ADD|SUB|MUL|DIV]:ADD
OUTPUT =30
```

**UNTIL Loop:**

*Until* loop is very similar to the while loop, except that the loop executes until the expression executes successfully. As long as the given condition fails, the loop continues. The following is the basic syntax for *until* loop.

```
until [ condition ]
do
    command1
    command2
    command3
done
```

If the given condition is false then the given commands will be executed till the condition get pass or true. Until loop is actually the reverse of the while loop and also in other words these two loops are in vice versa in terms of logic in condition checking.

The following is the example narrates the usage of *until* loop.

```
#!/bin/bash
x=1
until [ $x -gt 5 ]
do
echo "Welcome $x times"
x=$(( $x + 1 ))
done
```

**Lab Exercises:**

37. Script for basic Until loop.

```
$vim script37.sh
#!/bin/bash
### Until Loop
a=5
until [ $a -le 0 ]
do
echo "This is $a iteration"
a=`echo $a-1|bc`
done
```

Output:

```
[root@mlinux7 scripts]# chmod +x script37.sh
[root@mlinux7 scripts]# ./script37.sh
This is 5 iteration
This is 4 iteration
This is 3 iteration
This is 2 iteration
This is 1 iteration
```

38. Script to do Numerical operations using until loop and also check the given values have the valid data or not.

```
$ vim script38.sh
#!/bin/bash
### Numerical operations using Until Loop.
read -p 'Enter Number of Iterations: ' i
if [ -n $i ]; then
    test $i -eq 0 >/dev/null 2>&1
    if [ $? -eq 2 ]; then
        echo "Pl enter Integers only"
        exit 2
    fi
else
    echo "You should enter a value.."
    exit 1
fi
until [ $i -le 0 ]
do
    read -p 'Enter the 1st Value: ' a
    read -p 'Enter the 2nd Value: ' b
    if [ -z "$a" -o -z "$b" ]; then
        echo "Invalid Input!!"
        exit 1
    fi
    read -p 'Enter the operator[ADD|SUB|MUL|DIV]: ' op
    case $op in
        ADD) OUT=$((a+b));;
        SUB) OUT=$((a-b));;
        MUL) OUT=$((a*b));;
        DIV) OUT=$((a/b));;
        *) echo "Invalid Operator!!" ;exit 1;;
    esac
    echo "OUTPUT = $OUT"
    i=`echo $i - 1|bc`
done
```



**Output:**

```
[root@mlinux7 scripts]# chmod +x script38.sh
[root@mlinux7 scripts]# ./script38.sh
Enter Number of Iterations: 1
Enter the 1st Value:
Enter the 2nd Value: 2
Invalid Input!!
[root@mlinux7 scripts]# ./script38.sh
Enter Number of Iterations: 1
Enter the 1st Value: 2
Enter the 2nd Value:
Invalid Input!!
[root@mlinux7 scripts]# ./script38.sh
Enter Number of Iterations: 1
Enter the 1st Value: 2
Enter the 2nd Value: 8
Enter the operator[ADD|SUB|MUL|DIV]: MUL
OUTPUT = 16
```

39. Script for sleep command.

```
$ vim script39.sh
#!/bin/bash
## Sleep command will sleep the CPU for the number of seconds given.
sleep 10
## The above command will sleep for 10 seconds and executes
#commands.
date
Output:
[root@mlinux7 scripts]# chmod +x script39.sh
[root@mlinux7 scripts]# ./script39.sh
Tue Oct 31 11:58:22 IST 2017
```

**FOR Loop:**

Unlike **While** and **Until** loop, **FOR** Loop operates on list of items given, **for** loop are typically used when the number of operations is known before entering the bash loop. Bash supports two kinds of for loop. The first form of bash for loop is:

```
for varname in list
    do
        commands ##Body of the loop
    done
```

In the above syntax:

- for, in, do and done are keywords
- List is any list which has list of items
- varname is any Bash variable name.

In this form, the **for** statement executes the command enclosed in a body, once for each item in the list. The current item from the list will be stored in a variable "varname" each time through the loop. This varname can be processed in the body of the loop. This list can be a variable that contains several words separated by spaces. If list is missing in for statement, then it takes the positional parameters that were passed into the shell.

The following is the example narrating **FOR** Loop first form.

```
#!/bin/bash
for var in 1 2 3 4 5
do
echo "Iteration $var"
done
```

The second form of for loop is similar to **for** loop in 'C' programming language, which has three expression (initialization, condition and updating).

```
for (( expr1; expr2; expr3 ))
do
    commands
done
```

- In the above bash for command syntax, before the first iteration, expr1 is evaluated. This is usually used to initialize variables for the loop.
- All the statements between do and done is executed repeatedly until the value of expr2 is TRUE.
- After each iteration of the loop, expr3 is evaluated. This is usually use to increment a loop counter.

The following is the example narrating **FOR** Loop Second form:

```
#!/bin/bash
for (( i=0; i <= 5; i++ ))
do
    echo $i
done
```



### **Lab Exercises:**

#### **40. Script for basic FOR loop.**

```
$ vim script40.sh
#!/bin/bash
### FOR LOOP --First Method
for var in VALUE1 VALUE2 VALUE3
do
    echo $var
    sleep 1
done
```

#### **Output:**

```
[root@mlinux7 scripts]# chmod +x script40.sh
[root@mlinux7 scripts]# ./script40.sh
VALUE1
VALUE2
VALUE3
```

**41.** Script for basic FOR loop and use command to give the inputs.

```
$vim script41.sh
#!/bin/bash
## For Loop -- Second Method
for var in `echo VALUE1 VALUE2 VALUE3`
do
    echo $var
    sleep 1
done

echo "....."
for var in `cat out`
do
    echo $var
    sleep 1
done
```

**Output:**

```
[root@mlinux7 scripts]# cat out
VALUE4
VALUE5
VALUE6
[root@mlinux7 scripts]# chmod +x script41.sh
[root@mlinux7 scripts]# ./script41.sh
VALUE1
VALUE2
VALUE3
.....
VALUE4
VALUE5
VALUE6
```

**42.** Script for Numerical operations using FOR Loop.

```
$vim script42.sh
#!/bin/bash
### Numerical Operations Using FOR LOOP and CASE Statements.
read -p 'Enter 1st Value: ' a
read -p 'Enter 2nd Value: ' b
if [ -z $a -o -z $b ]; then
    echo "Invalid Input"
    exit 1::
fi
for op in ADD SUB MUL DIV
do
    case $op in
        ADD) echo "Addition=$((a+b))";;
        SUB) echo "Subtraction=$((a-b))";;
        MUL) echo "Multiplication=$((a*b))";;
        DIV) echo "Division=$((a/b))";;
    esac
done
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script42.sh
[root@mlinux7 scripts]# ./script42.sh
Enter 1st Value: 20
Enter 2nd Value: 5
Addition=25
Subtraction=15
Multiplication=100
Division=4
```

**Select Loop with PS3 statement:**

The bash variable PS3 is exclusively used for select loop prompting message. If you would like to use the custom prompt then set the PS3 variable with the message. The following example shows using of PS3 in select loop.

```
#!/bin/bash
PS3="Please select your Option>" 
select var in linux unix
do
echo "You Selected $var"
done
```

**Note:** You can see the difference in output while executing the program.

Usually *select* loop includes the either *case* or *Else-if* Statements for better control on the loops.

**Lab Exercises:**

**43. Script for basic Select loop**

```
$vim script43.sh
#! /bin/bash
###Select Loop
select var in VALUE1 VALUE2 VALUE3
do
echo "You have selected --> $var"
done
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script43.sh
[root@mlinux7 scripts]# ./script43.sh
1) VALUE1
2) VALUE2
3) VALUE3
#? 1
You have selected --> VALUE1
#? 3
You have selected --> VALUE3
#? 2
You have selected --> VALUE2
#? ^C
```

**Note:** Press **CTRL + C** to come out of the script.

**44. Script for basic select loop and using PS3 variable.**

```
$ vim script44.sh
#!/bin/bash
### Select Loop using PS3 variable to get the custom prompt
PS3='Select the Value > '
select var in VALUE1 VALUE2 VALUE3
do
    echo "You have selected --> $var"
done
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script44.sh
[root@mlinux7 scripts]# ./script44.sh
1) VALUE1
2) VALUE2
3) VALUE3
Select the Value > 1
You have selected --> VALUE1
Select the Value > 3
You have selected --> VALUE3
Select the Value > 2
You have selected --> VALUE2
Select the Value > ^C
```

**45. Script for Numerical Operations using select loop.**

```
$vim script45.sh
#!/bin/bash
##Numerical Operations using Select Loop
PS3='Pl Select a value > '
select op in ADD SUB MUL DIV EXIT
do
    if [ $op = "EXIT" ]; then
        exit
    fi
    read -p 'Enter Val1: ' a
    read -p 'Enter Val2: ' b
    if [ -z "$a" -o -z "$b" ]; then
        echo "Invalid Inupt!!!"
        exit 1
    fi
    case $op in
    ADD) OUT=$((a+b));;
    SUB) OUT=$((a-b));;
    MUL) OUT=$((a*b));;
    DIV) OUT=$((a/b));;
    esac
    echo "Output = $OUT"
done
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script45.sh
[root@mlinux7 scripts]# ./script45.sh
1) ADD
2) SUB
3) MUL
4) DIV
5) EXIT
Pl Select a value > 1
Enter Val1: 10
Enter Val2: 2
Output = 12
Pl Select a value > 4
Enter Val1: 10
Enter Val2: 2
Output = 5
Pl Select a value > 5
```

**Loop Control:**

The *break* and *continue* are loop control commands in shell to have better control of loops in some cases of infinite loop or even in regular loops. The *break* command terminates the loop (breaks out of it), while *continue* causes a jump to the next iteration of the loop, skipping all the remaining commands in that particular loop cycle.

Let's see how to use the *break* command.

```
#!/bin/bash
PS3="Please select your Option> "
select var in Linux Unix
do
    case $var in
        Linux) cat /etc/issue; break ;;
        Unix) uname -a; break ;;
    esac
done
```

And now let's see the *continue* option

```
#!/bin/bash
a=1
while [ $a -lt 10 ]; do
    if [ $a -gt 5 ]; then
        echo "$a is greater than 5"
        a=$((a+1))
        continue
    fi
    echo "$a is less than or equal to 5"
    a=$((a+1))
done
```

**Lab Exercises:**

**46. Script for controlling the loop using break command.**

```
$vim script46.sh
#!/bin/bash
### break command.
a=1
while [ $a -le 10 ]
do
    a=`echo $a+1|bc`
    echo "This is before break command"
    if [ $a -eq 3 ]; then
        echo "Break command Encountered"
        break
    fi
    echo "This is after break command"
done
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script46.sh
[root@mlinux7 scripts]# ./script46.sh
This is before break command
This is after break command
This is before break command
Break command Encountered
```

**47. Script for controlling Loop using continue command.**

```
$vim script47.sh
#!/bin/bash
### Continue command .
a=1
while [ $a -le 5 ]
do
    a=`echo $a+1|bc`
    echo "This is before continue command"
    if [ $a -gt 3 ]; then
        echo "Continue Command encountered"
        continue
    fi
    echo "This is after continue command"
done
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script47.sh
[root@mlinux7 scripts]# ./script47.sh
This is before continue command
This is after continue command
This is before continue command
This is after continue command
This is before continue command
Continue Command encountered
This is before continue command
Continue Command encountered
This is before continue command
Continue Command encountered
```

**48.** Script for infinite loop using while loop and usage of break command to come out of loop.

\$ vim script48.sh

```
#!/bin/bash
## Come out of infinite loop using break
a=1
while true
do
    echo "Iteration No : $a"
    a=`echo $a+1|bc`
    if [ $a -gt 5 ]; then
        echo "Break Command Encountered"
        break
    fi
done
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script48.sh
[root@mlinux7 scripts]# ./script48.sh
Iteration No : 1
Iteration No : 2
Iteration No : 3
Iteration No : 4
Iteration No : 5
Break Command Encountered
```

**49.** Script for Numerical Operations using While, Select loop and also use break and continue statements to control these loops.

\$ vim script49.sh

```
#!/bin/bash
### While & Select Loop using break and continue command
PS3='Select an Option: '
while true
do
    read -p 'Enter 1st Value: ' a
    read -p 'Enter 2nd Value: ' b
    if [ -z "$a" -o -z "$b" ]; then
        echo "Invalid Inputs!!!"
        continue
    fi
    select op in ADD SUB MUL DIV EXIT
    do
        case $op in
            ADD) echo "Addition = $((a+b))";break;;
            SUB) echo "Subtract = $((a-b))";break;;
            MUL) echo "Multiply = $((a*b))";break;;
            DIV) echo "Divide = $((a/b))";break;;
            EXIT) exit;;
            *) echo "Wrong option selected..!! Try Again..!!";
    sleep 1; continue;;
        esac
    done
done
```

Output:

```
[root@mlinux7 scripts]# chmod +x script49.sh
[root@mlinux7 scripts]# ./script49.sh
Enter 1st Value: 10
Enter 2nd Value: 20
1) ADD
2) SUB
3) MUL
4) DIV
5) EXIT
Select an Option: 6
Wrong option selected..!! Try Again..!!
Select an Option: 3
Multiply = 200
Enter 1st Value: 5
Enter 2nd Value: 2
1) ADD
2) SUB
3) MUL
4) DIV
5) EXIT
Select an Option: 5
```



## FUNCTIONS

Bash shell functions are a way to group several UNIX / Linux commands for later execution using a single name for the group. Bash shell function can be executed just like a regular UNIX command. Shell functions are executed in the current shell context without creating any new process to interpret them.

Both bash aliases and functions allow you to define shortcuts for longer or more complicated commands. However, aliases don't allow control-flows, arguments, and other trickery things which these functions allows.

### Using Functions:

The syntax for creating a functions goes as follows.

```
name( )
{
list
}
```

In the above syntax **name** is the name of the function and **list** is the list of commands. The list of commands, list, is referred to as the body of the function. And also the parenthesis i.e, ( { } ) are required followed by the name of function name. The job of a function is to bind name to list, so that whenever name is specified, list is executed. When a function is defined, list is not executed; the shell parses list to ensure that there are no syntax errors and stores name in its list of commands.

Let's see an example for the usage of functions:

```
#!/bin/bash
#### Defining a function
myfunc() {
echo "This is a function"
}
myfunc #### Calling a function
```

### Return Status in functions:

Instead of simply returning from the function we can return with some status like the same we have in exit status. We can return only the value and a string cannot be returned here in case of how the other programming languages do.

→ You can check the return status of a function using \$? Variable only.

Let's see an example for the same with status:

```
#!/bin/bash
myfunc() {
    echo "Executing 1st Command"
    return $?
}
myfunc
if [ $? -eq 0 ]; then
    echo "Command in the function completed successfully"
else
    echo "Command in the function failed with an error"
fi
```

### Local Variables in functions:

By default all the variables in shell are global. Modifying such variables will change the value in the whole script. So this may lead to have bugs while building a large scripts, we can overcome this by defining a variables locally in the function using local command. The scope these functions will be within the function and also global variables will temporarily overwritten by local variables.

The following is the syntax to define a local variable in functions.

```
function name( ){
    local var=$1
    ...
}
```

Let's see the same with an example

```
#!/bin/bash
myfunc1() {
    a=10
}
myfunc2() {
    local b=20
}
myfunc1
myfunc2
echo "a=$a"
echo "b=$b"
```



### Export a Function:

We can export a function using **export** command **-f** option. The following example explains the function exporting.

```
fname() {
echo "Welcome to Scripting"
}
export -f fname
```

### Read-only Functions:

We can make a function as readonly using **readonly** command **-f** option. The following example narrates readonly function.

```
fname() {
echo "Welcome to Scripting"
}
readonly -f fname
```

### Lab Exercises:

50. Script for defining and calling a function.

```
#vim script50.sh
#!/bin/bash
###Defining a function
NAME() {
echo "VPTS"
echo -n "Today date is "
date
}
### Using a function
NAME
Output:
[root@mlinux7 scripts]# chmod +x script50.sh
[root@mlinux7 scripts]# ./script50.sh
VPTS
Today date is Tue Oct 31 14:28:02 IST 2017
```

51. Script to call a function defined in shell as environment function.

```
$ vim script51.sh
#!/bin/bash
### Calling a function which is defined in the local shell
MYFUNC
Output:
[root@mlinux7 scripts]# vim script51.sh
[root@mlinux7 scripts]# chmod +x script51.sh
[root@mlinux7 scripts]# ./script51.sh
./script51.sh: line 3: MYFUNC: command not found
[root@mlinux7 scripts]# MYFUNC () {
> echo "Hello World"
> }
[root@mlinux7 scripts]# export -f MYFUNC
[root@mlinux7 scripts]# ./script51.sh
Hello World
```

**Note:** Observe that initially the script failed and later after defining a function in local shell and after exporting it, then if we execute the same script, it gives the output as expected.

52. Basic return command to come out from the function.

```
$ vim script52.sh
#!/bin/bash
#### Controlling a function using return command.

MYFUNC () {
    echo "Welcome to VPTS"
    return
    echo "Have a Nice Day"
}
MYFUNC
echo "Today's date is `date`"
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script52.sh
[root@mlinux7 scripts]# ./script52.sh
Welcome to VPTS
Today's date is Tue Oct 31 14:48:58 IST 2017
```

53. Script to show the return status of a function.

```
$ vim script53.sh
#!/bin/bash
### return a function with a specific status
MYFUNC() {
    cp file1 file2 >/dev/null 2>&1
    return $?
}
MYFUNC
if [ $? -eq 0 ]; then
    echo "My Function completed successfully"
    exit 0
else
    echo "My Function failed"
    exit 1
fi
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script53.sh
[root@mlinux7 scripts]# ./script53.sh
My Function failed
```

The **cp** command in the function failed due to some issue and the same status have been returned from the function to main script. Finally using the exit status we are displaying a message whether it was successful or not.

#### 54. Script to do numerical operation using functions.

```
$vim script54.sh
#!/bin/bash
##### Numerical operations using functions.
USAGE() {
    echo "Invalid Data.. Please try Again"
    exit
}
ADD() { echo "Addition=$(( $a+$b ))"; }
SUB() { echo "Subtraction=$(( $a-$b ))"; }
MUL() { echo "Multiplication=$(( $a*$b ))"; }
DIV() { echo "Division=$(( $a/$b ))"; }
read -p 'Enter 1st Value: ' a
read -p 'Enter 2nd Value: ' b
if [ -z "$a" -o -z "$b" ]; then
    USAGE
fi
read -p 'Enter the Operator[ADD|SUB|MUL|DIV]: ' op
case $op in
    ADD) ADD;;
    SUB) SUB;;
    MUL) MUL;;
    DIV) DIV;;
    *) USAGE;;
esac
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script54.sh
[root@mlinux7 scripts]# ./script54.sh
Enter 1st Value: 10
Enter 2nd Value:
Invalid Data.. Please try Again
[root@mlinux7 scripts]# ./script54.sh
Enter 1st Value: 10
Enter 2nd Value: 20
Enter the Operator[ADD|SUB|MUL|DIV]: HHH
Invalid Data.. Please try Again
[root@mlinux7 scripts]# ./script54.sh
Enter 1st Value: 10
Enter 2nd Value: 20
Enter the Operator[ADD|SUB|MUL|DIV]: ADD
Addition=30
```



TECHNO SOLUTIONS

## OPTION PARSING

### Special Variables:

The shell defines several special variables that are relevant to option parsing. In addition to these, a few variables give the status of commands that the script executes. The following table describes all of the special variables defined by the shell.

Variable	Description
<b>\$0</b>	The name of the command being executed. For shell scripts, this is the path with which it was invoked.
<b>\$n</b>	These variables correspond to the arguments with which a script was invoked. Here <i>n</i> is a positive decimal number corresponding to the position of an argument (the first argument is \$1, the second argument is \$2, and so on).
<b>\$#</b>	The number of arguments supplied to a script.
<b>\$*</b>	All the arguments are double quoted. If a script receives two arguments, \$* is equivalent to \$1 \$2.
<b>\$@</b>	All the arguments are individually double quoted. If a script receives two arguments, \$@ is equivalent to \$1 \$2.
<b>\$?</b>	The exit status of the last command executed.
<b>\$\$</b>	The process number of the current shell. For shell scripts, this is the process ID under which they are executing.
<b>\$!</b>	The process number of the last background command.

Let's see how to use a special variable

```
#!/bin/bash
echo "Script Name: $0"
echo "First Parameter : $1"
echo "Second Parameter: $2"
echo "All Values: $*"
echo "All Values: $@"
echo "Arguments Passed: $#"
```

### Usage Statements:

Another common use for \$0 is in the usage statement for a script, which is a short message informing the user how to invoke the script properly. All scripts used by more than one user should include such a message.

In general, the usage statement is something like the following:

```
echo "Usage: $0 [options] [files]"
```

### Using Basename

Currently, the message displays the entire path with which the shell script was invoked, but what is really required is the name of the shell script. You can correct this by using the basename command. The basename command takes an absolute or relative path and returns the file or directory name. Its basic syntax is

```
basename file
```

Let's see it with an example

```
[root@mlinux7 ~]# basename /usr/bin/bash  
bash
```

You can use the basename command in echo input giving it as command in command quotes.

```
$ USAGE="Usage: `basename $0` Input"  
$ echo $USAGE
```

#### Shift Command:

Using ***shift*** command, command line arguments can be accessed. This command causes the positional parameters shift to the left. Shift [n] where n defaults to 1. It is useful when several parameters need to be parsed to a script because the values are limited to only 9.

For example in early \$1=10, \$2=20, \$3=30. If we use the ***shift*** command the pointer will shift a value and \$1=20 and \$2=30. If we give a number to ***shift*** command then it will shift that many values at a time. A detailed example will be coming later in the manual.



### Lab Exercises:

55. Script to see the values parsed to the script.

```
#vim script55.sh
#!/bin/bash
##### Special Variables
echo 'echo $0= '$0 ### Name of the script
echo 'echo $1= '$1 ### First value parsed to the script
echo 'echo $2= '$2 ### Second value
echo 'echo $3= '$3 ### Third Value
echo 'echo $*= '$* ### All values parsed to script
echo 'echo ${@}= '$@ ### All values parse to script
echo 'echo $#= '$# ### Number of values parsed to script
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script55.sh
[root@mlinux7 scripts]# ./script55.sh 10 20 xyz
echo $0= ./script55.sh
echo $1= 10
echo $2= 20
echo $3= xyz
echo $*= 10 20 xyz
echo ${@}= 10 20 xyz
echo $#= 3
```

56. Script to get the use of basename and dirname commands.

```
$vim script56.sh
#!/bin/bash
### basename and dirname commands
echo 'Script Name ($0)= '$0
echo "Basename of the Script (basename \$0)= `basename $0`"
echo "Basedir of the Script (dirname \$0)= `dirname $0`"
```

#### Output:

```
[root@mlinux7 scripts]# chmod +x script56.sh
[root@mlinux7 scripts]# ./script56.sh
Script Name ($0)= ./script56.sh
Basename of the Script (basename $0)= script56.sh
Basedir of the Script (dirname $0)= .

[root@mlinux7 /]# root/scripts/script56.sh
Script Name ($0)= root/scripts/script56.sh
Basename of the Script (basename $0)= script56.sh
Basedir of the Script (dirname $0)= root/scripts
```

**Observation:** If you use **basename** you will get only the name of the script instead of the getting the complete path, and also sometimes we need the base directory and we can get that from **dirname** command.

**57. Script to do Numerical Operations using Special variables by parsing the values.**

```
$vim script57.sh
#!/bin/bash
### Numerical Operations using Special Variables.

USAGE() {
    echo -e "Usage : `basename $0` val1 val2 operator\nOperators Allowed = [ADD|SUB|MUL|DIV]\nEx : `basename $0` 10 11
ADD"
    exit 1
}
### Checking the number of values parsed and equal to 3 or not.
if [ "$#" -ne "3" ]; then
    USAGE
    fi
### Checking the given values are integers or not.
test $1 -eq 0 >/dev/null 2>&1
STAT1=$?
test $2 -eq 0 >/dev/null 2>&1
STAT2=$?
if [ $STAT1 -eq 2 -o $STAT2 -eq 2 ];then
    echo "The given values are not integers"
    USAGE
    fi
case $3 in
    ADD) OUT=$((1+$2));;
    SUB) OUT=$((1-$2));;
    MUL) OUT=$((1*$2));;
    DIV) OUT=$(echo scale=3; $1/$2|bc);;
    *) echo "Invalid Operator!!"; USAGE;;
esac
echo "Result = $OUT"
exit 0
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script57.sh
[root@mlinux7 scripts]# ./script57.sh 10 20 MUL
Result = 200
[root@mlinux7 scripts]# chmod +x script57.sh
[root@mlinux7 scripts]# ./script57.sh
Usage : script57.sh val1 val2 operator
Operators Allowed = [ADD|SUB|MUL|DIV]
Ex : script57.sh 10 11 ADD
[root@mlinux7 scripts]# ./script57.sh 10 20 ADD
Result = 30
[root@mlinux7 scripts]# ./script57.sh 10 20 MUL
Result = 200
[root@mlinux7 scripts]# ./script57.sh 30 15 DIV
Result = 2.000
```

### 58. Values parsing to functions.

```
$ vim script58.sh
#!/bin/bash
### Option parsing to functions
func1() {
echo "Function1 values"
echo 'echo $1= '$1
echo 'echo $2= '$2
}
func2() {
echo "Function2 values"
echo 'echo $1= '$1
echo 'echo $2= '$2
}
func1 ax bx
func2
```

**Output:**

```
[root@mlinux7 scripts]# vim script58.sh
[root@mlinux7 scripts]# chmod +x script58.sh
[root@mlinux7 scripts]# ./script58.sh
Function1 values
echo $1= ax
echo $2= bx
Function2 values
echo $1=
echo $2=
```



**Note:** You can see the two different functions and we are parsing the values to main script and also we are parsing the values to 1<sup>st</sup> function. Hence we are getting the values parsed in function. In the second function we are not parsing any values and trying to access the values of script i.e. \$1 and \$2, but we are getting null values as functions will have its own environment and its own values.

### 59. Script to show the usage of **tr** command to convert the lower case to upper case.

```
$vim script59.sh
#!/bin/bash
#### Script for tr command to convert small character to capital alphabets.
read -p 'Enter Some Small Alphabets: ' c
c=`echo $c|tr [a-z] [A-Z]`
echo "Converted into Capital Alphabets: $c"
```

**Output:**

```
[root@mlinux7 scripts]# chmod +x script59.sh
[root@mlinux7 scripts]# ./script59.sh
Enter Some Small Alphabets: vpts
Converted into Capital Alphabets: VPTS
```

## 60. Script for the usage of shift command.

```
$ vim script60.sh
#!/bin/bash
### shift command is used to load the values if the given values are more than
##than 9
## execute the command as shown ./script62.sh aa ab ac ad ae af ag ah ai aj ak al
echo "Before Shift Command"
echo '$1 = '$1
echo '$2 = '$2
echo '$9 = '$9
echo '$10 = '$10
#####
echo 'After using shift command'
shift
echo '$1 = '$1
echo '$2 = '$2
echo 'After using shift 9 command'
shift 9
echo '$1 = '$1
```

### **Output:**

```
[root@mlinux7 scripts]# chmod +x script60.sh
[root@mlinux7 scripts]# ./script62.sh aa ab ac ad ae af ag ah ai aj ak al
Before Shift Command
$1 = aa
$2 = ab
$9 = ai
$10 = aa0
After using shift command
$1 = ab
$2 = ac
After using shift 9 command
$1 = ak
```

## GETOPTS

Getopts obtains options and their arguments from a list of parameters that follows the standard POSIX.2 option syntax (that is, single letters proceeded by a - and possibly followed by an argument value). Typically, shell scripts use **getopts** to parse arguments passed to them. When you specify args on the **getopts** command line, **getopts** parses those arguments instead of the script command line.

The process by which **getopts** parses the options given on the command line is

1. The getopts option examines all the command line arguments, looking for arguments starting with the - character.
2. When an argument starting with the - character is found, it compares the characters following the – to the characters given in the option-siring.

3. If a match is found, the specified variable is set to the option: otherwise, variable is set to the? Character.
4. Steps 1 through 3 are repeated until all the options have been considered.

Getopts always includes with while loop in order to read all the options in a loop manner and also most of the times getopt has to include in order to read the arguments and the values parsed to arguments and to store in a constant variable.

**OPTARG:** Stores the value of the argument found by getopt.

**OPTIND:** Contains the index of the next argument to be processed.

After all the options considered were done then OPTIND is set to the number of last value.

Let's understand it with an example

```
#!/bin/bash
while getopts a:b: var
do
    case $var in
        a) a=$OPTARG;;
        b) b=$OPTARG;;
        \?) echo "Invalid Option" ; exit ;;
    esac
done
echo -e "a=$a\tb=$b"
```

If any value is missing to the argument or any other unknown options were parsed then getopt by default gives errors. In order to handle these errors we can either set OPTERR value to 0 or we can start the options in getopt with colon.

**OPTERR:** (Values 0 or 1) Indicates if Bash should display error messages generated by the getopt built-in. The value is initialized to 1 on every shell startup - so be sure to always set it to 0 if you don't want to see annoying messages!

The other way of managing the unwanted output is redirecting the error output to /dev/null.

**Using OPTERR:**

```
OPTERR=0
while getopts a:b: var
```

**Using Colon:**

```
while getopts :a:b: var
```

**Using redirection:**

```
while getopts a:b: var >/dev/null 2>&1
```

**Parsing values to functions:**

The special variables we are talking about are different for each and every command we execute, likewise different for each and individual script. Function will have its own environment created and at the same time the special variables also will be initialized for a function too in shell.

The following example narrates the usage of functions with special variables.

```
#!/bin/bash

### Defining a function

myfunc() {
    echo '$1 of function =' $1
    echo '$2 of function =' $2
    echo '$* of function =' $*
    echo '$# of function =' $#
}

myfunc 10 20
```

In the above example you can see the values parsed to the function and we accessing those values like the same as in normal script using the special variables.

**Note:** The scope of these special variables in a function is in the function itself, it cannot affect the other functions as well in the main program.

### Lab Exercises:

#### 60. Script for option parsing using GETOPTS.

```
#vim script60.sh
#!/bin/bash
### Option Parsing using GETOPTS
if [ "$#" -ne "6" ]; then
    echo "Invalid parameters"
    exit 1
fi
while getopts a:b:c: var >/dev/null 2>&1
do
    case $var in
        a) a=$OPTARG;;
        b) b=$OPTARG;;
        c) c=$OPTARG;;
        *) echo Invalid Suffix exit 1;;
    esac
done
echo "Values given to suffix a = $a"
echo "Values given to suffix b = $b"
echo "Values given to suffix c = $c"
```

### Output:

```
[root@mlinux7 scripts]# chmod +x script60.sh
[root@mlinux7 scripts]# ./script60.sh
Invalid parameters
[root@mlinux7 scripts]# ./script60.sh -a 5 -b 25 -c 30
Values given to suffix a = 5
Values given to suffix b = 25
Values given to suffix c = 30
```

**Note:** In the above example we are giving the values with a suffix in the command line and we are loading the values with **getopts** in the script.

## 61. Numerical Operations using GETOPTS.

```
$ vim script61.sh
#!/bin/bash
### Numerical Operators using GETOPTS option parsing
### Validating number of values parsed.

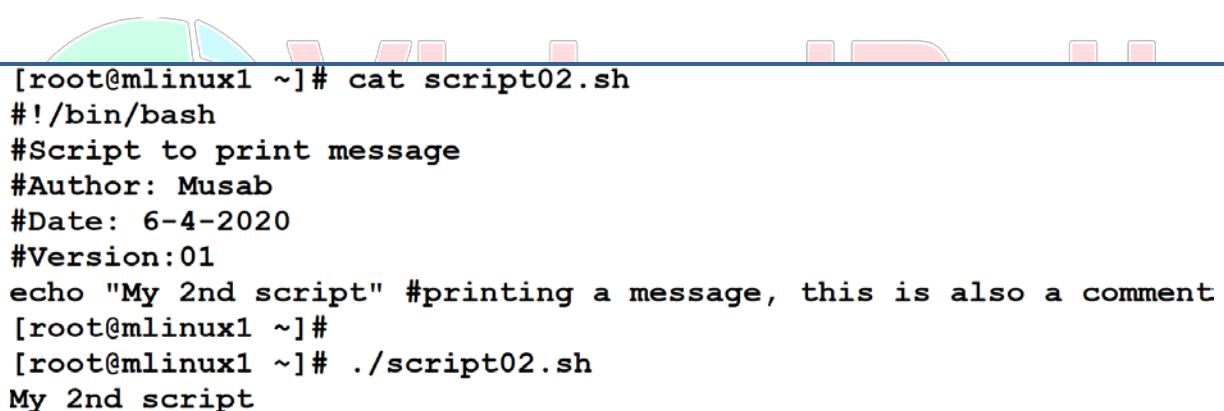
USAGE() {
    echo -e "Usage : `basename $0` -a val1 -b val2 -o operator\nEx: `basename $0` -a 10 -b 20 -o ADD"
    exit 1
}
if [ "$#" -ne "6" ]; then
    echo "Invalid Number of Arguments"
    USAGE
fi
while getopts a:b:o: var>/dev/null 2>&1
do
    case $var in
        a) a=$OPTARG;;
        b) b=$OPTARG;;
        o) op=$OPTARG;;
        *) echo "Invalid Suffix Given"; USAGE;;
    esac
done
op=`echo $op|tr [A-Z] [a-z]`
case $op in
    add) OUT=$((a+b));;
    sub) OUT=$((a-b));;
    mul) OUT=$((a*b));;
    div) OUT=`echo scale=3;$a/$b|bc`;;
    *) echo "Invalid Operator given"; USAGE;;
esac
echo "Result = $OUT"
exit 0
```

### Output:

```
[root@mlinux7 scripts]# chmod +x script61.sh
[root@mlinux7 scripts]# ./script61.sh
Invalid Number of Arguments
Usage : script61.sh -a val1 -b val2 -o operator
Ex: script61.sh -a 10 -b 20 -o ADD
[root@mlinux7 scripts]# ./script61.sh -a 10 -b 5 -o MUL
Result = 50
```

## SAMPLE CLASSROOM SCRIPTS WITH OUTPUT

```
[root@mlinux1 ~]# cat script01.sh
#!/bin/bash
echo "MY FIRST SCRIPT"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script01.sh
MY FIRST SCRIPT
```



```
[root@mlinux1 ~]# cat script02.sh
#!/bin/bash
#Script to print message
#Author: Musab
#Date: 6-4-2020
#Version:01
echo "My 2nd script" #printing a message, this is also a comment
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script02.sh
My 2nd script
```

```
[root@mlinux1 ~]# cat script03.sh
#!/bin/bash
#Script to def and use variables
name=Musab
loc=INDIA

echo "Your Name is $name"
echo "Your Location is $loc"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script03.sh
Your Name is Musab
Your Location is INDIA
[root@mlinux1 ~]#
```

```
[root@mlinux1 ~]# cat script04.sh
#!/bin/bash
#Script to create users and assign password
user=syed
pass=`date |md5sum |cut -c 1-7` #generating random passwords

echo "Adding a user $user"
useradd $user
echo "User $user added successfully"

echo $pass |passwd $user --stdin # assigning password in single attempt

echo -e "Username=$user\nPassword=$pass"

chage -d 0 $user # forcing the user to change password at next login
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script04.sh
Adding a user syed
User syed added successfully
Changing password for user syed.
passwd: all authentication tokens updated successfully.
Username=syed
Password=88dc9eb
```

```
[root@mlinux1 ~]# cat script05.sh
#!/bin/bash
#Script to do calculations
a=100
b=40

echo "Add=$((a+b)) "
echo "Sub=$((a-b)) "
echo "Mul=$((a*b)) "
echo "Div=$((a/b)) "
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script05.sh
Add=140
Sub=60
Mul=4000
Div=2
```

```
[root@mlinux1 ~]# cat mem.sh
#!/bin/bash
#Script to convert mem usage into percentages
TOTAL=`free -m |awk '/Mem/{print $2}'`
USED=`free -m |awk '/Mem/{print $3}'`
FREE=`free -m |awk '/Mem/{print $4}'`

echo "Total Memory = $TOTAL MB"
echo "Used Memory = $((($USED*100/$TOTAL)) %)"
echo "Free Memory = $((($FREE*100/$TOTAL)) %)"

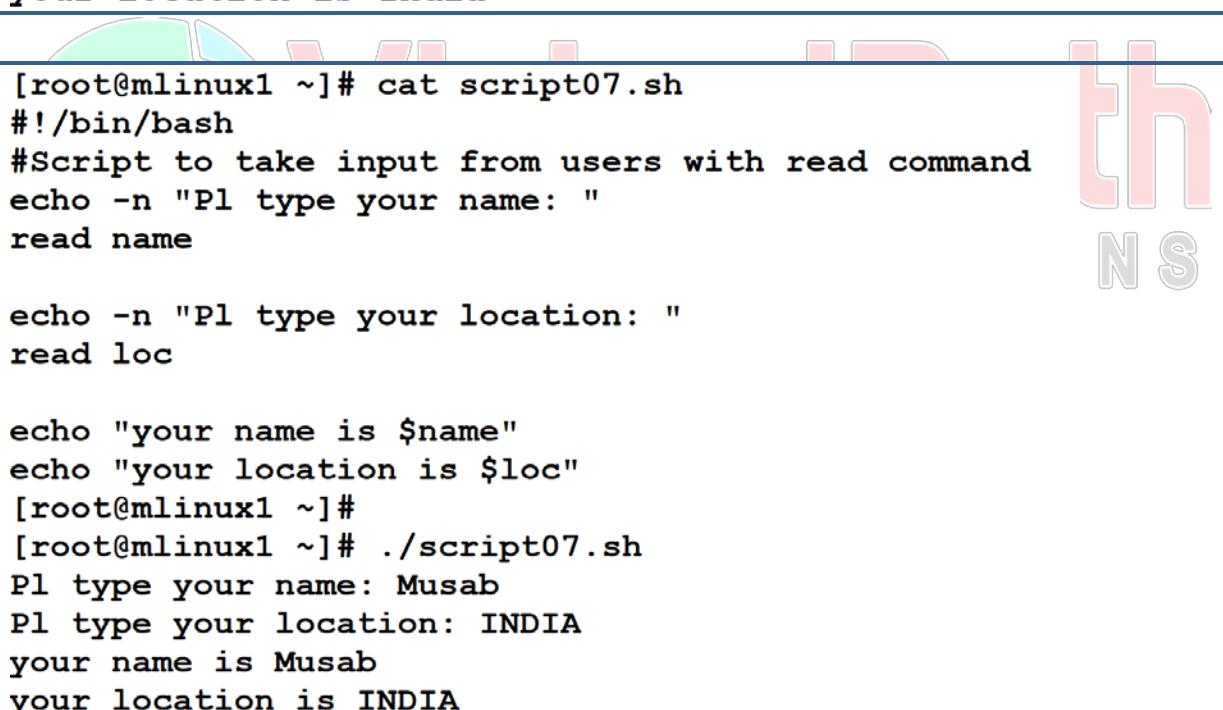
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./mem.sh
Total Memory = 1838 MB
Used Memory = 27 %
Free Memory = 52 %
```

TECHNO SOLUTIONS

```
[root@mlinux1 ~]# cat script06.sh
#!/bin/bash
#Script to take input from users with read command
echo "Pl type your name"
read name

echo "Pl type your location"
read loc

echo "your name is $name"
echo "your location is $loc"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script06.sh
Pl type your name
Musab
Pl type your location
India
your name is Musab
your location is India
```



```
[root@mlinux1 ~]# cat script07.sh
#!/bin/bash
#Script to take input from users with read command
echo -n "Pl type your name: "
read name

echo -n "Pl type your location: "
read loc

echo "your name is $name"
echo "your location is $loc"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script07.sh
Pl type your name: Musab
Pl type your location: INDIA
your name is Musab
your location is INDIA
```

```
[root@mlinux1 ~]# cat script08.sh
#!/bin/bash
#Script to take input from users with read command
read -p "Pl type your name: " name
read -p "Pl type your location: " loc

echo "your name is $name"
echo "your location is $loc"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script08.sh
Pl type your name: Musab
Pl type your location: TS-INDIA
your name is Musab
your location is TS-INDIA
```

```
[root@mlinux1 ~]# cat script09.sh
#!/bin/bash
#Script to take passwords in input using read command
read -p "Pl type user name: " user
read -s -p "Pl type password: " pass #secretly taking input

echo -e "\nYour username is $user\nYour passord is $pass"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script09.sh
Pl type user name: Musab
Pl type password:
Your username is Musab
Your passord is redhat123
```

```
[root@mlinux1 ~]# cat script10.sh
#!/bin/bash
#Script to do calculations
read -p "Pl give 1st val: " a
read -p "Pl give 2nd val: " b

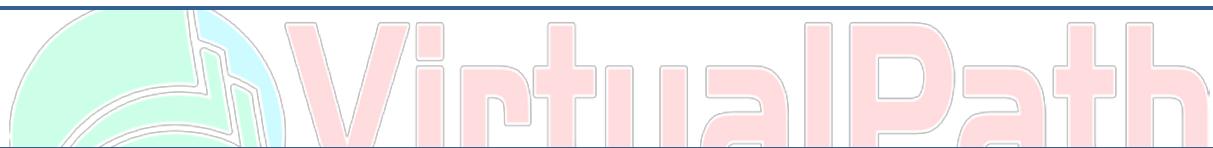
echo "Add=$((a+b))"
echo "Sub=$((a-b))"
echo "Mul=$((a*b))"
echo "Div=$((a/b))"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script10.sh
Pl give 1st val: 100
Pl give 2nd val: 20
Add=120
Sub=80
Mul=2000
Div=5
```



```
[root@mlinux1 ~]# cat script11.sh
#!/bin/bash
#script to use spl variables
echo '$0=$0'
echo '$1=$1'
echo '$2=$2'
echo '$3=$3'
echo '$*=*$'
echo '$@=$@'
echo '$#= $#'
echo '$$$=$$'
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script11.sh 10 20 30
$0=./script11.sh
$1=10
$2=20
$3=30
$*=10 20 30
$@=10 20 30
$#=3
$$=3550
```

```
[root@mlinux1 ~]# cat script12.sh
#!/bin/bash
#Script to do calculations using spl var
a=$1
b=$2

echo "Add=$((a+b)) "
echo "Sub=$((a-b)) "
echo "Mul=$((a*b)) "
echo "Div=$((a/b)) "
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script12.sh 100 20
Add=120
Sub=80
Mul=2000
Div=5
```



```
[root@mlinux1 ~]# cat script13.sh
#!/bin/bash
#Script to do calculations using function
calc(){
echo "Add=$((a+b)) "
echo "Sub=$((a-b)) "
echo "Mul=$((a*b)) "
echo "Div=$((a/b)) "
}
read -p "Pl give 1st val: " a
read -p "Pl give 2nd val: " b
calc
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script13.sh
Pl give 1st val: 100
Pl give 2nd val: 20
Add=120
Sub=80
Mul=2000
Div=5
```

```
[root@mlinux1 ~]# cat script14.sh
#!/bin/bash
#Script to do calculations with function by using return option
calc(){
echo "Add=$((a+b))"
echo "Sub=$((a-b))"
return # return is used to break the function, but cont with script
echo "Mul=$((a*b))"
echo "Div=$((a/b))"
}
read -p "Pl give 1st val: " a
read -p "Pl give 2nd val: " b
calc
echo "The script completed"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script14.sh
Pl give 1st val: 100
Pl give 2nd val: 20
Add=120
Sub=80
The script completed
```

- Creating a function to create script file and add shebang stmt.
- Adding shebang and comment line.
- Opening file in vim editor and at exit adding +x permission to file

```
[root@mlinux1 ~]# type myscript
myscript is a function
myscript ()
{
    i=`ls -1 script*|tail -1|sed -e 's/script//' -e 's/.sh//'`;
    i=$((i+1));
    echo '#!/bin/bash' > script$i.sh;
    echo '#Script to' >> script$i.sh;
    vim script$i.sh;
    chmod +x script$i.sh;
    ls --color=auto --color=auto -l script$i.sh
}
[root@mlinux1 ~]#
[root@mlinux1 ~]# tail -11 .bash_profile
myscript ()
{
    i=`ls -1 script*|tail -1|sed -e 's/script//' -e 's/.sh//'`;
    i=$((i+1));
    echo '#!/bin/bash' > script$i.sh;
    echo '#Script to' >> script$i.sh;
    vim script$i.sh;
    chmod +x script$i.sh;
    ls --color=auto -l script$i.sh
}
export -f myscript
```

```
[root@mlinux1 ~]# #myscript
[root@mlinux1 ~]# cat script15.sh
#!/bin/bash
#Script to print message
echo "File created by function"
```

```
[root@mlinux1 ~]# cat script16.sh
#!/bin/bash
#Script to use case stmt
read -p "Pl select an OS[UNIX|LINUX]: " op
case $op in
    UNIX)uname -a ;;
    LINUX)cat /etc/redhat-release ;;
    *)echo -e "\e[31mINVALID OPTION\e[0m"
esac
[root@mlinux1 ~]# ./script16.sh
Pl select an OS[UNIX|LINUX]: UNIX
Linux mlinux1.vpts.com 3.10.0-1062.el7.x86_64 #1 SMP Thu Jul 18 20:25:13
[root@mlinux1 ~]# ./script16.sh
Pl select an OS[UNIX|LINUX]: LINUX
Red Hat Enterprise Linux Server release 7.7 (Maipo)
[root@mlinux1 ~]# ./script16.sh
Pl select an OS[UNIX|LINUX]: ABC
INVALID OPTION
```

```
[root@mlinux1 ~]# cat script17.sh
#!/bin/bash
#Script to do calculations with case statement
calc(){
read -p "Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: " op
op=`echo $op|tr [a-z] [A-Z]`
case $op in
    ADD)echo "Add=$((a+b))" ;;
    SUB)echo "Sub=$((a-b))" ;;
    MUL)echo "Mul=$((a*b))" ;;
    DIV)echo "Div=$((a/b))" ;;
    EXIT)echo -e "\e[33mExiting the script\e[0m"; exit ;;
    *)echo -e "\e[31mInvalid operator selected, try again...\e[0m" ;;
esac
calc
}

read -p "Pl give 1st val: " a
read -p "Pl give 2nd val: " b
calc
[root@mlinux1 ~]# ./script17.sh
Pl give 1st val: 100
Pl give 2nd val: 20
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: add
Add=120
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: sub
Sub=80
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: mul
Mul=2000
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: div
Div=5
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: xyz
Invalid operator selected, try again...
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: exit
Exiting the script
```

```
[root@mlinux1 ~]# cat script18.sh
#!/bin/bash
#Script to use simple if
read -p "Pl give full path of file to check: " file
if [ -e "$file" ];then
    echo "File Exist"
fi
[root@mlinux1 ~]# ./script18.sh
pl give full path of file to check: /etc/passwd
File Exist
```

```
[root@mlinux1 ~]# cat script19.sh
#!/bin/bash
#Script to use if then else
read -p "Pl give full path of file to check: " file
if [ -e "$file" ];then
    echo "File Exist"
else
    echo "File doesn't Exist"
fi
[root@mlinux1 ~]# ./script19.sh
pl give full path of file to check: /etc/passwd
File Exist
[root@mlinux1 ~]# ./script19.sh
pl give full path of file to check: /xyz
File doesn't Exist
```



```
[root@mlinux1 ~]# cat script20.sh
#!/bin/bash
#Script to do calculations with case statement
calc(){
read -p "Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: " op
op=`echo $op|tr [a-z] [A-Z]`
case $op in
    ADD)echo "Add=$((a+b))" ;;
    SUB)echo "Sub=$((a-b))" ;;
    MUL)echo "Mul=$((a*b))" ;;
    DIV)echo "Div=$((a/b))" ;;
    EXIT)echo -e "\e[33mExiting the script\e[0m"; exit ;;
    *)echo -e "\e[31mInvalid operator selected, try again...\e[0m" ;;
esac
}
#String check for empty values
read -p "Pl give 1st val: " a
if [ -z "$a" ];then
    echo -e "\e[31mInvalid Input\e[0m"
    exit 1
fi
read -p "Pl give 2nd val: " b
if [ -z "$b" ];then
    echo -e "\e[31m Invalid Input\e[0m"
    exit 1
fi
calc
```

```
[root@mlinux1 ~]# cat script21.sh
#!/bin/bash
#Script to do calculations with case stat and if cond with string comparisons
calc(){
read -p "Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: " op
op=`echo $op|tr [a-z] [A-Z]`
if [ "$op" = "ADD" ];then
    echo "Add=$((a+b))"
fi

if [ "$op" = "SUB" ];then
    echo "Sub=$((a-b))"
fi

if [ "$op" = "MUL" ];then
    echo "Mul=$((a*b))"
fi

if [ "$op" = "DIV" ];then
    echo "Div=$((a/b))"
fi

if [ "$op" = "EXIT" ];then
    echo -e "\e[33mExiting the script\e[0m"; exit
fi
calc
}

read -p "Pl give 1st val: " a
if [ -z "$a" ];then
    echo -e "\e[31mInvalid Input\e[0m"
    exit 1
fi
read -p "Pl give 2nd val: " b
if [ -z "$b" ];then
    echo -e "\e[31m Invalid Input\e[0m"
    exit 1
fi

calc

[root@mlinux1 ~]# ./script21.sh
Pl give 1st val:
Invalid Input
[root@mlinux1 ~]# ./script21.sh
Pl give 1st val: 100
Pl give 2nd val:
Invalid Input
[root@mlinux1 ~]# ./script21.sh
Pl give 1st val: 100
Pl give 2nd val: 20
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: add
Add=120
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: sub
Sub=80
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: mul
Mul=2000
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: div
Div=5
Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: exit
Exiting the script
```

```
[root@mlinux1 ~]# cat script22.sh
#!/bin/bash
#Script to do ping test
read -p "Pl give the IP to test: " ip
ping -c4 $ip &>/dev/null
if [ $? -eq 0 ];then
    echo "$ip is pinging"
else
    echo "$ip is not pinging"
fi
[root@mlinux1 ~]# ./script22.sh
Pl give the IP to test: 192.168.10.20
192.168.10.20 is pinging
[root@mlinux1 ~]# ./script22.sh
Pl give the IP to test: 192.168.10.30
192.168.10.30 is not pinging
```

```
[root@mlinux1 ~]# cat script23.sh
#!/bin/bash
#Script to do calculations with case stat and elif cond with string comparisons
calc(){
read -p "Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: " op
op=`echo $op|tr [a-z] [A-Z]`
if [ "$op" = "ADD" ];then
    echo "Add=$((a+b))"

elif [ "$op" = "SUB" ];then
    echo "Sub=$((a-b))"

elif [ "$op" = "MUL" ];then
    echo "Mul=$((a*b))"

elif [ "$op" = "DIV" ];then
    echo "Div=$((a/b))"

elif [ "$op" = "EXIT" ];then
    echo -e "\e[33mExiting the script\e[0m"; exit

else
    echo -e "\e[31mInvalid operator selected, try again...\e[0m"
fi
calc
}

read -p "Pl give 1st val: " a
if [ -z "$a" ];then
    echo -e "\e[31mInvalid Input\e[0m"
    exit 1
fi
read -p "Pl give 2nd val: " b
if [ -z "$b" ];then
    echo -e "\e[31m Invalid Input\e[0m"
    exit 1
fi
calc
```

```
[root@mlinux1 ~]# cat script24.sh
#!/bin/bash
#Script to check the status of service using nested if
read -p "Pl type the name of service to check: " ser
comm=`systemctl is-active $ser`
if [ "$comm" = "active" ];then
    echo $ser is already active
else
    echo "$ser is not running"
    read -p "Would you like to start $ser [Y/N]: " input
    if [ "$input" = Y ]; then
        echo "Starting $ser"
        systemctl enable $ser --now
    else
        echo "Exiting without starting service"
    fi
fi
[root@mlinux1 ~]# ./script24.sh
Pl type the name of service to check: crond
crond is not running
Would you like to start crond [Y/N]: Y
Starting crond
Created symlink from /etc/systemd/system/multi-user.target.wants/crond.service
[root@mlinux1 ~]# systemctl is-active crond
active
[root@mlinux1 ~]# systemctl is-enabled crond
enabled
```

```
[root@mlinux1 ~]# cat script25.sh
#!/bin/bash
#Script to do calculations with case stat and elif cond with compound expr
calc(){
read -p "Pl select an operator[ADD|SUB|MUL|DIV|EXIT]: " op
op=`echo $op|tr [a-z] [A-Z]`
if [ "$op" = "ADD" ];then
    echo "Add=$((a+b))"

elif [ "$op" = "SUB" ];then
    echo "Sub=$((a-b))"

elif [ "$op" = "MUL" ];then
    echo "Mul=$((a*b))"

elif [ "$op" = "DIV" ];then
    echo "Div=$((a/b))"

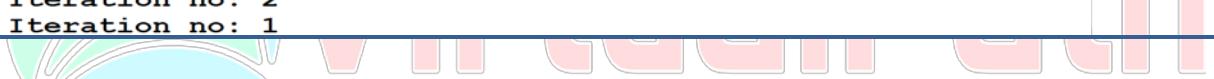
elif [ "$op" = "EXIT" ];then
    echo -e "\e[33mExiting the script\e[0m"; exit

else
    echo -e "\e[31mInvalid operator selected, try again...\e[0m"
fi
calc
}

read -p "Pl give 1st val: " a
read -p "Pl give 2nd val: " b
if [ -z "$a" -o -z "$b" ];then
    echo -e "\e[31m Invalid Input\e[0m"
    exit 1
fi
calc
```

```
[root@mlinux1 ~]# cat script26.sh
#!/bin/bash
#Script to use while loop
read -p "Pl give number of iterations: " i
while [ $i -gt 0 ]
do
    echo "Iteration no: $i"
    sleep 1
i=$((i-1))
done
[root@mlinux1 ~]# ./script26.sh
Pl give number of iterations: 5
Iteration no: 5
Iteration no: 4
Iteration no: 3
Iteration no: 2
Iteration no: 1
```

```
[root@mlinux1 ~]# cat script27.sh
#!/bin/bash
#Script to use until loop
read -p "Pl give number of iterations: " i
until [ $i -le 0 ]
do
    echo "Iteration no: $i"
    sleep 1
i=$((i-1))
done
[root@mlinux1 ~]# ./script27.sh
Pl give number of iterations: 5
Iteration no: 5
Iteration no: 4
Iteration no: 3
Iteration no: 2
Iteration no: 1
```



```
[root@mlinux1 ~]# cat script28.sh
#!/bin/bash
#Script to do calculations with case statement
calc(){
for op in ADD SUB MUL DIV EXIT
do
    case $op in
        ADD)echo "Add=$((a+b))" ;;
        SUB)echo "Sub=$((a-b))" ;;
        MUL)echo "Mul=$((a*b))" ;;
        DIV)echo "Div=$((a/b))" ;;
        EXIT)echo -e "\e[33mExiting the script\e[0m"; exit ;;
        *)echo -e "\e[31mInvalid operator selected, try again...\e[0m" ;;
    esac
done
}
read -p "Pl give 1st val: " a
read -p "Pl give 2nd val: " b
if [ -z "$a" -o -z "$b" ];then
    echo -e "\e[31m Invalid Input\e[0m"
    exit 1
fi
calc
[root@mlinux1 ~]# ./script28.sh
Pl give 1st val: 100
Pl give 2nd val: 2
Add=102
Sub=98
Mul=200
Div=50
Exiting the script
```

```
[root@mlinux1 ~]# cat script29.sh
#!/bin/bash
#Script to do calculations with select loop
calc(){
PS3="Pl Select an option: "
select op in ADD SUB MUL DIV EXIT
do
    case $op in
        ADD)echo "Add=$((a+b))" ;;
        SUB)echo "Sub=$((a-b))" ;;
        MUL)echo "Mul=$((a*b))" ;;
        DIV)echo "Div=$((a/b))" ;;
        EXIT)echo -e "\e[33mExiting the script\e[0m"; exit ;;
        *)echo -e "\e[31mInvalid operator selected, try again...\e[0m" ;;
    esac
done
}
read -p "Pl give 1st val: " a
read -p "Pl give 2nd val: " b
if [ -z "$a" -o -z "$b" ];then
    echo -e "\e[31m Invalid Input\e[0m"
    exit 1
fi
calc
[root@mlinux1 ~]# ./script29.sh
Pl give 1st val: 100
Pl give 2nd val: 20
1) ADD
2) SUB
3) MUL
4) DIV
5) EXIT
Pl Select an option: 1
Add=120
Pl Select an option: 2
Sub=80
Pl Select an option: 5
Exiting the script
```

```
[root@mlinux1 ~]# cat script30.sh
#!/bin/bash
#Script to use getopt
while getopts a:b:o: opt
do
    case $opt in
        a)a=$OPTARG;;
        b)b=$OPTARG;;
        o)o=$OPTARG;;
    esac
done
echo -e "\na=$a\nb=$b\no=$o\n"

case $o in
    ADD)echo "Add=$((a+b))" ;;
    SUB)echo "Sub=$((a-b))" ;;
    MUL)echo "Mul=$((a*b))" ;;
    DIV)echo "Div=$((a/b))" ;;
    *)echo -e "\e[31mInvalid operator selected, try again...\e[0m" ;;
esac
[root@mlinux1 ~]# ./script30.sh -a 10 -b 2 -o ADD
a=10
b=2
o=ADD
Add=12
```

```
[root@mlinux1 ~]# cat script31.sh
#!/bin/bash
#Script to use getopt with usage stmt
usage(){
    echo "Usage: `echo $0` -a val -b val -o operator[ADD|SUB|MUL|DIV]"
    echo "Usage: `echo $0` -a 100 -b 20 -o ADD"
    exit
}
if [ $# -le 0 ];then
    usage
fi

while getopt a:b:o: opt
do
    case $opt in
        a)a=$OPTARG;;
        b)b=$OPTARG;;
        o)o=$OPTARG;;
    esac
done

echo -e "\na=$a\tb=$b\to=$o\n"

case $o in
    ADD)echo "Add=$((a+b))" ;;
    SUB)echo "Sub=$((a-b))" ;;
    MUL)echo "Mul=$((a*b))" ;;
    DIV)echo "Div=$((a/b))" ;;
    *)echo -e "\e[31mInvalid operator selected, try again...\e[0m" ;;
esac

[root@mlinux1 ~]# ./script31.sh
Usage: ./script31.sh -a val -b val -o operator[ADD|SUB|MUL|DIV]
Usage: ./script31.sh -a 100 -b 20 -o ADD
[root@mlinux1 ~]# ./script31.sh -a 100 -b 20 -o MUL

a=100      b=20      o=MUL

Mul=2000
```

```
[root@mlinux1 ~]# cat script32.sh
#!/bin/bash
#Script to use trap option
trap '' 1 2 20 ##trapping a signal, not to disturb the script
echo "Backup of system started"
sleep 1
echo "Backup is in progress, pl don't disturb the script"
sleep 10
echo -e "\nBackup completed successfully"
[root@mlinux1 ~]#
[root@mlinux1 ~]# ./script32.sh
Backup of system started
Backup is in progress, pl don't disturb the script
^C^C^C^C^Z^Z^Z^Z^Z^Z
Backup completed successfully
```

## SOME ADDITIONAL IMPORTANT STUFF AND SCRIPTS

### Here Document or heredoc

A here document (or **heredoc**) is a way of getting text input into a script without having to feed it in from a separate file. If you've got a small quantity of data you don't expect to change often, it's a quick and tidy way of keeping script and data together. For more complicated data or scripts, separating the two is still usually a better idea.

The basic version of this in bash looks like this:

```
#!/bin/bash
cat << EOF
several lines of
my data
listed here
EOF
```

#### ***Output***

```
several lines of
my data
listed here
```

A heredoc can also be used to comment or inactive multiple lines instead of using "#". An example for commenting multiple lines with heredoc is as follows.

```
#!/bin/bash
### Numerical operations using while loop.
read -p 'Pl Enter Number for iterations: ' i
<<EOF if [ -n $i ]; then ##### from here
        test $i -eq 0 >/dev/null 2>1
        if [ $? -eq 2 ]; then
                echo "Pl enter only integers"
                exit 2
        fi
else
        echo "Pl enter a value."
        exit 1
fi ##### till here got commented
EOF
while [ $i -gt 0 ]
do
        read -p 'Enter the 1st value: ' a
        read -p 'Enter the 2nd value : ' b
        if [ -z $a -o -z $b ]; then
                echo "Invalid Input"
                exit 1
        fi
```

## SOME USEFUL SCRIPTS

1. Script to download data from ftp server using here document

```
#!/bin/bash
#scrip to download multiple files from ftp
server=192.168.10.81 ##server address
user=ftp                ##user name
pass=anything           ##password
dir=pub/down
file=*
#Giving instructions to connect to ftp server and downloading data
ftp -n $server <<FTP ##Start of heredoc
quote USER $user
quote PASS $pass
bin
prompt
hash
cd $dir
mget $file
FTP    ##End of heredoc
```

2. Script to take backup of a filesystem and storing in a directory

```
$vim backup.sh
#!/bin/bash
#Script to take backup of /etc filesystem
clear
echo "Backup of etc started"
sleep 2
tar -zcvf /opt/etc.tar.gz /etc
if [ $? -eq 0 ];then
    clear
    echo "Backup is Successful"
else
    clear
    echo "Backup failed"
fi
```

3. Script to take backup of /etc and transfer it to other server

```
$vim bkp2.sh
#!/bin/bash
##SCRIPT TO TAKE BACKUP
src=/etc          ##file to be backed up
dest=/opt         ## destination to store backup
server=192.168.10.61    ##Server to transfer backup
date=`date +%Y%m%d`
clear
echo "BACKUP OF /etc/ IS STARTING"
sleep 2
tar -czvf $dest/$src-$date.tgz $src
if [ $? -eq 0 ];then
    clear
    echo "BACKUP IS SUCCESFULL"
else
    echo "BACKUP FAILED"
    exit
fi
sleep 2
echo "COPYING THE FILE ON 61 SERVER"
scp $dest$src* $server:/opt
if [ $? -eq 0 ];then
    echo "copying done successfully"
else
    echo "Copying failed"
fi
```

**Note:** if there is a trusted relationship or password less login is configured the files will be automatically transferred, else it will wait for manual entry of password.

4. Taking backup of the user given input file, storing on user defined destination and transferring to other server.

```
#!/bin/bash
##SCRIPT TO TAKE BACKUP
server=192.168.10.61
date=`date +%Y%m%d`
clear
read -p 'Pl enter full path of the file to backup: ' file
read -p 'Pl enter the destination to store the backup: ' dest
if [ -z $file -o -z $dest ]; then
    echo "Invalid Input"
    exit
fi
echo "BACKUP IS STARTING"
sleep 2
```

```

tar -czvf $dest/$file-$date.tgz $file
if [ $? -eq 0 ];then
    clear
    echo "BACKUP IS SUCCESFULL"

else
    echo "BACKUP FAILED"
    exit
fi
sleep 2
echo "COPYING THE FILE ON 61 SERVER"
scp $dest/$file* $server:/opt
if [ $? -eq 0 ];then
    echo "copying done successfully"
else
    echo "Copying failed"
fi

```

**Note:** if there is a trusted relationship or password less login is configured the files will be automatically transferred, else it will wait for manual entry of password.

##### 5. Script to monitor CPU usage and send mails as per critical and warning situation

```

$vim cpumon.sh
#!/bin/bash
##PurPose: Real time CPU Utilization Monitoring Shell Script
##Date: 13th Nov 2017
HOSTNAME=`hostname`
PATHS="/"
WARNING=90
CRIT=98
CAT=/bin/cat
MAILER=/bin/mail
CRITmailto="YOUREMAIL@DOMAIN.COM" ##replace your email for critical alerts
mailto="YOUREMAIL@DOMAIN.COM"      ##replace your email for warning alerts
mkdir -p /var/log/cpuhistory
LOGFILE=/var/log/cpuhistory/hist-`date +%h%d%y`.log
touch $LOGFILE
for path in $PATHS
do
CPU_LOAD=`top -b -n 2 -d1 | grep "Cpu(s)" | tail -n1 | awk '{print $2}' | awk -F. '{print $1}'` 
if [ -n "$WARNING" -a -n "$CRIT" ]; then
    if [ "$CPU_LOAD" -ge "$WARNING" -a "$CPU_LOAD" -lt "$CRIT" ]; then
        echo " `date "+%F %H:%M:%S"` WARNING - $CPU_LOAD on Host
$HOSTNAME" >> $LOGFILE
        echo "CPU Load is Warning $CPU_LOAD on $HOSTNAME" | $MAILER -s
"CPU Load is Warning $CPU_LOAD on $HOSTNAME" $mailto
        exit 1
    fi
fi
done

```

```

elif [ "$CPU_LOAD" -ge "$CRIT" ]; then
    echo "`date "+%F %H:%M:%S"` CRITICAL - $CPU_LOAD on $HOSTNAME" >>
$LOGFILE
    echo "CPU Load is Critical $CPU_LOAD on $HOSTNAME" | $MAILER -s "CPU Load is
Critical $CPU_LOAD on $HOSTNAME" $CRITmailto
    exit 2
else
    echo "`date "+%F %H:%M:%S"` OK - $CPU_LOAD on $HOSTNAME" >> $LOGFILE
    exit 0
fi
done

```

*My dear students/aspirants, scripting is all about logic and creativity. There are many situations in which you can use scripting for automation, monitoring and other important activities. It's all about your necessity and requirements which you can put in a script and make your work easy. After all "**Necessity is the mother of all inventions**". So, keep inventing and exploring the power of scripting, you'll never be bored of it and will always be surprised and admire it. –Musabuddin Syed.*

**TECHNO SOLUTIONS**