LINUX

Reference Guide

With Lab Exercises

Student Name: _____

Batch: _____

Branch: _____

Contents

INTRODUCTION TO LINUX……

RHEL 6 BASIC GRAPHICAL INSTALLATION……

BASIC COMMANDS……

MANAGING FILE SYSTEMS AND PARTITIONS……

LOGICAL VOLUME MANAGEMENT (LVM)……

USER AND GROUP ADMINISTRATION……

NETWORK CONFIGURATION AND TROUBLESHOOTING……

MANAGING SELINUX (BASICS OF SELINUX)……

BOOTING PROCEDURE AND KERNEL PARAMETERS……

JOB AUTOMATION……

ADMINISTRATING REMOTE SYSTEM……

ENHANCED USER SECURITY WITH SUDO……

SOFTWARE MANAGEMENT ……

BACKUP AND RESTORE USING TAR AND GZIP……

MANAGE INSTALLED SERVICES……

MANAGING PROCESSES……

FTP (FILE TRANSFER PROTOCOL) SERVER……

NFS (NETWORK FILE SYSTEM) SERVER……

SAMBA SERVER……

DNS (DOMAIN NAME SYSTEM) SERVER……

WEB SERVER (APACHE)……

KICKSTART AND NETWORK INSTALLATIONS……

INTRODUCTION TO LINUX

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,Seven etc.

Multi User - Multi Tasking Operating System

These 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" programing 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.

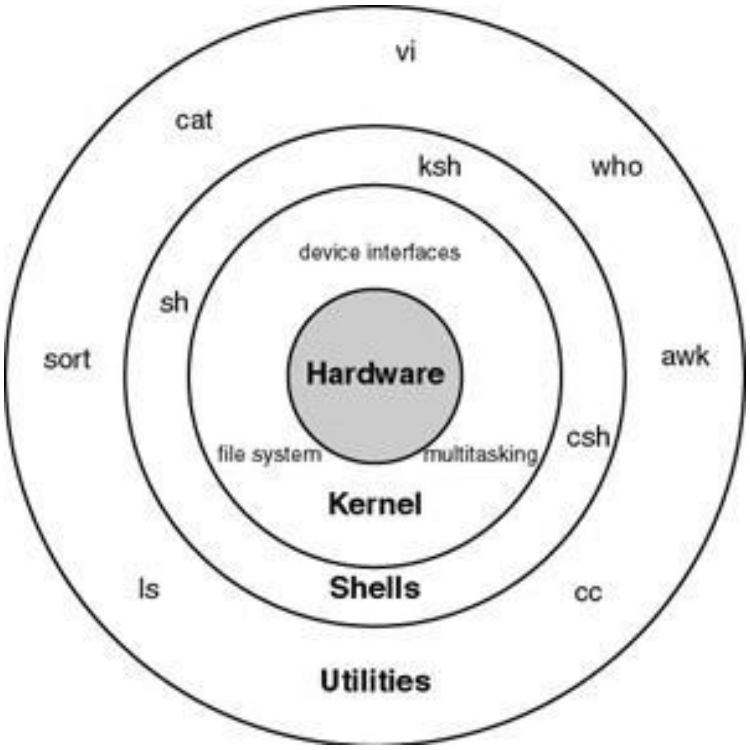
Why Linux?

- 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 SYSTEM

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

/etc 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 is containing variable data like mails, log files

/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

RHEL 6 BASIC GRAPHICAL INSTALLATION

Minimum and Recommended Requirements to install RHEL 6 are:

Hardware	Recommended Requirement for RHEL6-32BIT	Minimum Requirement for RHEL6-32 BIT	Recommended Requirement for RHEL6-64BIT	Minimum Requirement for RHEL6-64BIT
PROCESSOR	AMD/INTEL DUAL CORE	AMD/INTEL P IV	AMD/INTEL CORE 2 DUO	AMD/INTEL DUAL CORE
MOTHER BOARD	NORMAL	NORMAL	VT ENABLED	VT ENABLED
RAM	1 GB	384-512 MB	2 GB	768-1GB
HARD DISK	20 GB	15 GB	40 GB	20 GB

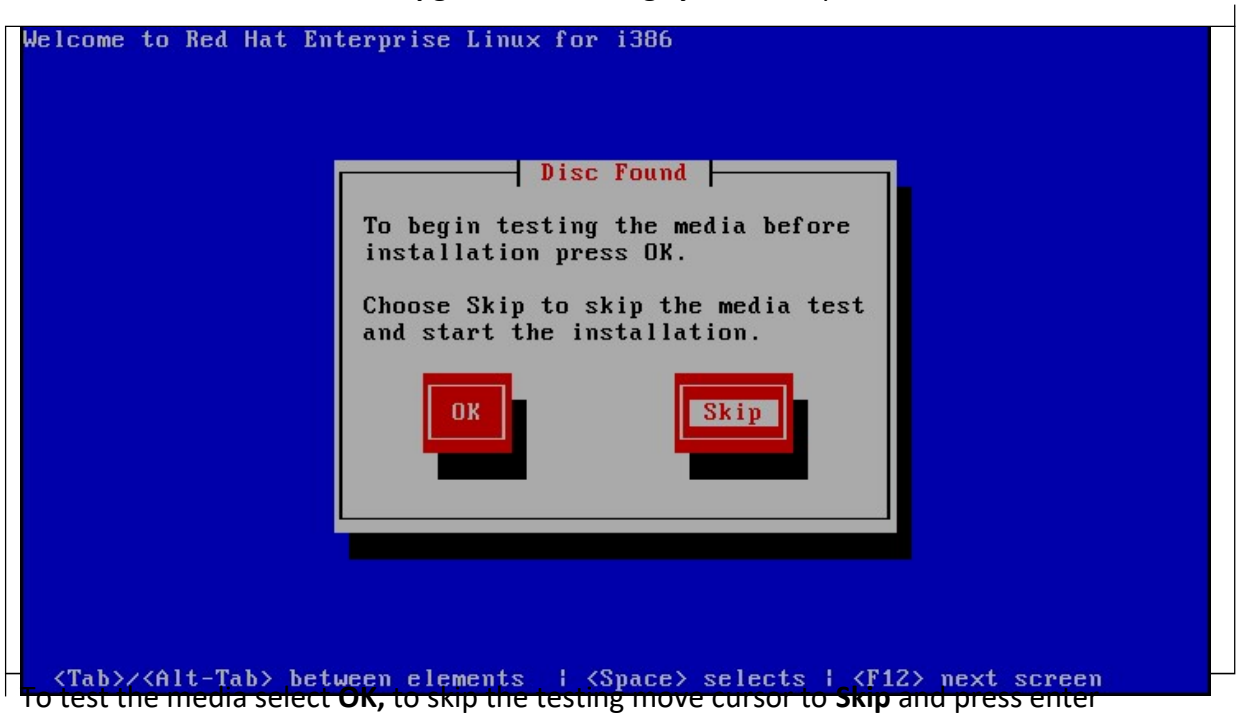
Minimum Partition creation and sizes for basic installation

Partition Name	Size For 32 Bit	Size For 64 Bit
/ (root)	8 to 10 GB	15 to 20 GB
/boot	200 MB	200 MB
SWAP	Twice of RAM	Twice of RAM

Installing RHEL6 with above specification

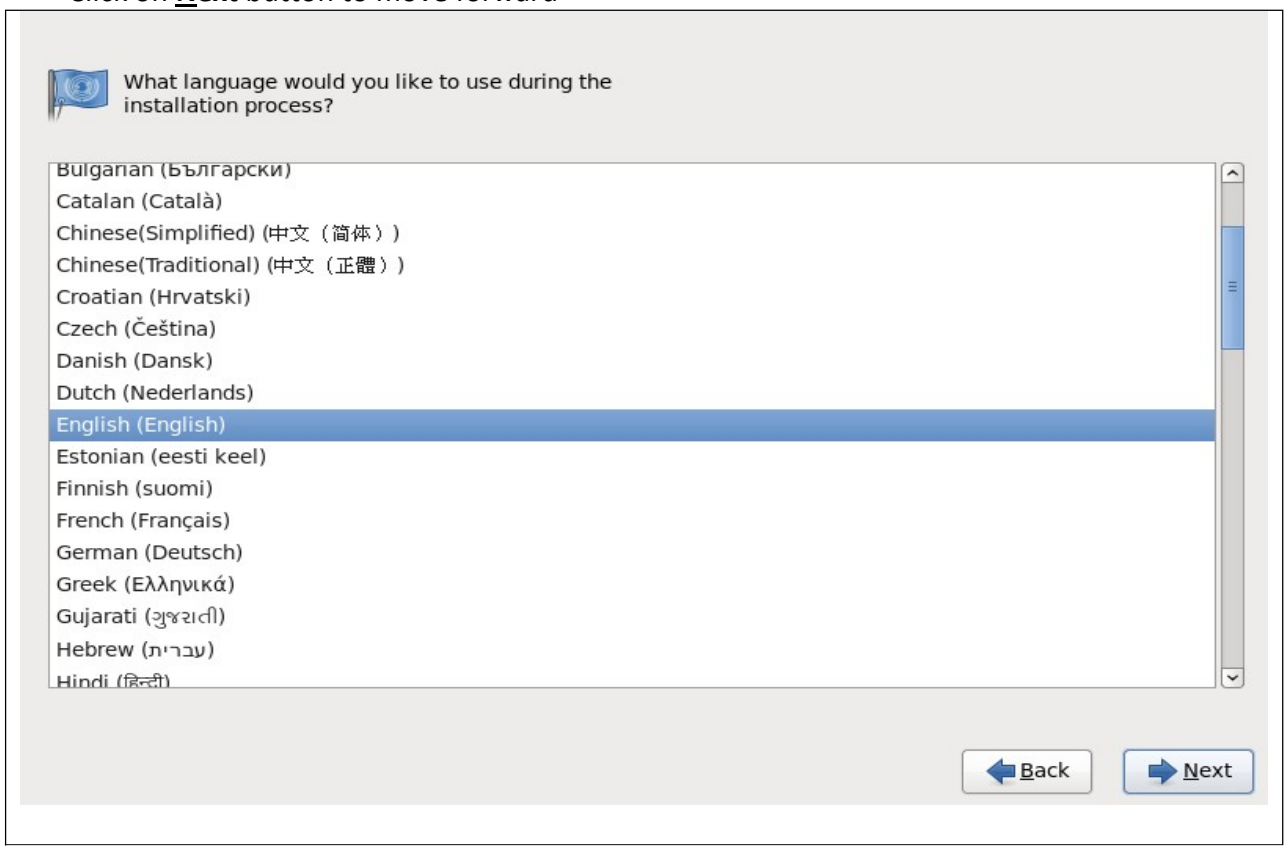
- Enter into BIOS setting and make CD/DVD Drive as first boot device
- Make sure that VT (Virtual Technology) is enabled for RHEL6-64 bit systems
- Insert the RHEL 6 CD/DVD into CD/DVD drive and boot the system
- If booted from CD/DVD Rom the following screen will be displayed

- Move the cursor to **Install or upgrade an existing system** and press **Enter**

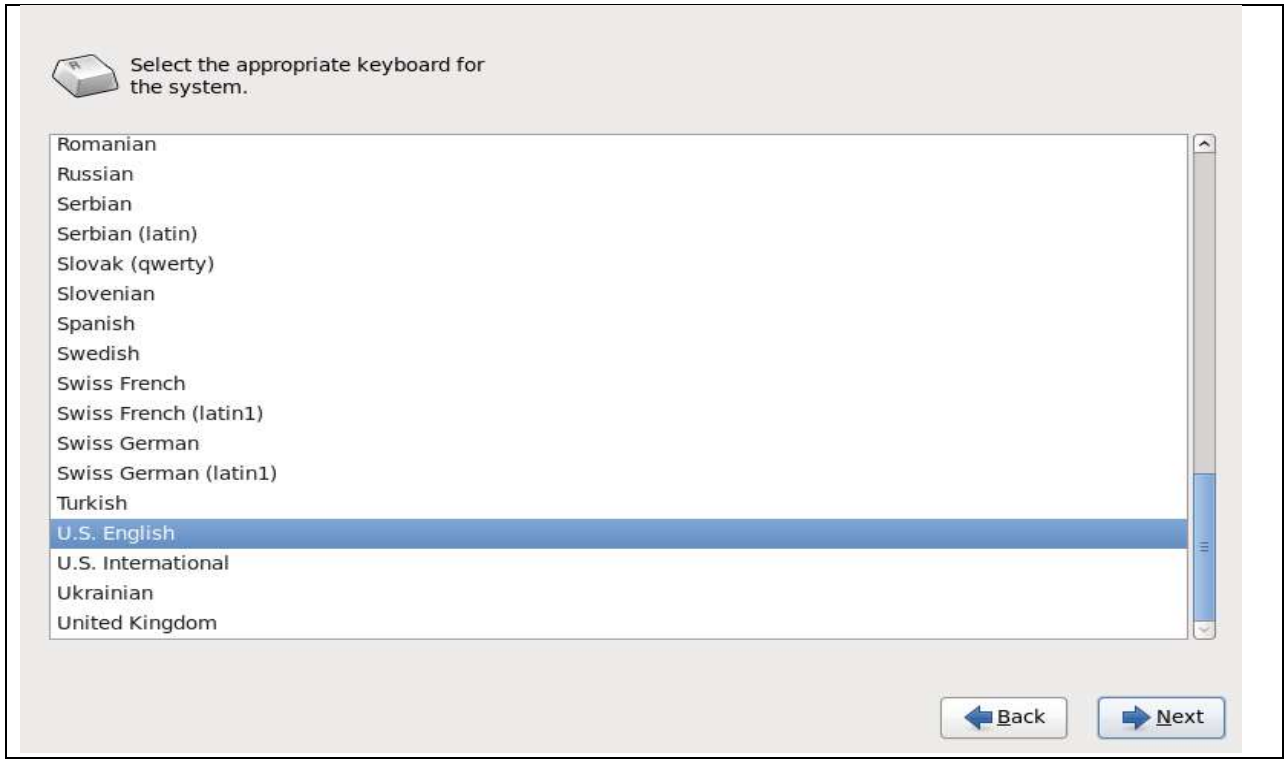


- To test the media select **OK,** to skip the testing move cursor to **Skip** and press enter

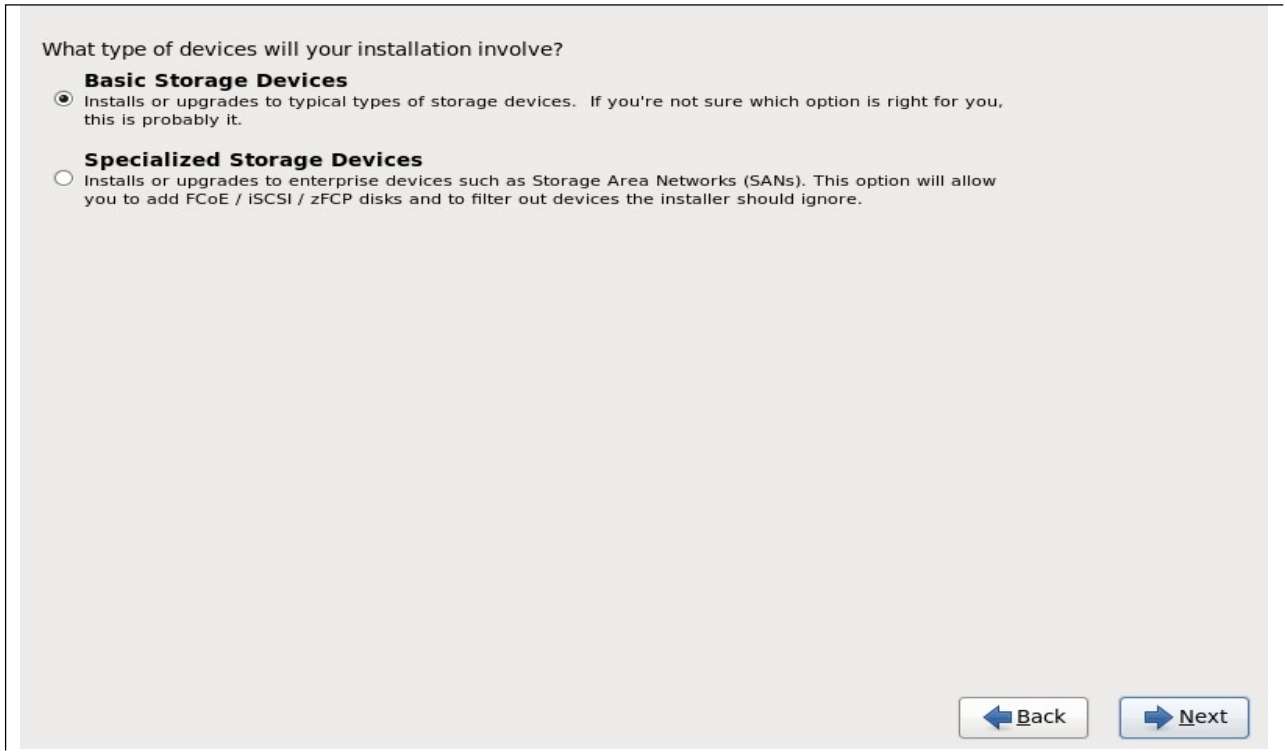
- Click on **Next** button to move forward



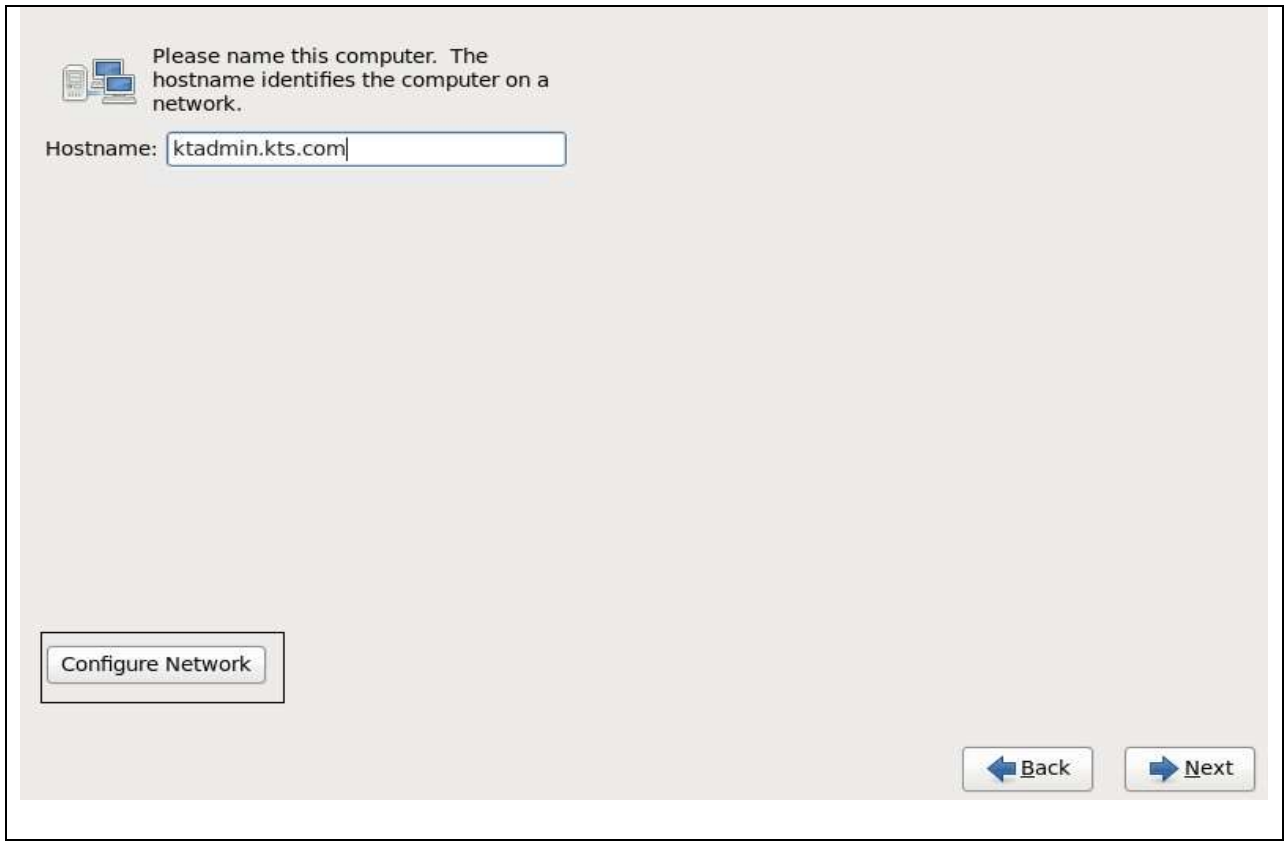
- Select your desired language, usually **English.** Click **Next** to continue



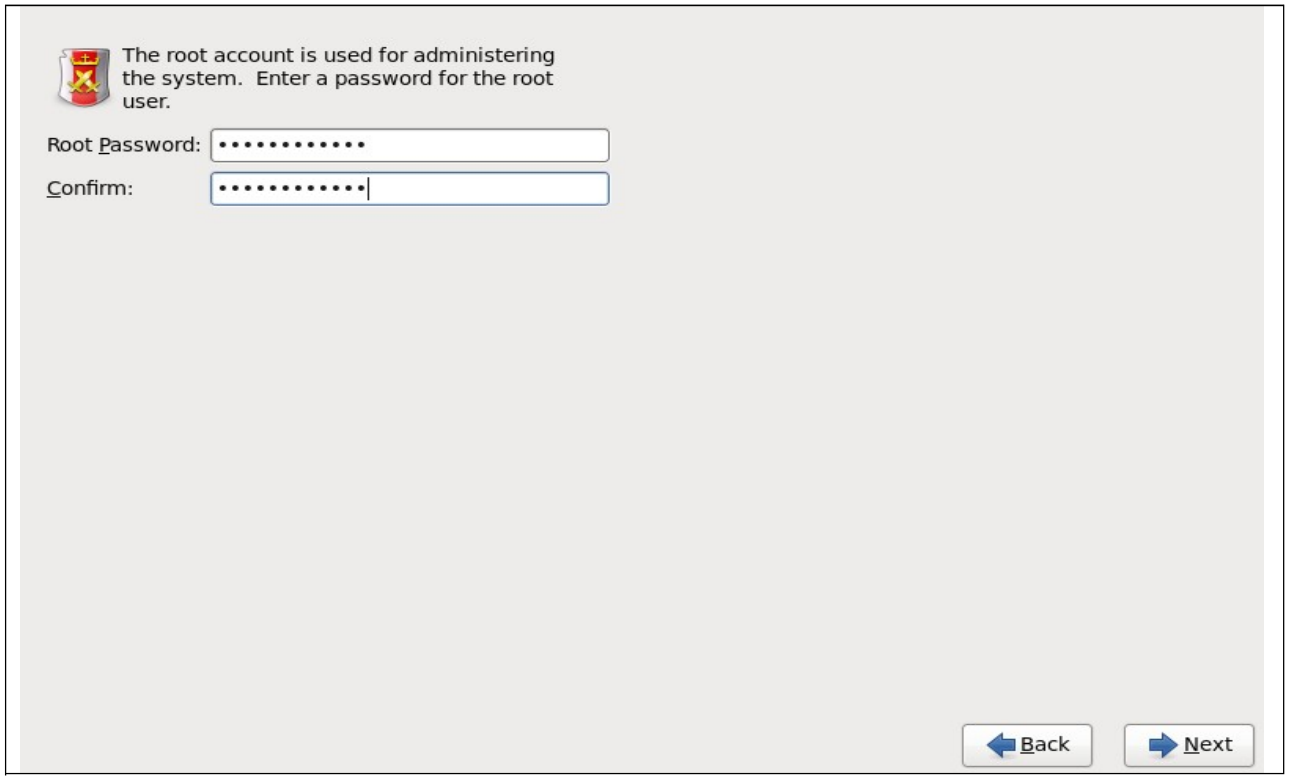
- Select the keyboard type as required usually **U.S English**, click **Next** to continue



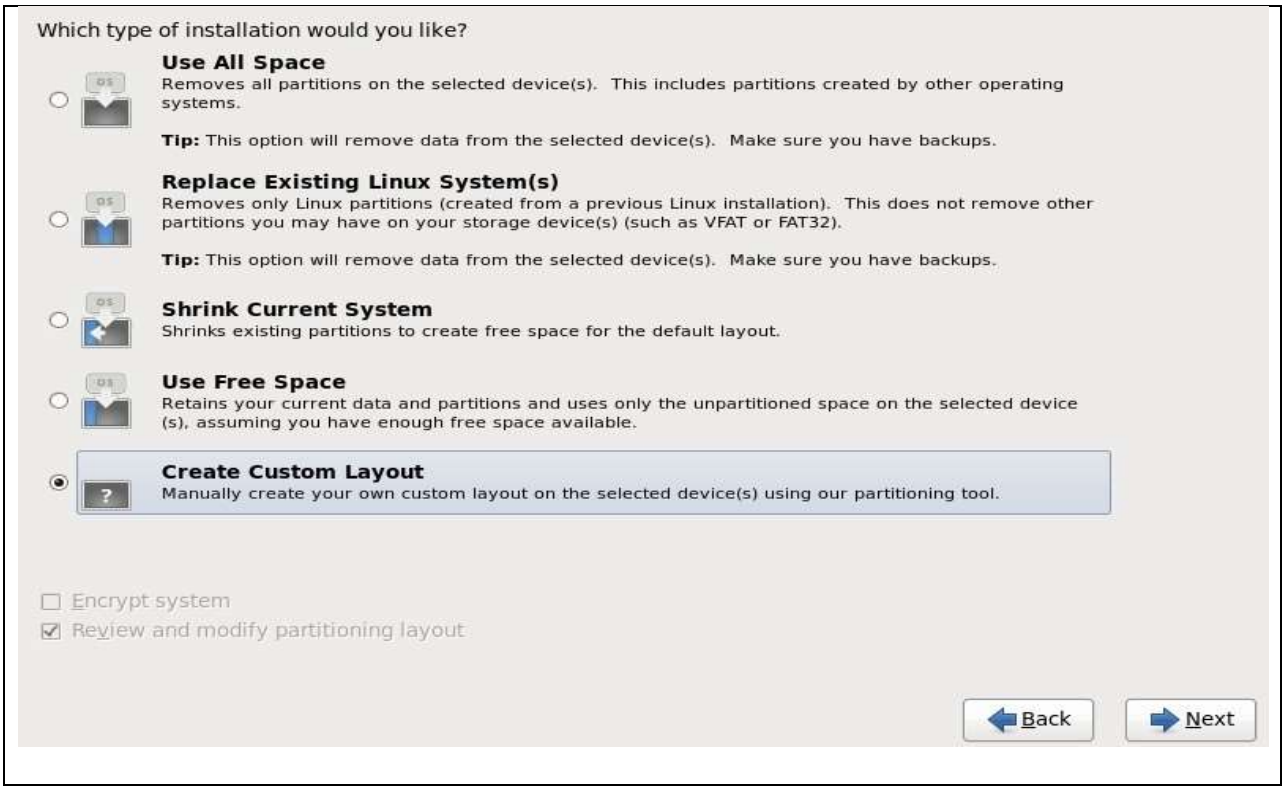
- Select the type of storage for the Computer. Click **Next** to continue



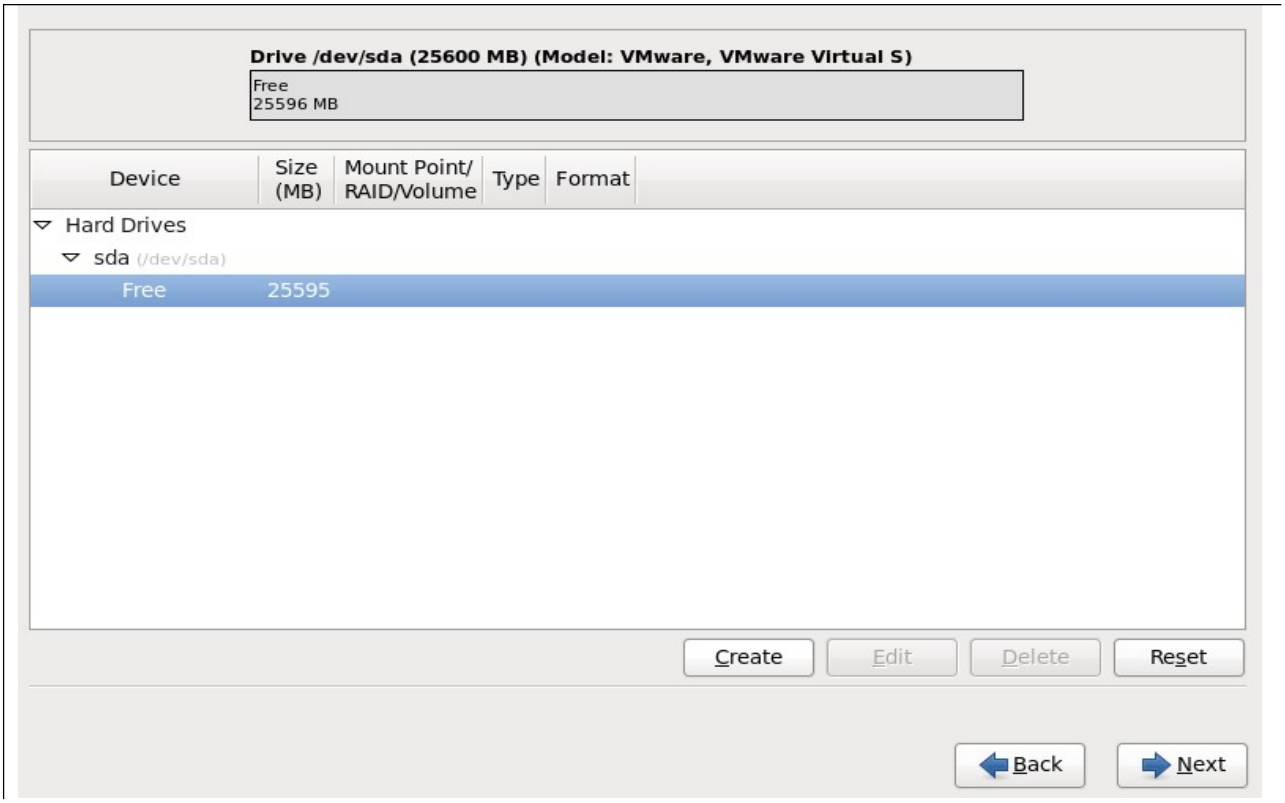
- Assign a hostname to the system, if wish to give ip address click on **Configure Network,** else Click **Next** to continue
- Select the nearest city in your Time Zone and Click on **Next** to continue



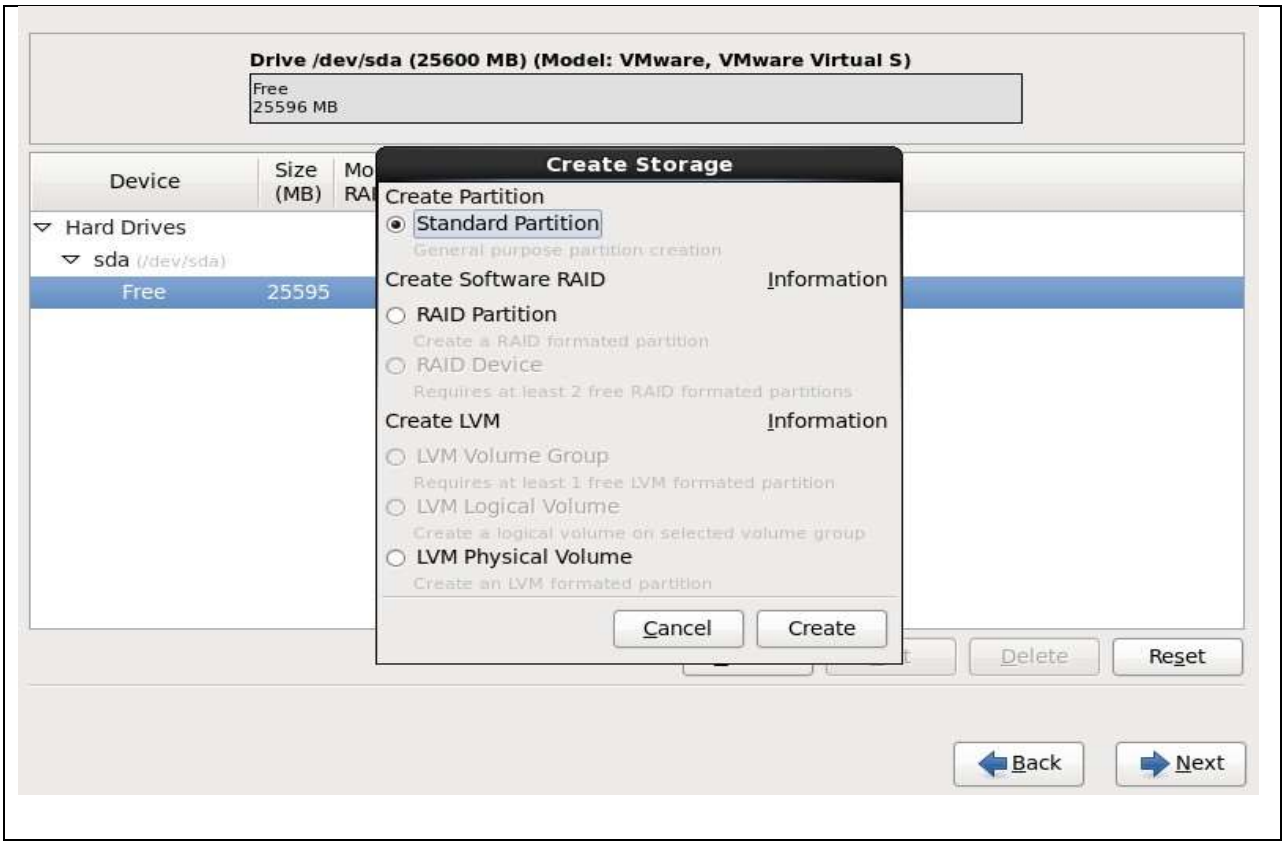
- Assign some password for **root,** then click on **Next** to continue



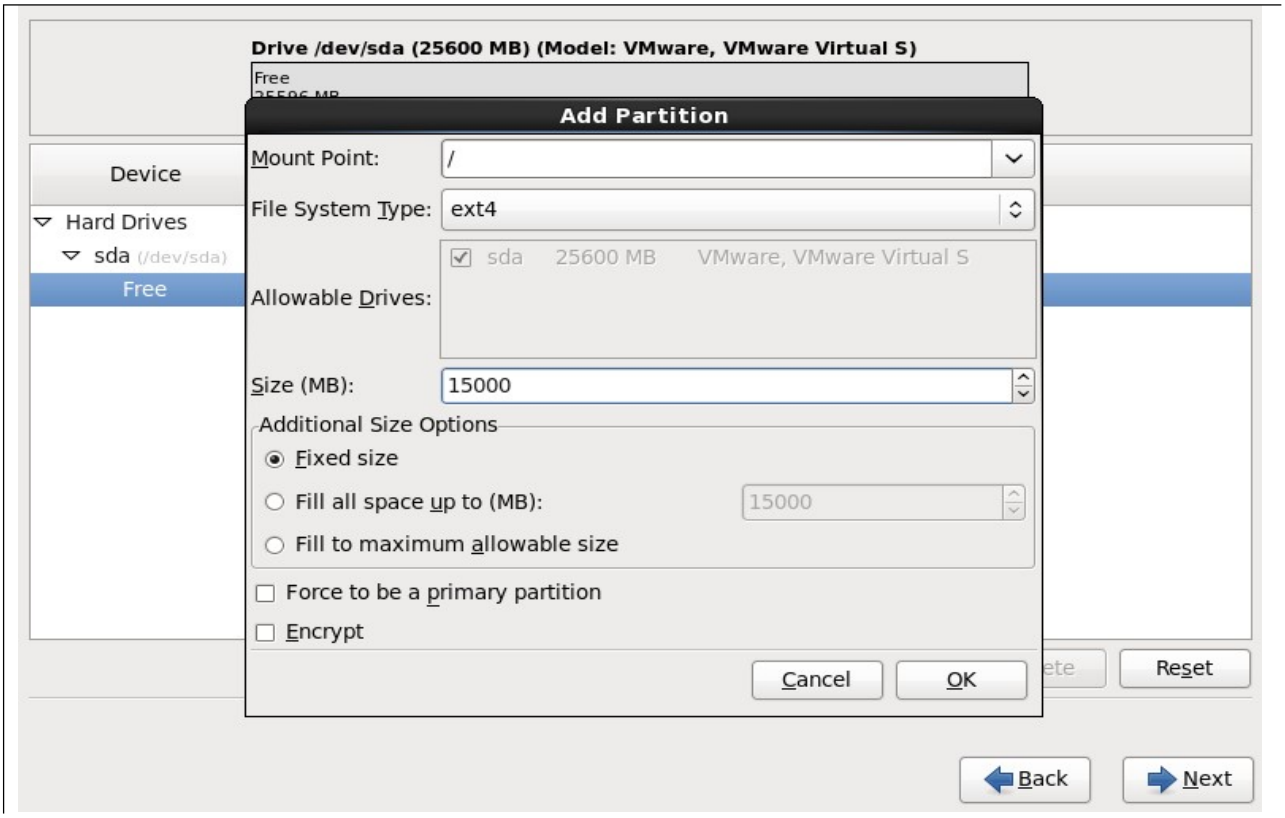
- Select the type of partitioning you want, to create your own partitions with custom sizes, select **Create Custom Layout** and click on **Next** to continue



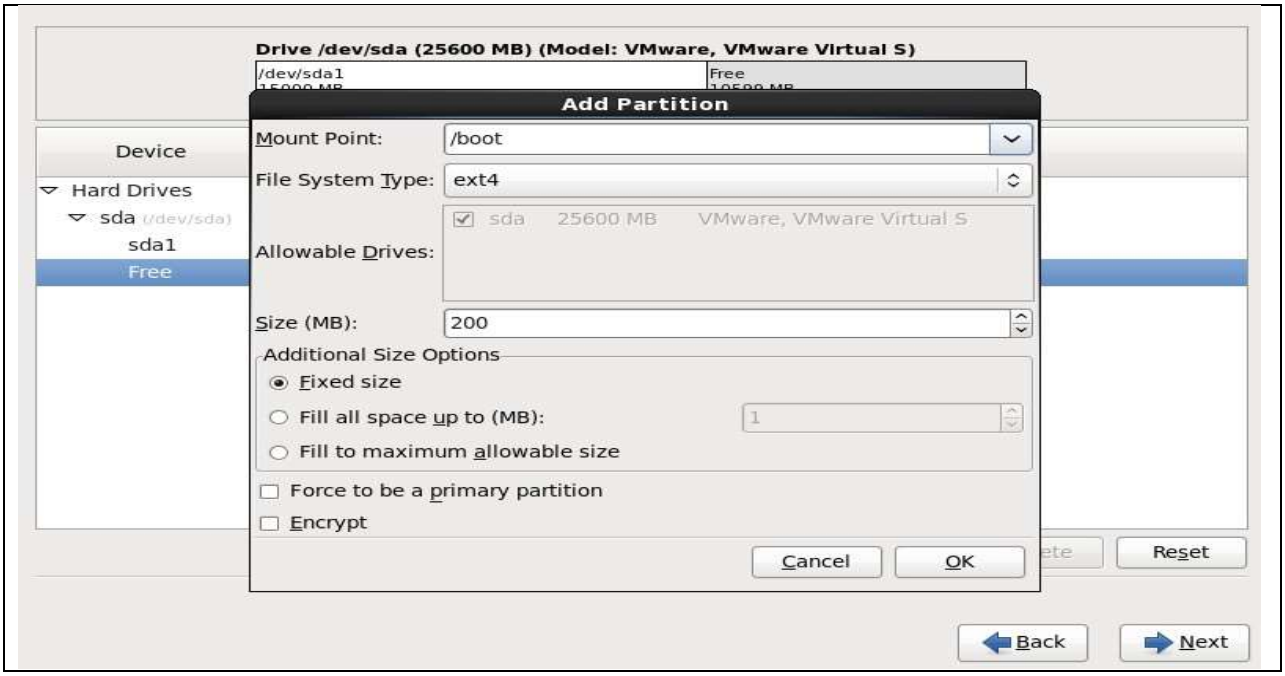
- Click on the **Free** space, then click on **Create** to create your own partitions


Drive /dev/sda (25600 MB) (Model: VMware, VMware Virtual S)
Free
25596 MB
Device
Size (MB)
Hard Drives
sda (/dev/sda)
Free
25595
Create Storage
Create Partition
Standard Partition
General purpose partition creation
Create Software RAID
Information
RAID Partition
Create a RAID formated partition
RAID Device
Requires at least 2 free RAID formated partitions
Create LVM
Information
LVM Volume Group
Requires at least 1 free LVM formated partition
LVM Logical Volume
Create a logical volume on selected volume group
LVM Physical Volume
Create an LVM formated partition
Cancel
Create
Delete
Reset
Back
Next

- Check the box beside **Standard Partition,** Click on **Create** to continue.

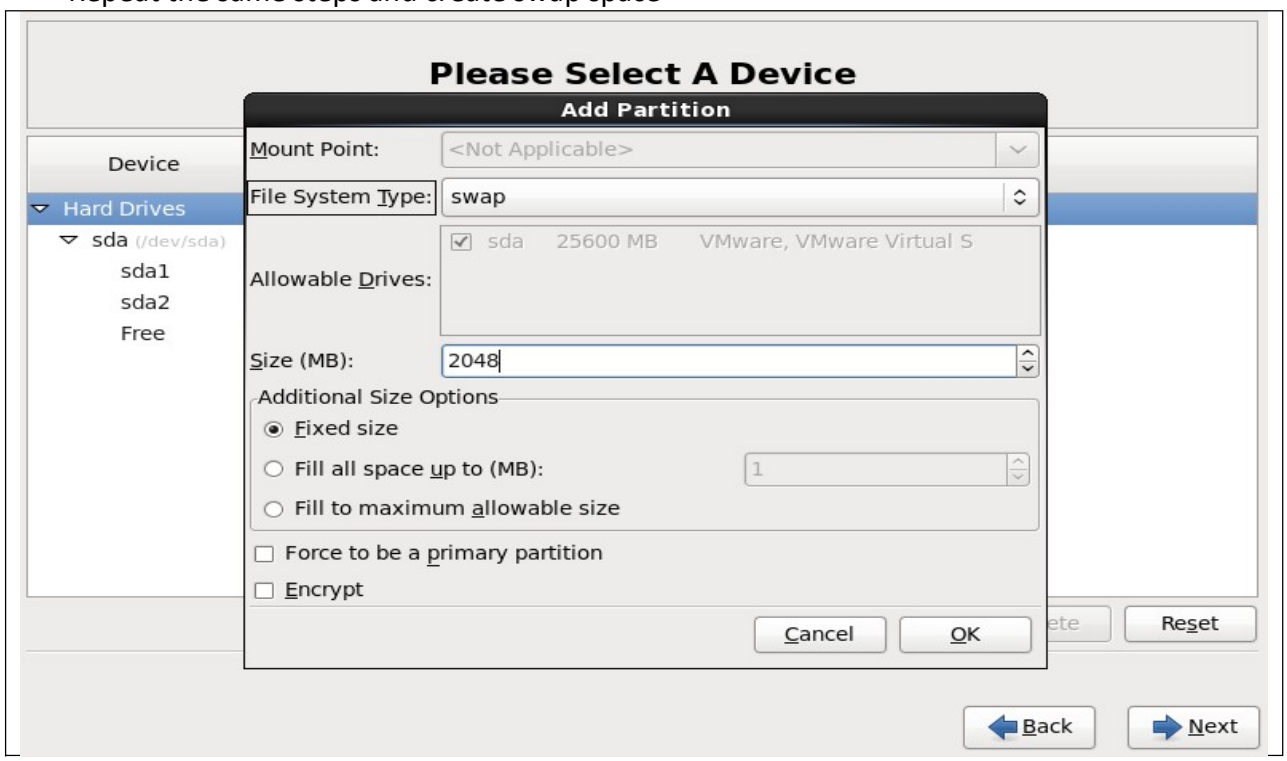


- Select / from **Mount Point** Box, give the size you wish for it and click on **OK** to create it.
- Select the **Free** space again and click on **Create** to create another partition. Also Check the box beside **Standard Partition,** Click on **Create** to continue

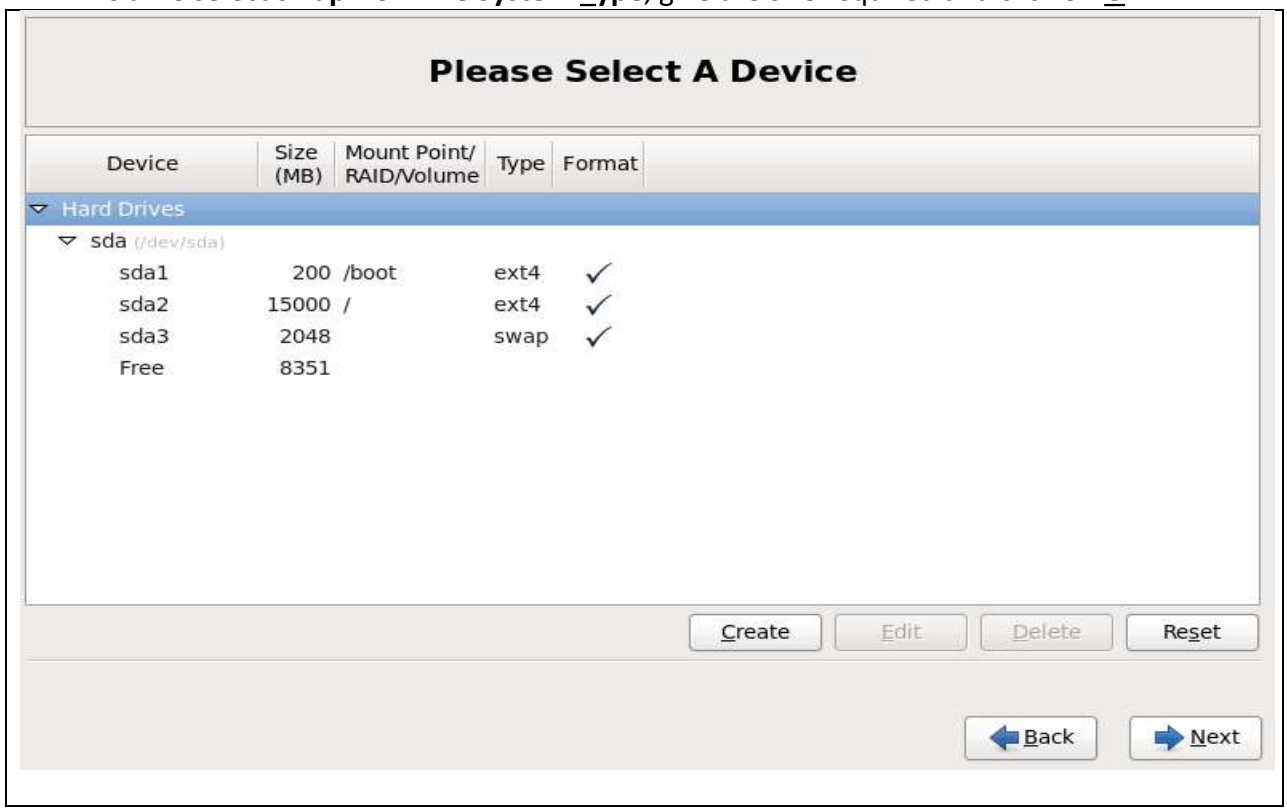


- Select **/boot** from **Mount Point** Box, give the size 200 MB for it and click on **OK** to create it.

- Repeat the same steps and create swap space



- This time select **swap** from **File System Type**, give the size required and click on **OK**

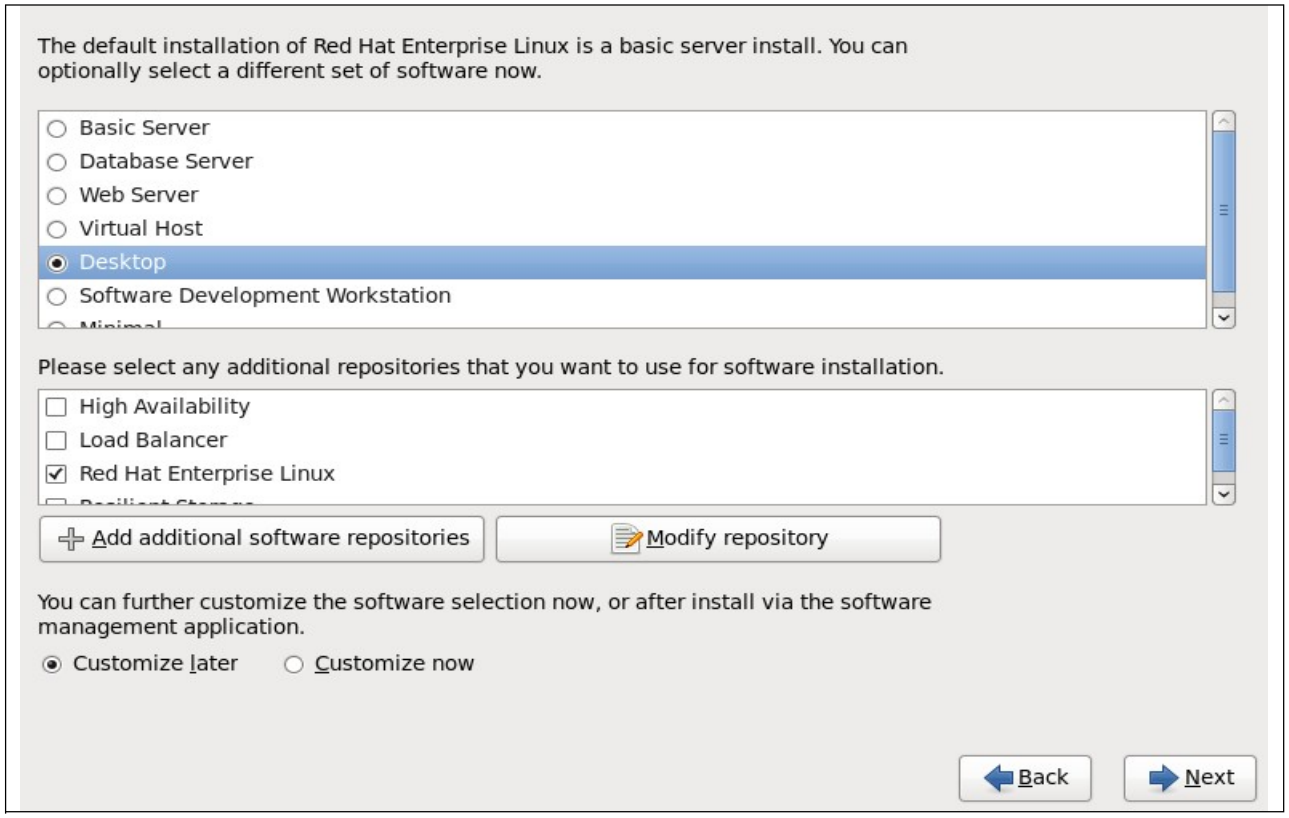


- Verify the partition and click on **Next** to continue with it.

- Click on **Format** to format the partition and continue with it.

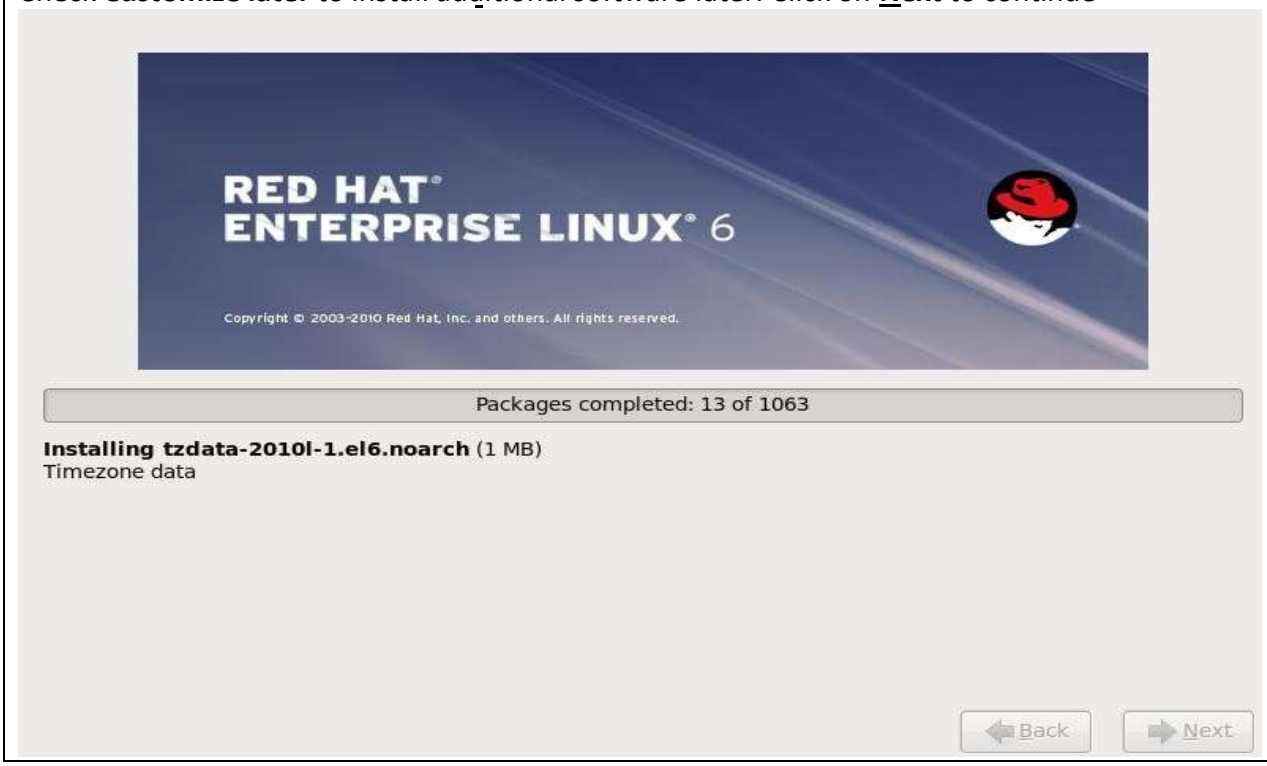
- Click on **Write changes to disk** to continue, if wish make changes click on **Go back.**

- To change the name of boot loader select **Edit** and assign new name to it.
- To assign password to boot loader check the box beside **Use boot loader password** and assign a password to it.
- To keep all as default, just click on **Next** button to continue.

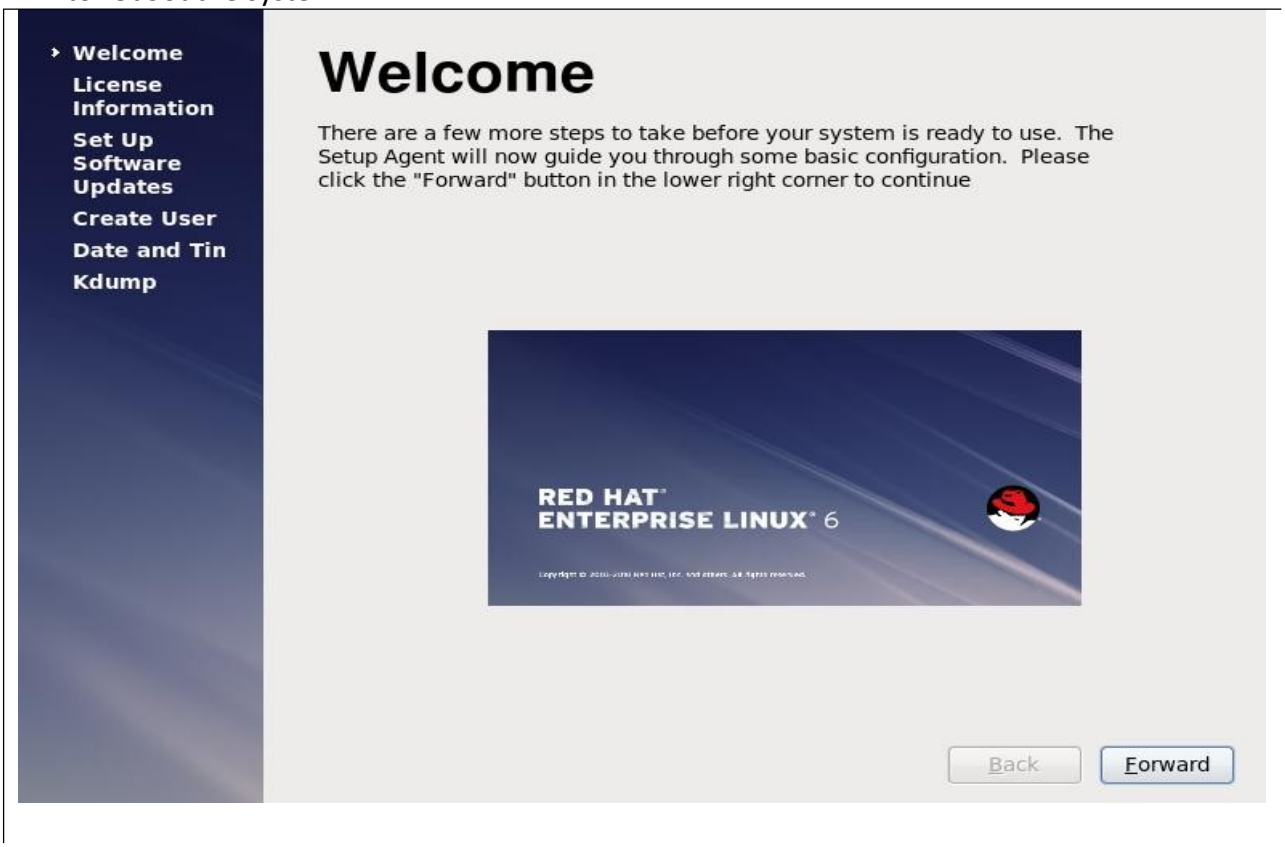


- Select **Desktop** to have a graphical environment in RHEL6.

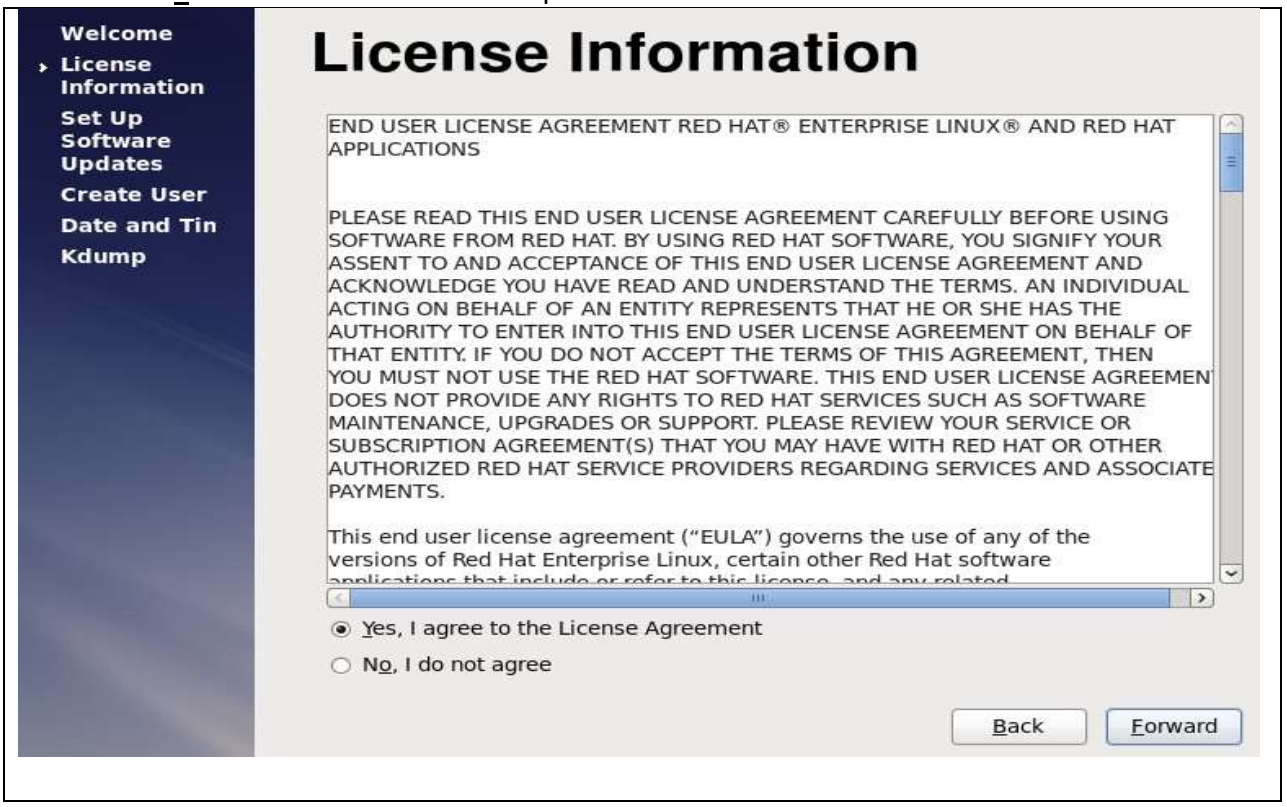
Check **Customize later** to install additional software later. Click on **Next** to continue



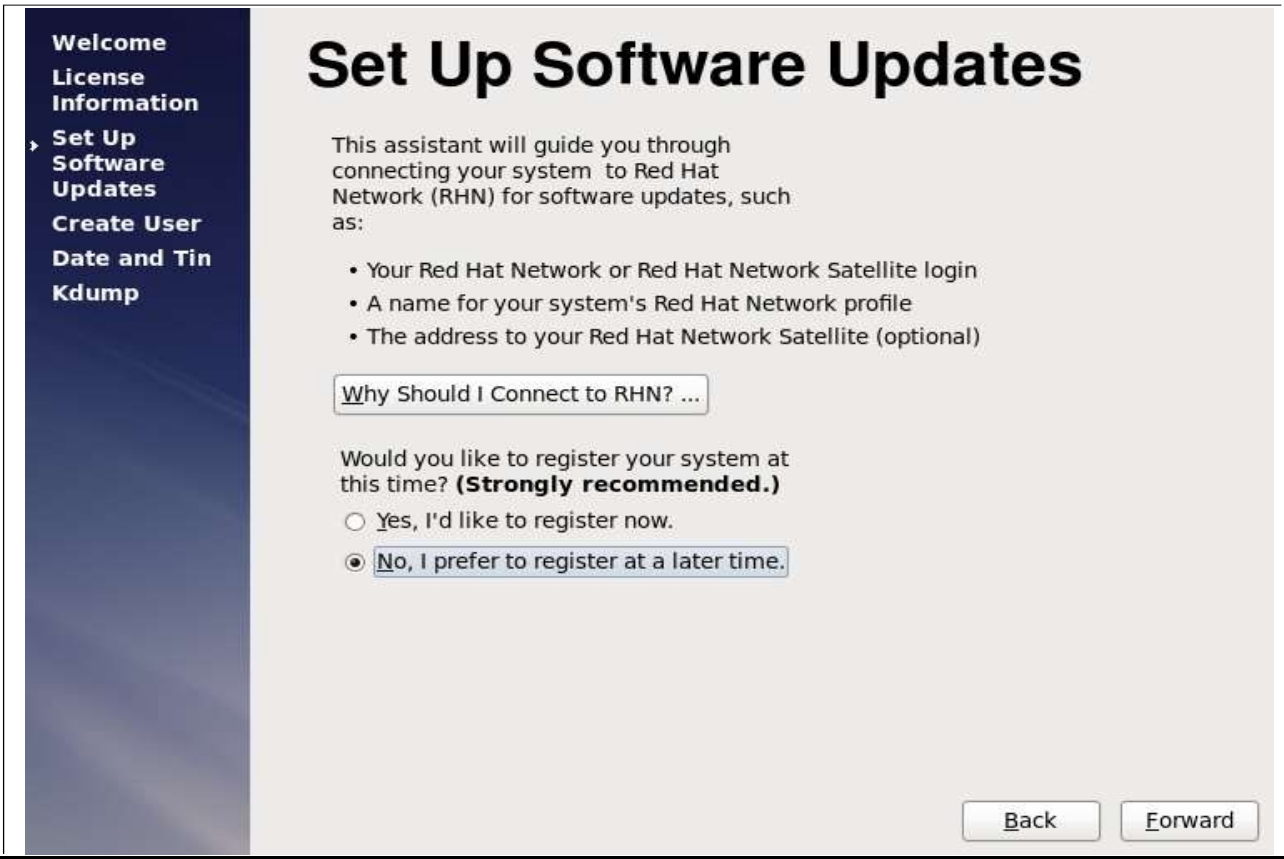
- Now sit back and relax until the installation is completed
- When above prompt is displayed, remove the CD/DVD from the drive and click on **Reboot** to reboot the system.



- Click on **Forward** to move to next step.



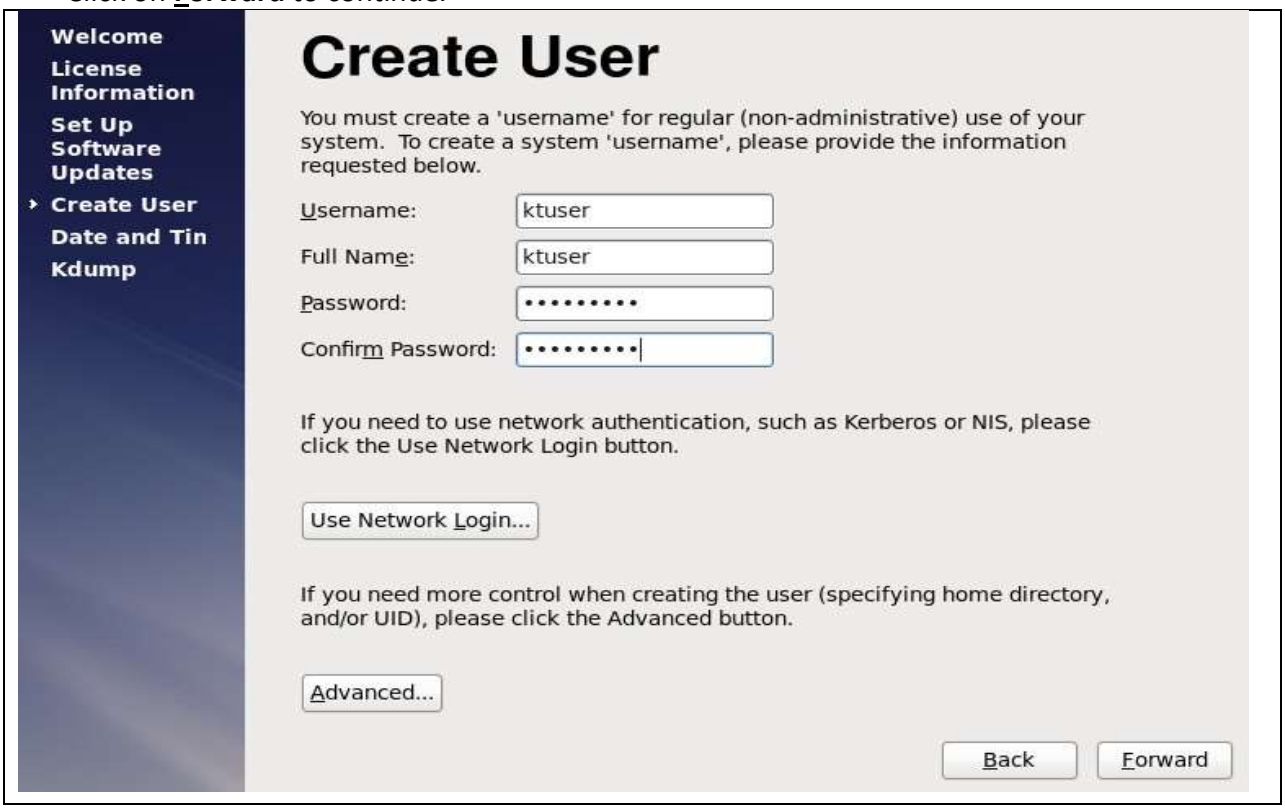
- Accept the license agreement and click on **Forward** to continue



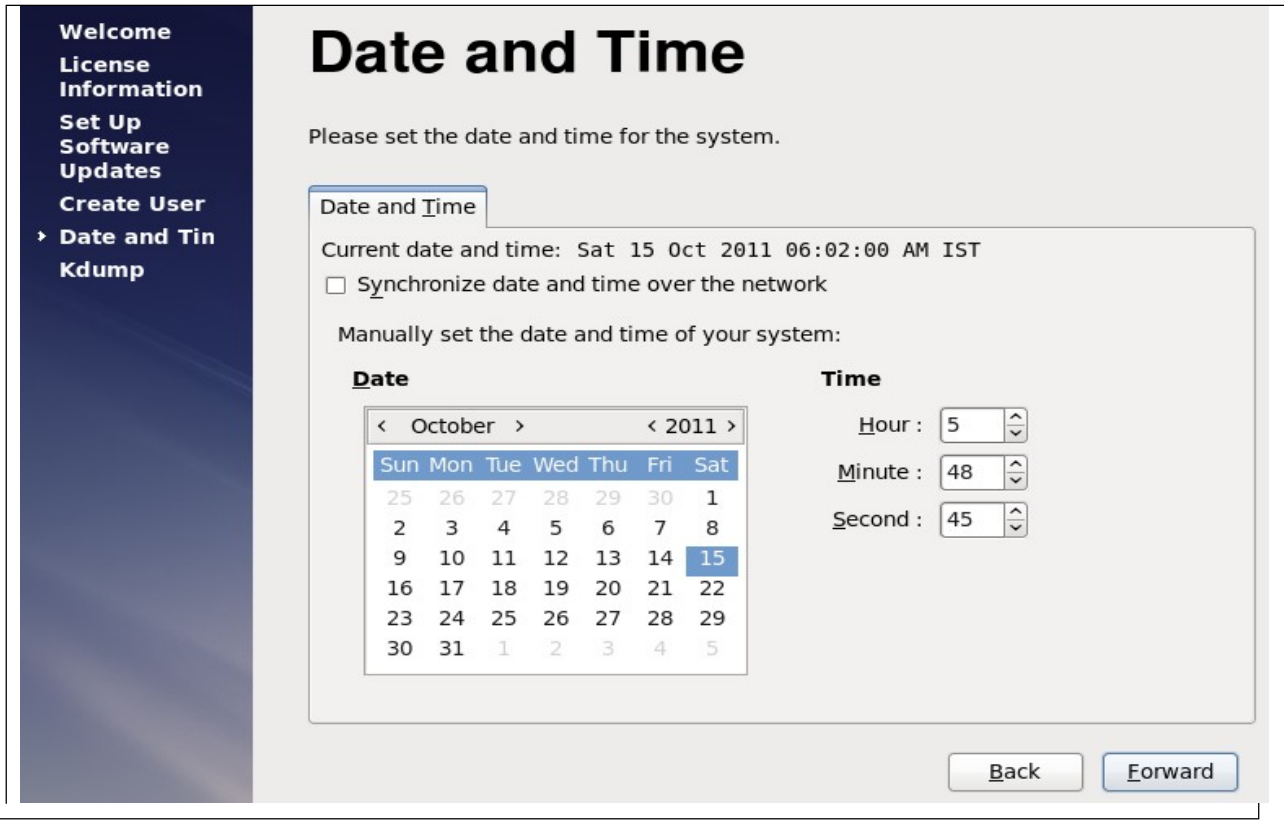
- Check **No, I prefer to register at a later time.** to skip the registration and click on **Forward.**

- Click on **No thanks,** to move to next step

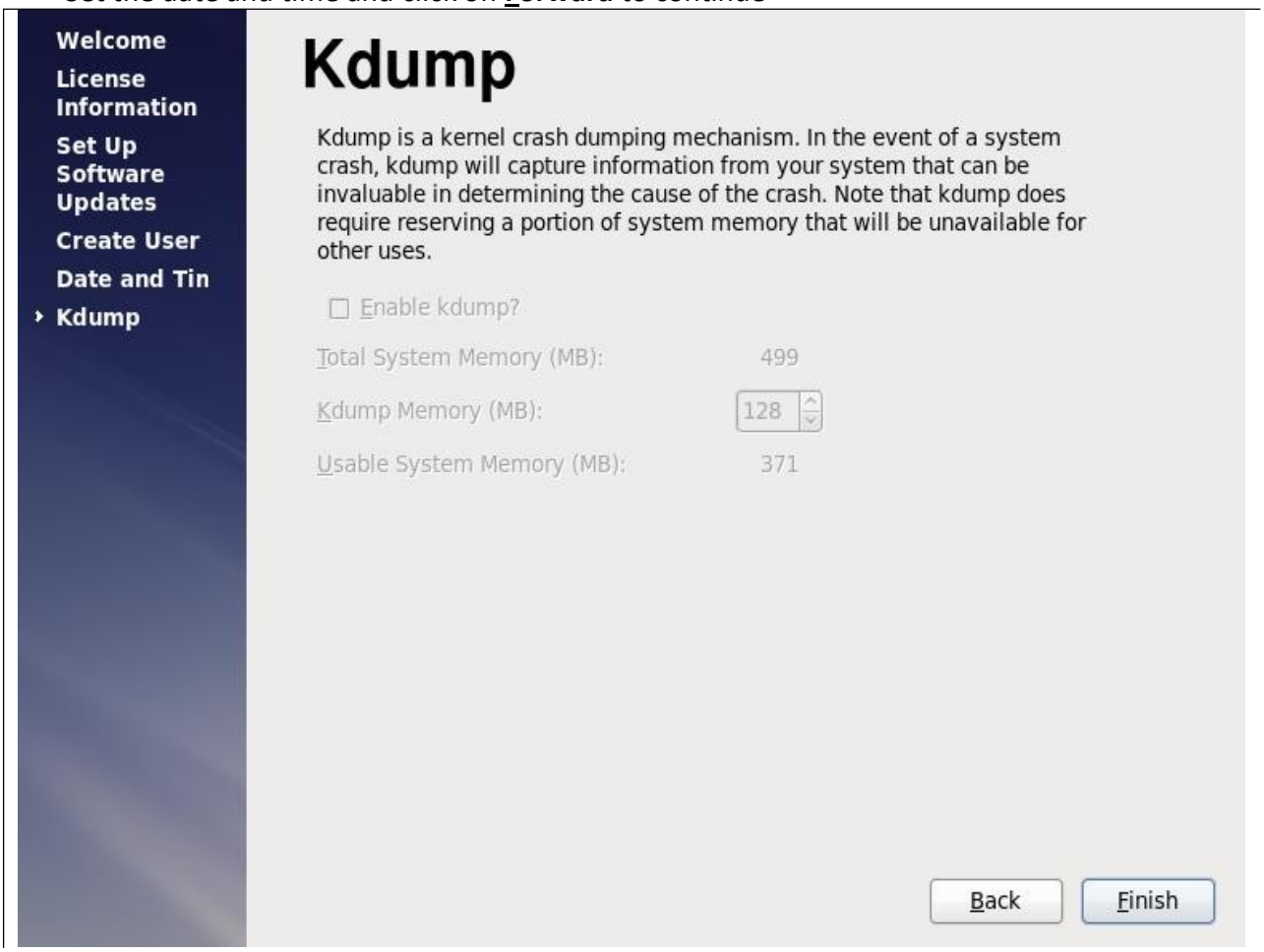
- Click on **Forward** to continue.



- Give a name to create a user and assign it a password. Click on **Forward** to continue.



- Set the date and time and click on **Forward** to continue



- Click on **Finish** and congratulation your installation is now completed.
- Login using either ktuser or root user.

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 ktfile)

  Hello World

  Ctrl+d (To save the file)

```
[root@ktlinux ~]# cat > ktfile
Hello World
```

To display the content of the file

cat filename (say ktfile)

```
[root@ktlinux ~]# cat ktfile
Hello World
[root@ktlinux ~]#
```

To append the data in the already existing file

 # cat >> <filename>

 # cat >> ktfile

Ctrl+d (to save the changes)

```
[root@ktlinux ~]# cat >> ktfile
Welcome to Kernel Technologies
[root@ktlinux ~]#
```

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

Creating a Directory:

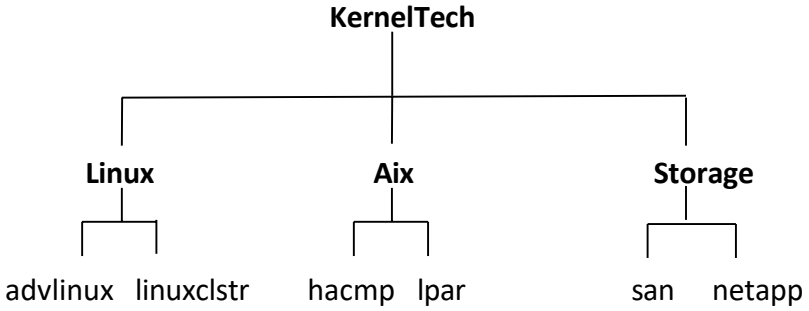
#mkdir <dir name>

#mkdir ktdir

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

Making multiple directories inside a directory

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



#mkdir –p KernelTech/{Linux/{advlinux,linuxclstr},Aix/{hacmp,lpar},Storage/{san,netapp}}

Check it by using **tree** command or **ls –R** command

```
[root@ktlinux ~]# mkdir -p KernelTech/{Linux/{advlinux,linuxclstr},Aix/{hacmp,lpar},Storage/{san,net
app}}
[root@ktlinux ~]# tree KernelTech/
KernelTech/
├── Aix
│   ├── hacmp
│   └── lpar
├── Linux
│   ├── advlinux
│   └── linuxclstr
└── Storage
    ├── netapp
    └── san

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

Copying files into directory

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

#cp file1 ktdir

```
[root@ktlinux ~]# cp file1 ktdir
[root@ktlinux ~]# cd ktdir
[root@ktlinux ktdir]# ls
file1
[root@ktlinux ktdir]#
```

Copying directories from one location to other

 # cp –rvfp <dir name> <destination name>

#cp –rvfp ktdir2 ktdir

```
[root@ktlinux ~]# cp -rvfp ktdir2 ktdir
`ktdir2' -> `ktdir/ktdir2'
`ktdir2/file2' -> `ktdir/ktdir2/file2'
`ktdir2/file3' -> `ktdir/ktdir2/file3'
`ktdir2/file4' -> `ktdir/ktdir2/file4'
`ktdir2/file1' -> `ktdir/ktdir2/file1'
`ktdir2/file5' -> `ktdir/ktdir2/file5'
[root@ktlinux ~]# cd ktdir
[root@ktlinux ktdir]# ls
file1  file2  ktdir2
[root@ktlinux ktdir]#
```

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

#mv <filename> <Destination directory>

#mv file2 ktdir

```
[root@ktlinux ~]# mv file2 ktdir
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  file1  install.log
Desktop          Downloads  file3  install.log.syslog
[root@ktlinux ~]# cd ktdir
[root@ktlinux ktdir]# ls
file1  file2
[root@ktlinux ktdir]#
```

Moving a Directory from one location to other

#mv <dir name> <destination dir name>

#mv ktdir ktdir2

```
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  file1  install.log          ktdir
Desktop          Downloads  file3  install.log.syslog   ktdir2
[root@ktlinux ~]# mv ktdir ktdir2
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  file1  install.log          ktdir2
Desktop          Downloads  file3  install.log.syslog   ktfile
[root@ktlinux ~]# cd ktdir2
[root@ktlinux ktdir2]# ls
file1  file2  file3  file4  file5  ktdir
[root@ktlinux ktdir2]#
```

Renaming a File

#mv <old name> <new name>

#mv ktfile kernelfile

```
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         ktfile  Pictures  Templates
Desktop          Downloads  install.log.syslog  Music   Public    Videos
[root@ktlinux ~]# cat ktfile
Welcome to Kernel Tech
[root@ktlinux ~]# mv ktfile kernelfile
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         kernelfile  Pictures  Templates
Desktop          Downloads  install.log.syslog  Music       Public    Videos
[root@ktlinux ~]# cat kernelfile
Welcome to Kernel Tech
[root@ktlinux ~]#
```

Renaming a Directory

- The procedure and command for renaming the directory is exactly same as renaming a file.
  #mv old name new name
  #mv ktdir kerneldir

```
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         kernelfile
Desktop          Downloads  install.log.syslog  ktdir
[root@ktlinux ~]# mv ktdir kerneldir
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         kerneldir
Desktop          Downloads  install.log.syslog  kernelfile
[root@ktlinux ~]#
```

Removing a File

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

```
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         kerneldir
Desktop          Downloads  install.log.syslog  kernelfile
[root@ktlinux ~]# rm kernelfile
rm: remove regular file `kernelfile'? y
```

Without prompting:

```
[root@ktlinux ~]# rm -f kernelfile
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         kerneldir
Desktop          Downloads  install.log.syslog  Music
[root@ktlinux ~]#
```

Removing an Empty directory

#rmdir dirname

```
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         kerneldir
Desktop          Downloads  install.log.syslog  ktdir
[root@ktlinux ~]# rmdir ktdir
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         kerneldir
Desktop          Downloads  install.log.syslog  Music
[root@ktlinux ~]#
```

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@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         kerneldir
Desktop          Downloads  install.log.syslog  Music
[root@ktlinux ~]# rmdir kerneldir/
rmdir: failed to remove `kerneldir/': Directory not empty
[root@ktlinux ~]# rm -rf kerneldir/
[root@ktlinux ~]# ls
anaconda-ks.cfg  Documents  install.log         Music
Desktop          Downloads  install.log.syslog  Pictures
```

VIM EDITOR

VI Visual display editor

VIM Visual display editor improved

This is command mode editor for files. Other editors in Linux are emacs, gedit
vi editor is most popular

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 page**
G	**To go to end of the page**
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 -ld l*	Directory listing only
#ls –ld 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@ktlinux ~]# ln -s ktfile ktfile.slink
[root@ktlinux ~]# ls -li ktfile ktfile.slink
5934 -rwxrwxrwx. 2 root root 0 Sep 17 09:21 ktfile
8394 lrwxrwxrwx. 1 root root 6 Sep 19 07:21 ktfile.slink -> ktfile
[root@ktlinux ~]#
```

Creating a Hard link:

#ln <source file> <Destination>

```
[root@ktlinux ~]# ln ktfile ktfile.hlink
[root@ktlinux ~]# ls -li ktfile ktfile.hlink
5934 -rwxrwxrwx. 2 root root 0 Sep 17 09:21 ktfile
5934 -rwxrwxrwx. 2 root root 0 Sep 17 09:21 ktfile.hlink
[root@ktlinux ~]#
```

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

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

#grep –i kernel ktfile (lets grep the word **kernel** whether upper of lower case in the file **ktfile**)

```
[root@ktlinux ~]# grep -i kernel ktfile
Welcome to Kernel Tech
Welcome to kernel Tech
Welcome to KERNEL TECH
[root@ktlinux ~]#
```

To display a word and 2 lines after the word:

#grep –nA2 wheel /etc/group

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

To display a word and 2 lines after the word:

#grep -nB2 wheel /etc/group

```
[root@ktlinux ~]# 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 kernel ktfile

```
[root@ktlinux ~]# cat ktfile
Welcome to Kernel Tech
Linux is Freedom
[root@ktlinux ~]# grep -v Kernel ktfile
Linux is Freedom
[root@ktlinux ~]#
```

To display the searched word in color
#grep --color root /etc/passwd

Combining grep with other commands
 # cat ktfile | grep –I kernel (pipe | is used to combine to commands)
#ls –l |grep –I ktfile
 # ifconfi g |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:/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@ktlinux ~]# 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@ktlinux ~]# 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@ktlinux ~]# 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
ktuser:x:501:501::/home/ktuser:/bin/bash
```

To display the custom lines

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

```
[root@ktlinux ~]# tail -5 /etc/passwd
ktuser:x:500:500:ktuser:/home/ktuser:/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
[root@ktlinux ~]#
```

Sort:

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

#sort filename

```
[root@ktlinux ~]# cat ktfile
Welcome to Kernel Tech
Welcome to Kernel Tech
Welcome to Kernel Tech
Linux is Freedom
Linux is Freedom
Linux is Freedom
[root@ktlinux ~]# sort ktfile
Linux is Freedom
Linux is Freedom
Linux is Freedom
Welcome to Kernel Tech
Welcome to Kernel Tech
Welcome to Kernel Tech
[root@ktlinux ~]#
```

To sort the file according to numbers
#sort –d ktfile or #sort –h ktfile

```
[root@ktlinux ~]# cat ktfile
4.Welcome to Kernel Tech
2.Welcome to Kernel Tech
3.Welcome to Kernel Tech
1.Linux is Freedom
6.Linux is Freedom
5.Linux is Freedom
[root@ktlinux ~]# sort -h ktfile
1.Linux is Freedom
2.Welcome to Kernel Tech
3.Welcome to Kernel Tech
4.Welcome to Kernel Tech
5.Linux is Freedom
6.Linux is Freedom
```

To remove the duplicate entries from the output
#sort –u ktfile

```
[root@ktlinux ~]# cat ktfile
Welcome to Kernel Tech
Welcome to Kernel Tech
Welcome to Kernel Tech
Linux is Freedom
Linux is Freedom
Linux is Freedom
[root@ktlinux ~]# sort -u ktfile
Linux is Freedom
Welcome to Kernel Tech
[root@ktlinux ~]#
```

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@ktlinux ~]# 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@ktlinux ~]# cat hello
hello,how,are,you
[root@ktlinux ~]# 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@ktlinux ~]# cat ktfile
Welcome to Kernel Tech
[root@ktlinux ~]# sed 's/Tech/Technologies/g' ktfile
Welcome to Kernel Technologies
[root@ktlinux ~]# cat ktfile
Welcome to Kernel Tech
```

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@ktlinux ~]# cat ktfile
Welcome to Kernel Tech
[root@ktlinux ~]# sed 's/Tech/Technologies/g' ktfile > ktf1
[root@ktlinux ~]# cat ktf1
Welcome to Kernel Technologies
```

Appending another output in same the same file

```
[root@ktlinux ~]# cat ktfile2
Ameerpet - Hyderabad
[root@ktlinux ~]# cat ktfile2 >> ktf1
[root@ktlinux ~]# cat ktf1
Welcome to Kernel Technologies
Ameerpet - Hyderabad
```

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 ktfile |tee ktf1

```
[root@ktlinux ~]# cat ktfile |tee ktf1
Welcome to Kernel Tech
[root@ktlinux ~]# cat ktf1
Welcome to Kernel Tech
```

Appending data in the same file using tee command
Syn: cat filename |tee –a filename2
#cat ktfile1 | tee –a ktf1

```
[root@ktlinux ~]# cat ktfile |tee ktf1
Welcome to Kernel Tech
[root@ktlinux ~]# cat ktfile2 |tee -a  ktf1
Ameerpet - Hyderabad
[root@ktlinux ~]# cat ktf1
Welcome to Kernel Tech
Ameerpet - Hyderabad
```

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
-name	For searching a file with its name
-inum	For searching a file with particular inode number
-type	For searching a particular type of file
-user	For files whose owner is a particular user
-group	For files belonging to particular group

Finding a File with name
#find / -name Kernel Tech

```
[root@ktlinux ~]# find / -name KernelTech
find: File system loop detected; `/var/named/
as `/var/named'.
/root/KernelTech
```

Finding a file with its inode number
#find / -inum 5934

```
[root@ktlinux ~]# find / -inum 5934
/sys/devices/virtual/block/loop3/dev
find: File system loop detected; `/var/
as `/var/named'.
find: `/proc/9206/task/9206/fd/5': No s
find: `/proc/9206/task/9206/fdinfo/5':
find: `/proc/9206/fd/5': No such file o
find: `/proc/9206/fdinfo/5': No such fi
/root/ktfile.hlink
/root/ktfile
[root@ktlinux ~]#
```

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

```
[root@ktlinux ~]# find / -user ktuser
find: File system loop detected; `/var
as `/var/named'.
/var/spool/mail/ktuser
/home/ktuser
/home/ktuser/.mozilla
/home/ktuser/.mozilla/plugins
/home/ktuser/.mozilla/extensions
/home/ktuser/kernel2
/home/ktuser/.bashrc
/home/ktuser/.gnome2
/home/ktuser/kernel1
```

Finding the files whose group is "ktgroup"
#find / -group ktgroup

```
[root@ktlinux ~]# find / -group  ktgroup
find: File system loop detected; `/var/named/chroot
as `/var/named'.
/home/ktuser/kernel2
/home/ktuser/kernel1
/home/ktuser/kernel4
/home/ktuser/kernel5
/home/ktuser/kernel3
```

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' the contents of dir
w	Write, edit, append, delete file	Add/Del/Rename contents of dir
x	To run a command/shell script	To enter into dir using 'cd'

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 ktfile (where ktfile is the name of the file)

```
[root@ktlinux ~]# chmod u=rwx,g=rw,o=r ktfile
[root@ktlinux ~]# ls -l ktfile
-rwxrw-r--. 1 root root 0 Sep 17 09:21 ktfile
[root@ktlinux ~]#
```

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

#chmod ugo=rwx <file name>

```
[root@ktlinux ~]# ls -l ktfile
-rwxrw-r--. 1 root root 0 Sep 17 09:21 ktfile
[root@ktlinux ~]# chmod ugo=rwx ktfile
[root@ktlinux ~]# ls -l ktfile
-rwxrwxrwx. 1 root root 0 Sep 17 09:21 ktfile
```

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

- **#chmod u+x ktfile (Adding execute permission to user only)**
- **#chmod go-wx ktfile (Removing write and execute permissions from group and other)**
- **#chmod go+wx ktfile (Adding write and execute permissions from group and other)**
- **#chmod go=r ktfile (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 ktfile (where 7 means rwx i.e. 4+2+1, rw=6 i.e. 4+2 and 1 indicates x)

```
[root@ktlinux ~]# ls -l ktfile
-rwxrwxrwx. 1 root root 0 Sep 17 09:21 ktfile
[root@ktlinux ~]# chmod 764 ktfile
[root@ktlinux ~]# ls -l ktfile
-rwxrw-r--. 1 root root 0 Sep 17 09:21 ktfile
```

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

#chmod 777 ktfile

```
[root@ktlinux ~]# ls -l ktfile
-rwxrw-r--. 1 root root 0 Sep 17 09:21 ktfile
[root@ktlinux ~]# chmod 777 ktfile
[root@ktlinux ~]# ls -l ktfile
-rwxrwxrwx. 1 root root 0 Sep 17 09:21 ktfile
```

Likewise you can give different permissions according to your requirement

Removing all permissions from others

#chmod 770 ktfile (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@ktlinux ~]# umask
0022
[root@ktlinux ~]#
```

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@ktlinux ~]# umask
0022
[root@ktlinux ~]# touch ktfile2
[root@ktlinux ~]# ls -l ktfile2
-rw-r--r--. 1 root root 0 Sep 19 04:19 ktfile2
[root@ktlinux ~]#
```

- **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@ktlinux ~]# umask
0022
[root@ktlinux ~]# mkdir ktdir2
[root@ktlinux ~]# ls -ld ktdir2
drwxr-xr-x. 2 root root 4096 Sep 19 04:24 ktdir2
[root@ktlinux ~]#
```

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

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

MANAGING PARTITIONS & FILE SYSTEMS

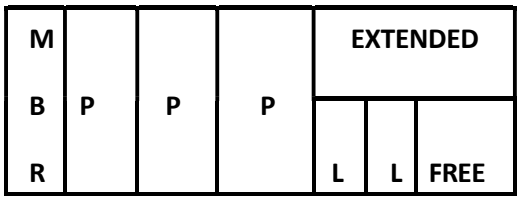
What is a partition?

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

The Structure of Disk Partition

- On the disk where O/S is installed, will have the first partition as **MBR.**
- **MBR** is a Master Boot Record, which contains two important utilities, **IPL** (Initial Program Loader) and **PTI** (Partition Table information)
- **IPL is** responsible for booting the operating the 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.**
- 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:

- Every disk can have only **3** Primary Partitions.
- **Primary Partition** is a partition which usually holds the **operating system**. Only **one** amongst the 3 primary partitions can be active which will be booted by **MBR** to load 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.
- **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.

Disk Identification:

Different type of disks will be having different initials in Linux

- **IDE** drive will be shown as **/dev/hda**
- **SCSI** drive will be shown as **/dev/sda**
- **Virtual** drive will be shown as **/dev/vda**

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 used in RHEL 6:

- The file systems supported in Linux are **ext2, ext3 and in RHEL 6 ext4, vfat, etc.**
- **Ext** 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
1.	Stands for Second Extended File System	Stands for Third Extended File System	Stands for Fouth Extended File System
2.	It was introduced in 1993	It was introduced in 2001	It was introduced in 2008.
3.	Does not have journaling feature.	Supports Journaling Feature.	Supports Journaling Feature.
4.	Maximum File size can be from **16 GB to 2 TB**	Maximum File Size can be from **16 GB to 2 TB**	Maximum File Size can be from **16 GB to 16 TB**
5.	Maximum ext2 file system size can be from **2 TB to 32 TB**	Maximum ext3 file system size can be from **2 TB to 32 TB**	Maximum ext4 file system size is **1 EB (Exabyte). 1 EB = 1024 PB (Petabyte). 1 PB = 1024 TB (Terabyte).**
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).

MOUNTING:-

- 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 a additional file system is mounted.
- The /mnt directory exists by default on all Unix-like systems. It, 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 is 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

#fdisk –l or parted –l

```
[root@ktcl5 Desktop]# fdisk -l

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

[root@ktcl5 Desktop]# parted -l
Model: VMware Virtual disk (scsi)
Disk /dev/sda: 32.2GB
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   8599MB  8389MB  primary   ext4
 3      8599MB  12.8GB  4194MB  primary   ext4
 4      12.8GB  32.2GB  19.4GB  extended
 5      12.8GB  15.9GB  3146MB  logical   ext4
 6      15.9GB  18.0GB  2097MB  logical   linux-swap(v1)
```

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

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.

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. **1-7**

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@ktcl5 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

#partprobe /dev/sda
Or
#partx –a /dev/sda
Or
#kpartx /dev/sda

Note: In RHEL6 **partprobe** is not functioning properly, so it is recommended to use **partx** command only.

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

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@ktcl5 Desktop]# mkfs.ext4 /dev/sda7
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
128520 inodes, 513008 blocks
25650 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
2040 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 38 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@ktcl5 Desktop]#
```

- Likewise you can format the different partitions with different file systems like
- #mkfs.ext3 /dev/sda8
- #mkfs.vfat /dev/sda9

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/sda7 /kernel

```
[root@ktcl5 ~]# mkdir /kernel
[root@ktcl5 ~]# mount /dev/sda7 /kernel
[root@ktcl5 ~]#
```

To View all the mounted partitions

#mount

```
[root@ktcl5 ~]# mount
/dev/sda3 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)
/dev/sda5 on /home type ext4 (rw)
/dev/sda2 on /usr 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)
/dev/sr0 on /media/RHEL_6.0 x86_64 Disc 1 type iso9660 (ro,nosuid,nodev,uhelper
mode=0500)
/dev/sda7 on /kernel type ext4 (rw)
[root@ktcl5 ~]#
```

- 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 /kernel
- Add the data and exit the directory

Unmounting a partition

#umount <mount point directory>

#umount /kernel

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 Fri Nov  5 08:05:42 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
#
UUID=02419cbf-4435-4798-b879-8f821e15cd5b /                       ext4    defaults        1 1
UUID=46aa05b1-9d60-432a-9d64-69a0332f0b2d /boot                   ext4    defaults        1 2
UUID=4f72eaef-8667-450c-ab56-ed30d5dfb8b6 /home                   ext4    defaults        1 2
UUID=940a9c29-3a8f-4b6b-a84b-aaa7be57e2a8 /usr                    ext4    defaults        1 2
UUID=204d9293-fe4e-4dc2-89cd-816c7a906188 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/sda7               /kernel                 ext4    defaults        0 0
~
```

Device Name | Mount Point | Type of FS | Mount options | Dumping | Check Sequence

#mount –a

```
[root@ktcl5 ~]# mount -a
[root@ktcl5 ~]# mount
/dev/sda3 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)
/dev/sda5 on /home type ext4 (rw)
/dev/sda2 on /usr 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)
/dev/sr0 on /media/RHEL_6.0 x86_64 Disc 1 type iso9660 (ro,nosuid,nodev,uhelper
mode=0500)
/dev/sda7 on /kernel type ext4 (rw)
[root@ktcl5 ~]#
```

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

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 /dev/sda7**

```
[root@ktcl5 ~]# fuser -cu /dev/sda7
/dev/sda7:            5821c(ktuser)
[root@ktcl5 ~]#
```

- Check for the files which are open with **lsof /dev/sda7**
- Kill the open connections using **fuser –ck /kernel/hello** where hello is the file which is open.
- 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@ktcl5 ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda3             3.9G  383M  3.3G  11% /
tmpfs                 499M  372K  499M   1% /dev/shm
/dev/sda1             194M   39M  146M  21% /boot
/dev/sda5             2.9G   69M  2.7G   3% /home
/dev/sda2             7.7G  3.1G  4.3G  42% /usr
/dev/sr0              3.2G  3.2G     0 100% /media/RHEL_6.0 x86_64 Disc 1
/dev/sda7             486M   11M  450M   3% /kernel
[root@ktcl5 ~]#
```

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.**

Assigning label to the partition:

Assigning the label is giving some name to the partition. To assign label to the partition **e2label** command is used

Syntax

#e2label <partition name> <label> **To check the label**

#e2label /dev/sda7 ktdisk **#e2label /dev/sda7**

```
[root@ktcl5 ~]# e2label /dev/sda7 ktdisk
[root@ktcl5 ~]# e2label /dev/sda7
ktdisk
```

To list all the mounted partition along with their labels, use **mount –l** command

Mounting a partition using its label:

Mounting a **/dev/sda7** partition with its label **ktdisk,** verify it with **mount** command

```
[root@ktcl5 ~]# mount LABEL=ktdisk /kernel
[root@ktcl5 ~]# mount
```

Making a permanent mount point using label

- As we know that to make a permanent mount point, an entry has to be made in **/etc/fstab** file.

#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
LABEL=ktdisk            /kernel                 ext4    defaults        0 0
```

- Now use **mount –a** command and verify it with **mount** command whether it is mounted or not.

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@ktcl5 ~]# blkid /dev/sda7
/dev/sda7: LABEL="ktdisk" UUID="f489d0b1-ffca-4e21-917a-7b82c0edd255" TYPE="ext4"
[root@ktcl5 ~]#
```

#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 /kernel                ext4    defaults        0 0
```

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

Creating a Swap Partition:

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**

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

- To see the swap usage use
  #swapon –s

- To format the partition with swap file system use
  #mkswap <parititon name>

- To activate the swap space use
  #swapon <partition name>

- To deactivate the swap space use
  #swapoff <partition name>

Creating a Swap partion

- 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 **l**)
- Update the partition table using **partx –a** command

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

Turn on the newly created swap space and verify it.

- To turn on the swap space the syntax is
  #swapon /dev/sda6

```
[root@ktlinux /]# swapon /dev/sda6
[root@ktlinux /]# swapon -s
Filename                                Type            Size    Used    Priority
/dev/sda3                               partition       4095992 0       -1
/dev/sda6                               partition       522072  0       -2
[root@ktlinux /]# 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 mount 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_ktlinux-rootlv /                       ext4    defaults        1 1
UUID=ce33cb92-21b8-49c0-95fc-17f40437d44b /boot                   ext4    defaults        1 2
/dev/mapper/vg_ktlinux-homelv /home                   ext4    defaults        1 2
/dev/mapper/vg_ktlinux-usrlv /usr                     ext4    defaults        1 2
/dev/mapper/vg_ktlinux-varlv /var                     ext4    defaults        1 2
UUID=60dcea45-f68b-473d-b953-0fbc5b63d5fc 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/ktpart      /kernel                 ext4    defaults        0 0
/dev/sda6               swap                    swap    defaults        0 0
```

Removing the SWAP Partition

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

Encrypting a Partition using LUKS (Linux Unified Key Setup):

- LUKS is a standard format for device encryption.
- LUKS ensures the data protection inside the partition, especially against the data breach.
- It encrypts the partition or volume, which will decrypt only by providing correct password.
- The partition must be decrypted before the file system in it can be mounted.
- Once it is open (decrypted), you can work with the partition normally i.e. mounting and adding the data to the partition.
- After the completion of work the partition has to be closed i.e. encrypted, so that it cannot be mounted nor can be accessible by others, unless you lose password.

Commands used in LUKS encryption:

- **cryptsetup luksFormat**: To Format the partition with encryption, and assigning the password.
- **cryptsetup luksOpen:** To open or decrypt the partition. (password will be required) and the you need to assign some **name** to it, which will be used for further operation as **/dev/mapper/name.**
- **crypsetup luksClose:** To Close or encrypt back the partition after use.
- **cryptsetup luksAddKey:** To add the key (password) to the configuration to automatically decrypting the partition.

Steps to Encrypt the Partition:

1. Create a normal partition using fdisk.
2. Format the partition using **luks** and assign the passphrase.
3. Decrypt the partition.
4. Now format again using normal ext4 formatting.
5. Mount the partition, Make a permanent mount.
6. Access the partition and add the data
7. Unmount the partition, and close the partition i.e. encrypt back.

1. **Create a normal partition using fdisk.**

#fdisk /dev/sda and create a partition of size 500MB

```
   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
[root@ktlinux ~]#
```

2. **Format the partition using luks and assign the passphrase.**
   - To encrypting a partition using luks the command is
   - **#cryptsetup luksFormat /dev/sda5**
   - It will prompt us to continue, type uppercase **YES** to continue
   - Then it will ask you to assign a passphrase and verify it, which will be used later to decrypt the partition.

```
[root@ktlinux ~]# cryptsetup luksFormat /dev/sda5

WARNING!
========
This will overwrite data on /dev/sda5 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase:
Verify passphrase:
[root@ktlinux ~]#
```

3. **Decrypt the partition.**
   - To decrypt the partition for further use, make use of the following steps
   - **#cryptsetup luksOpen /dev/sda5 ktpart**
   - Where **ktpart** is the name given to the partition, it is mandatory to give a name to the partition. You can assign any name.
   - It will ask passphrase; enter the passphrase to decrypt it. (it should be the same as assigned in step 2)

   Note: From now onward the disk will be represented as **/dev/mapper/ktpart**
   Note: Don't use **/dev/sda5** to format the partition.

```
[root@ktlinux ~]# cryptsetup luksOpen /dev/sda5 ktpart
Enter passphrase for /dev/sda5:
[root@ktlinux ~]#
```

4. **Formatting the partition with ext4 file system.**

```
[root@ktlinux ~]# mkfs.ext4 /dev/mapper/ktpart
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
130048 inodes, 519908 blocks
25995 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
```

5. **Making the permanent mount point and mount the partition.**
   - For temporary mounting make a directory and use
   - **#mount /dev/mapper/ktpart /kernel**
   - Make the entry in **/etc/fstab** to make it permanent as shown below
   - Also make an entry in **/etc/crypttab** as shown below.

#vim /etc/fstab

```
UUID=60dcea45-f68b-473d-b953-0fbc5b63d5fc swap                    swap    defaults
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/ktpart      /kernel                 ext4    defaults        0 0
```

#vim /etc/crypttab

```
ktpart  /dev/sda5
```

6. **Access the partition and some data to it.**
   - Access the partition using **mount point**

```
[root@ktlinux ~]# cd /kernel
[root@ktlinux kernel]# ls
lost+found
[root@ktlinux kernel]# touch ktfile{1..5}
[root@ktlinux kernel]# ls
ktfile1  ktfile2  ktfile3  ktfile4  ktfile5  lost+found
[root@ktlinux kernel]# cd ..
[root@ktlinux /]#
```

7. **Unmount the partition, and close the partition i.e. encrypt it back**
   - **# umount /dev/mapper/ktpart**
   - **#cryptsetup luksClose /dev/mapper/ktpart**
   - **#mount –a** (To check encryption is working)

```
[root@ktlinux /]# umount /dev/mapper/ktpart
[root@ktlinux /]# cryptsetup luksClose /dev/mapper/ktpart
[root@ktlinux /]# mount -a
mount: special device /dev/mapper/ktpart does not exist
[root@ktlinux /]#
```

Saving the passphrase in file, to auto mount the partition.

- When you assign the label of the partition in the **/etc/crypttab,** the system will be halted at the time of boot and will ask you to enter the passphrase of that particular partition so that the partition can be decrypt and mounted.
- Either you should type the passphrase to continue or can ignore it by using **ctrl+c** to continue booting without decrypting and mounting the partition.
- In order to make the O/S to take the passphrase automatically and unlock the partition, we can save the passphrase in a file, so that it can take the passphrase and mount it and boot it normally without halting.

Steps to save the passphrase and adding it in LUKS configuration

- **Make a file and store the passphrase in it.**

#vim enphrs

```
kernel123
~
~
```

- **Change the permission of the file (600), and add the path of the file in /etc/crypttab**

Note: The permission of the file is changed so that groups and others may not be able to view and modify the contents of the file.

```
[root@ktlinux /]# chmod 600 enphrs
[root@ktlinux /]# ls -l enphrs
-rw-------. 1 root root 10 Sep 28 06:44 enphrs
[root@ktlinux /]# vim /etc/crypttab
ktpart  /dev/sda5  /enphrs
```

- **Add the key in LUKS configuration**

 # cryptsetup luksAddKey /dev/sda5 /enphrs and enter the **passphrase** of the partition

```
[root@ktlinux /]# cryptsetup luksAddKey /dev/sda5 /enphrs
Enter any passphrase:
[root@ktlinux /]#
```

After making the above changes, restart the system and check whether it is halting to ask you passphrase or booting continuously. It will not ask any passphrase for sure.

Removing the encryption

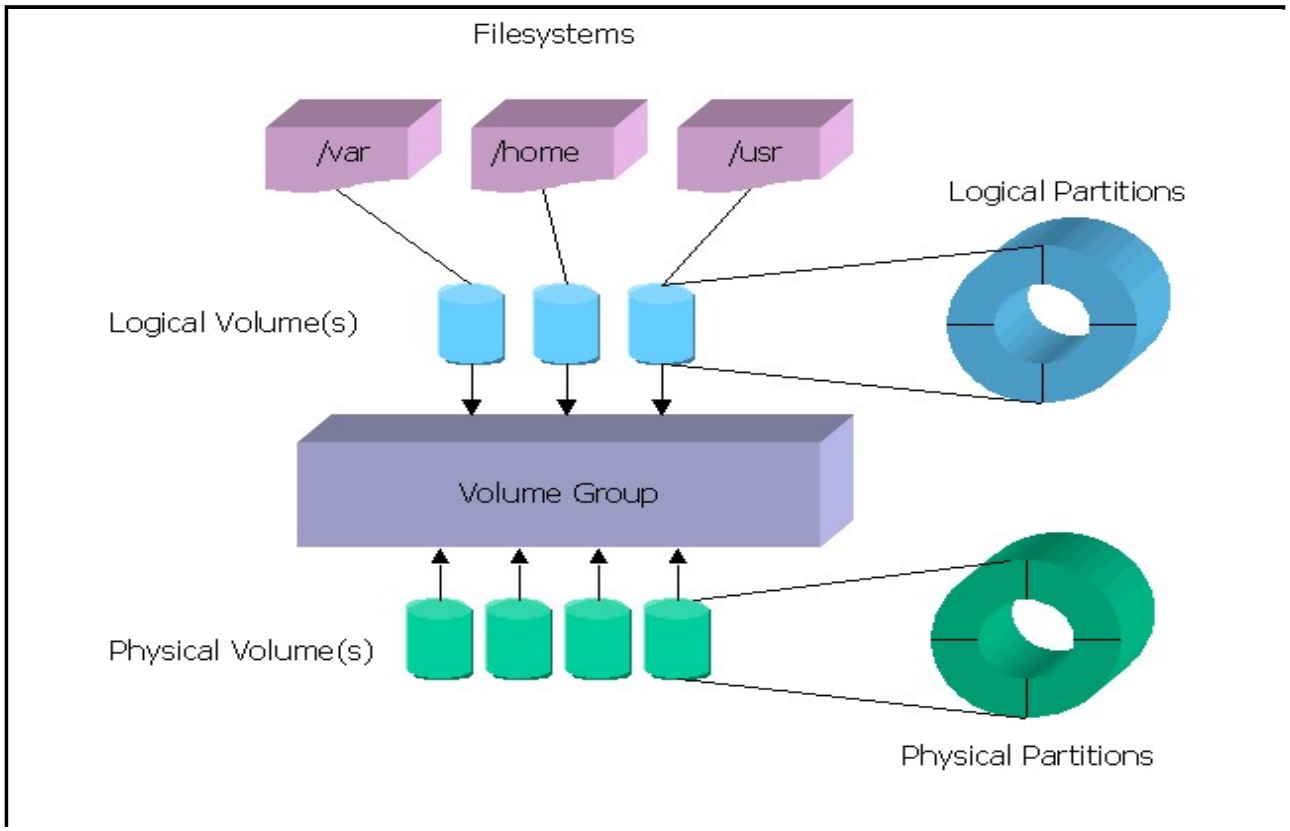
- Close the encryption as shown in **step 7** above and
- Format the partition normally in **ext4** file system.
- **#mkfs.ext4 /dev/sda5**
- Encryption will be removed.

Note: All the data in the partition will be lost, so make sure that you have backup of it before formatting.

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) 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
pvs	**Displays all the physical volumes**
vgs	**Displays all volume groups in the system**
lvs	**Displays all the logical volumes in the system**
pvdisplay	**Displays detailed information on physical volumes**
vgdisplay	**Displays detailed information on volume groups**
lvdisplay	**Displays detailed information on logical volumes**
pvcreate	**Create a new physical volume**
vgcreate	**Create a new volume group.**
lvcreate	**Creates a new logical volume**
vgextend	**Add a new physical disk to a volume group.**
lvextend	**Extends a logical volume**
lvresize	**Resizes a logical volume**
lvreduce	**Reduces a logical volume**
pvmove	**Moves/migrates data from one physical volume to another**
vgreduce	**Reduces a volume group by removing a PV from it.**
pvremove	**Deletes a physical volume**
vgremove	**Removes /Deletes a volume group**
lvremove	**Removes /Deletes a logical volume**

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@ktlinux Desktop]# pvcreate /dev/sda7
  Physical volume "/dev/sda7" successfully created
```

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

```
[root@ktlinux 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> <partition name>
  #vgcreate ktvg /dev/sda7

```
[root@ktlinux Desktop]# vgcreate ktvg /dev/sda7
  Volume group "ktvg" successfully created
```

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

```
[root@ktlinux Desktop]# vgs
  VG         #PV #LV #SN Attr   VSize   VFree
  ktvg         1   0   0 wz--n- 508.00m 508.00m
  vg_ktlinux   1   4   0 wz--n-  29.29g   1.95g
```

```
[root@ktlinux Desktop]# vgdisplay ktvg
  --- Volume group ---
  VG Name               ktvg
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  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               731Qe2-fDm0-PuZz-ZrjX-eki6-lEru-vAi36c
```

- 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 ktlv ktvg** (To create a LV of 200MB)

```
[root@ktlinux Desktop]# lvcreate -L 300M -n ktlv ktvg
  Logical volume "ktlv" created
[root@ktlinux Desktop]#
```

- 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 ktvg**

```
[root@ktlinux Desktop]# lvs
  LV     VG         Attr   LSize   Origin Snap%  Move Log Copy%  Conver
  ktlv   ktvg       -wi-a- 300.00m
  homelv vg_ktlinux -wi-ao   3.91g
  rootlv vg_ktlinux -wi-ao   3.91g
  usrlv  vg_ktlinux -wi-ao   9.77g
  varlv  vg_ktlinux -wi-ao   9.77g
```

```
[root@ktlinux Desktop]# lvdisplay ktvg
  --- Logical volume ---
  LV Name                /dev/ktvg/ktlv
  VG Name                ktvg
  LV UUID                x0NkZc-Qq11-6EwK-HuGj-YN1I-5vWV-XQ1JKB
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                300.00 MiB
  Current LE             75
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:4
```

- **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 ext3 or vfat.
- 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/ktvg/ktlv**

```
[root@ktlinux Desktop]# mkfs.ext4 /dev/ktvg/ktlv
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
76912 inodes, 307200 blocks
15360 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
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

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 32 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@ktlinux Desktop]#
```

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/ktvg/ktlv /ktdir**
- Verify the mounting with **mount** command
- Make it a permanent mount by making an entry in **/etc/fstab**

```
[root@ktlinux /]# mount /dev/ktvg/ktlv /ktdir
```

#vim /etc/fstab

```
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
/dev/mapper/ktpart      /kernel                 ext4    defaults        0 0
/dev/sda6               swap                    swap    defaults        0 0
/dev/ktvg/ktlv          /ktdir                  ext4    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 **partx –a** 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> <partition name>**
- **#vgextend ktvg /dev/sda8**
- Verify it **pvs** command

```
[root@ktlinux /]# pvcreate /dev/sda8
  Physical volume "/dev/sda8" successfully created
[root@ktlinux /]# vgextend ktvg /dev/sda8
  Volume group "ktvg" successfully extended
[root@ktlinux /]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sda1  vg_ktlinux lvm2 a-    29.29g   1.95g
  /dev/sda7  ktvg       lvm2 a-   508.00m 208.00m
  /dev/sda8  ktvg       lvm2 a-   508.00m 508.00m
[root@ktlinux /]#
```

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/ktvg/ktlv**
- Update the file system by using **resize2fs** command
- **#resize2fs /dev/vg/lv name**
- **#resize2fs /dev/ktvg/ktlv**
- Verify the change by using **df –h** command

```
[root@ktlinux /]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_ktlinux-rootlv
                      3.9G  450M  3.3G  13% /
tmpfs                1004M  300K 1004M   1% /dev/shm
/dev/sda2             194M   39M  146M  21% /boot
/dev/mapper/vg_ktlinux-homelv
                      3.9G   72M  3.6G   2% /home
/dev/mapper/vg_ktlinux-usrlv
                      9.7G  2.3G  6.9G  25% /usr
/dev/mapper/vg_ktlinux-varlv
                      9.7G  3.5G  5.7G  39% /var
/dev/mapper/ktvg-ktlv
                      291M   11M  266M   4% /ktdir
[root@ktlinux /]#
```

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

```
[root@ktlinux /]# lvextend -L +200M /dev/ktvg/ktlv
  Extending logical volume ktlv to 500.00 MiB
  Logical volume ktlv successfully resized
[root@ktlinux /]# resize2fs /dev/ktvg/ktlv
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/ktvg/ktlv is mounted on /ktdir; on-line resizing required
old desc_blocks = 2, new_desc_blocks = 2
Performing an on-line resize of /dev/ktvg/ktlv to 512000 (1k) blocks.
The filesystem on /dev/ktvg/ktlv is now 512000 blocks long.
```

- **Verify it by df –h**

```
[root@ktlinux /]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_ktlinux-rootlv
                      3.9G  450M  3.3G  13% /
tmpfs                1004M  300K 1004M   1% /dev/shm
/dev/sda2             194M   39M  146M  21% /boot
/dev/mapper/vg_ktlinux-homelv
                      3.9G   72M  3.6G   2% /home
/dev/mapper/vg_ktlinux-usrlv
                      9.7G  2.3G  6.9G  25% /usr
/dev/mapper/vg_ktlinux-varlv
                      9.7G  3.5G  5.7G  39% /var
/dev/mapper/ktvg-ktlv
                      485M   11M  450M   3% /ktdir
[root@ktlinux /]#
```

<u>Reducing the size of an LV</u>

- 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.
    - Organized the data before reducing the size of LV.
    - 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
- Organize the data in **LV** by using **e2fsck** command
- **#e2fsck -f /dev/ktvg/ktlv.**
- Update the file system by using **resize2fs** command
- **#resize2fs /dev/ktvg/ktlv 300M** (where **300M** is the approximate total size of **LV** after reduction)

- Now reduce the size by using # **lvreduce -L -200M /dev/ktvg/ktlv** 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.**
- **Umount the LV by using umount command**

```
[root@ktlinux /]# umount /ktdir
[root@ktlinux /]#
```

- **Organize the data in the file system.**

```
[root@ktlinux /]# e2fsck -f /dev/ktvg/ktlv
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
ktlv: 11/127512 files (0.0% non-contiguous), 26603/512000 blocks
[root@ktlinux /]#
```

- Update the file system about the size after reduction

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

[root@ktlinux /]#
```

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

```
[root@ktlinux /]# lvreduce -L -200M /dev/ktvg/ktlv
  WARNING: Reducing active and open logical volume to 300.00 MiB
  THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce ktlv? [y/n]: y
  Reducing logical volume ktlv to 300.00 MiB
  Logical volume ktlv successfully resized
[root@ktlinux /]#
```

- **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)
- **#df –h**

```
/dev/mapper/ktvg-ktlv
                      291M   11M  266M   4% /ktdir
[root@ktlinux /]#
```

Moving or Migrating the LV (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.**
    - Add new **PV,** which should be of the same size or higher than that of the replacing **PV** to the volume group.
    - Migrate the **PV**s contents to the new **PV** using following command
    - **#pvmove <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@ktlinux ~]# cd /ktdir/
[root@ktlinux ktdir]# ls
ktdata1   ktdata2  ktdata4  ktdata6  ktdata8  lost+found
ktdata10  ktdata3  ktdata5  ktdata7  ktdata9
[root@ktlinux ktdir]# cat ktdata1
Welcome to Kernel Technologies
[root@ktlinux ktdir]#
```

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

```
[root@ktlinux ~]# pvs
  PV         VG     Fmt  Attr PSize   PFree
  /dev/sda1  rootvg lvm2 a-    17.57g      0
  /dev/sda6  ktvg   lvm2 a-   508.00m 208.00m
  /dev/sda7         lvm2 a-   509.84m 509.84m
[root@ktlinux ~]# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sda6
  VG Name               ktvg
  PV Size               509.84 MiB / not usable 1.84 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              127
  Free PE               52
  Allocated PE          75
  PV UUID               11jki4-doLN-I0DB-P8T9-itBr-Hn6V-yVrrz2
```

- **Unmount the file system on that PV.**
- **#umount /ktdir**

- **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@ktlinux ~]# pvs
  PV         VG     Fmt  Attr PSize   PFree
  /dev/sda1  rootvg lvm2 a-    17.57g      0
  /dev/sda6  ktvg   lvm2 a-   508.00m 108.00m
  /dev/sda7         lvm2 a-   509.84m 509.84m
[root@ktlinux ~]# vgextend ktvg /dev/sda7
  Volume group "ktvg" successfully extended
[root@ktlinux ~]#
```

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

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

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

```
[root@ktlinux ~]# mount -a
[root@ktlinux ~]# cd /ktdir
[root@ktlinux ktdir]# ls
ktdata1   ktdata2  ktdata4  ktdata6  ktdata8  lost+found
ktdata10  ktdata3  ktdata5  ktdata7  ktdata9
[root@ktlinux ktdir]# cat ktdata1
Welcome to Kernel Technologies
[root@ktlinux ktdir]#
```

- **Remove the faulty 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 ktvg /dev/sda6**

```
[root@ktlinux ktdir]# vgreduce ktvg /dev/sda6
  Removed "/dev/sda6" from volume group "ktvg"
```

Deleting/Removing an 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 ktlv** (it will prompt to you to continue, press **y** to continue)
- Verify it by using **lvdisplay command**

```
[root@ktlinux ~]# umount /ktdir
[root@ktlinux ~]# vim /etc/fstab
[root@ktlinux ~]# lvremove /dev/ktvg/ktlv
Do you really want to remove active logical volume ktlv? [y/n]: y
  Logical volume "ktlv" successfully removed
[root@ktlinux ~]# lvdisplay ktvg
[root@ktlinux ~]#
```

- 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 a 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 ktvg**

```
[root@ktlinux ~]# vgremove ktvg
  Volume group "ktvg" successfully removed
[root@ktlinux ~]#
```

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

- 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 demolish it, it is just a matter of a moment. Isn't it….!

Creating a VG by specifying the 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: 0x00003d37

   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+  83  Linux
[root@ktlinux ~]# 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 ktvg2 –s 16 /dev/sda5

```
[root@ktlinux ~]# vgcreate ktvg2 -s 16 /dev/sda5
  Volume group "ktvg2" successfully created
[root@ktlinux ~]#
```

- Verify the PE size using **vgdisplay** command

```
[root@ktlinux ~]# vgdisplay ktvg2
  --- Volume group ---
  VG Name               ktvg2
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  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               lacNDZ-szQ6-Fq76-lZMZ-Px3K-KQXo-llMDQV
```

Creating an LV of 400MB, 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**
- **In Command we are specifying the no. of LE not the size of LE, as the size of LE is based on 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

- **Check the size of PE using vgdisplay command**

```
[root@ktlinux ~]# vgdisplay ktvg2
  --- Volume group ---
  VG Name               ktvg2
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  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               lacNDZ-szQ6-Fq76-lZMZ-Px3K-KQXo-llMDQV
```

- **Okay, now then we know the size of PE is 16, lets calculate how many LE is required to create an LV of 400 MB.**
- **The formula for calculating no. of LE is**
  <size of LV required, in MB> divided by Size of PE
  400/16 = 25

- **If the size of LV is to be 2 GB then first we need to convert GB into MB and then calculate 2x1024 / 16 =128.**

- You can use **#bc** command to do all the calculations. Use **ctrl+d or Ctrl+c** to quit the calculator

```
[root@ktlinux ~]# bc
bc 1.06.95
400/16
25
2*1024/16
128
```

- **So now we got the calculation done and we came to know that 25 LEs are required to create 400MB of LV.**
- **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 ktlv2 ktvg2

```
[root@ktlinux ~]# lvcreate -l 25 -n ktlv2 ktvg2
  Logical volume "ktlv2" created
[root@ktlinux ~]#
```

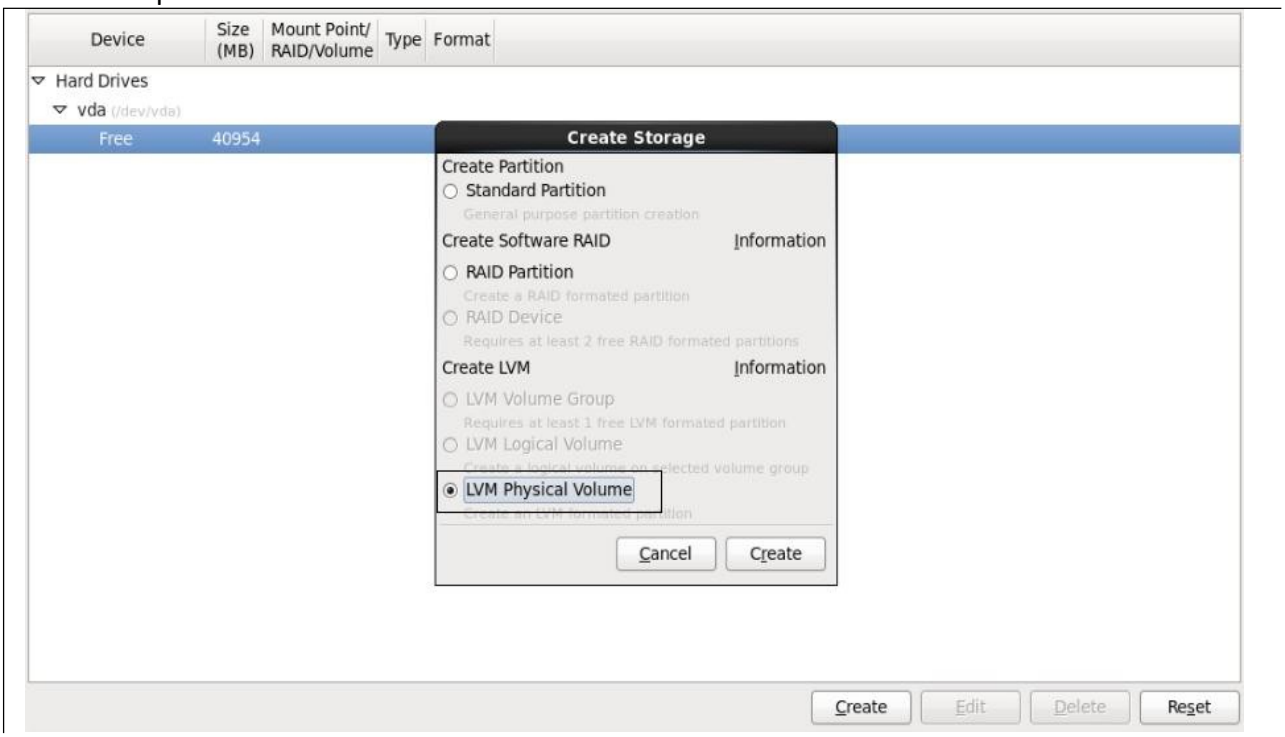
- **Now check the size of the LV "ktlv2" using lvdisplay command**
  #lvdisplay ktvg

```
[root@ktlinux ~]# lvdisplay ktvg2
  --- Logical volume ---
  LV Name                /dev/ktvg2/ktlv2
  VG Name                ktvg2
  LV UUID                t83rmU-kk2z-I83a-BXpo-18LH-aDM8-F0F1iW
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                400.00 MiB
  Current LE             25
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:5
```

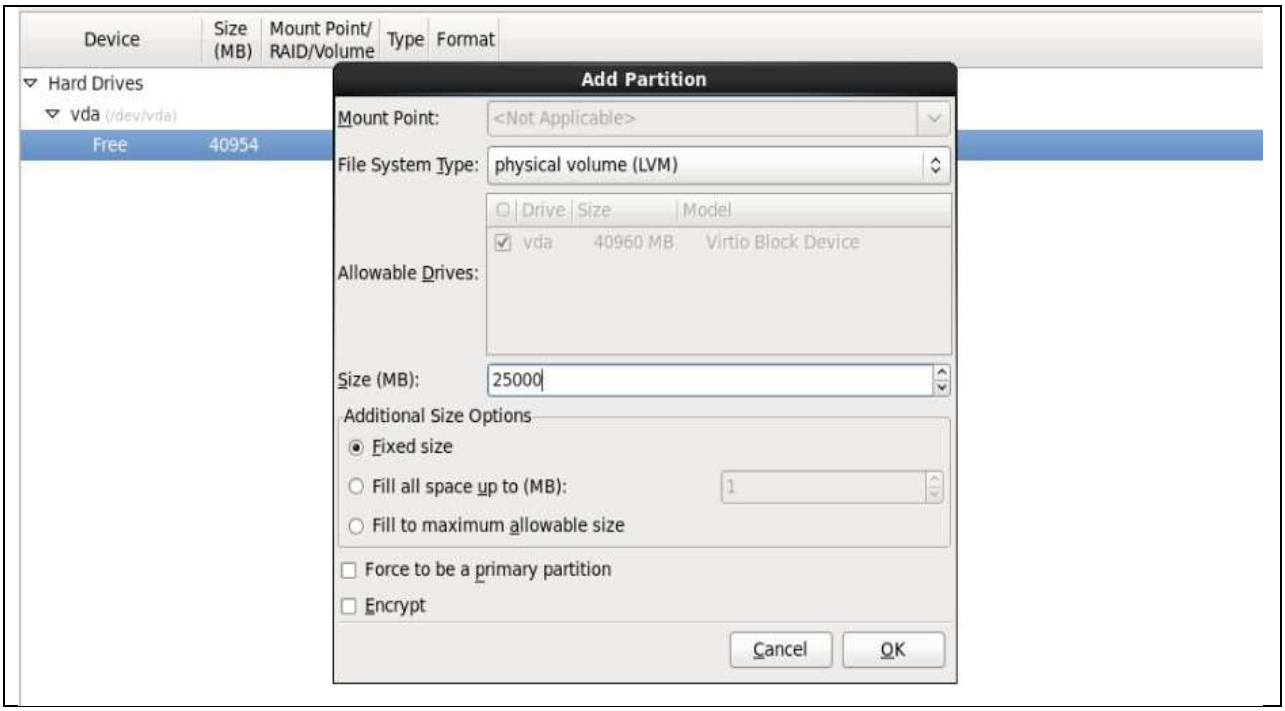
Installing Linux using LVM partitioning

- 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.

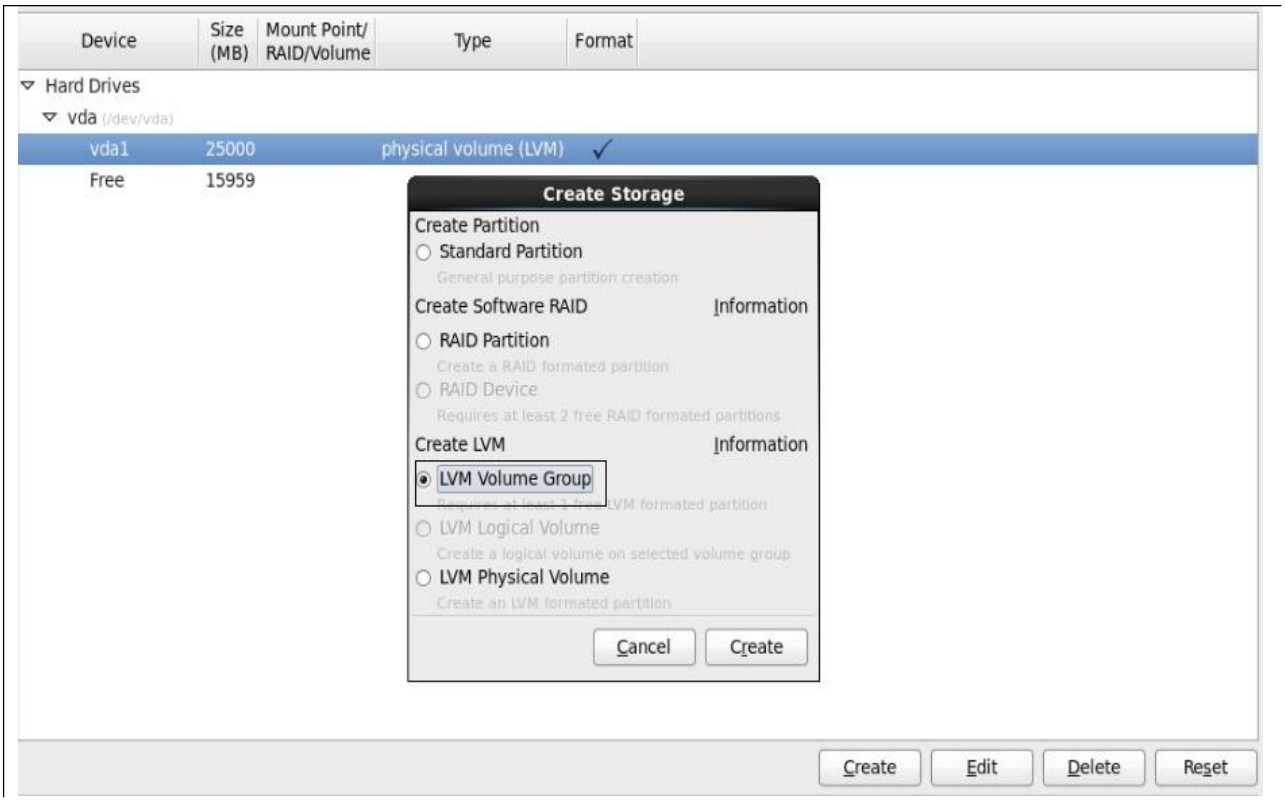
- Start the installation normally as done previously, but only at the time of partitioning follow the steps below.



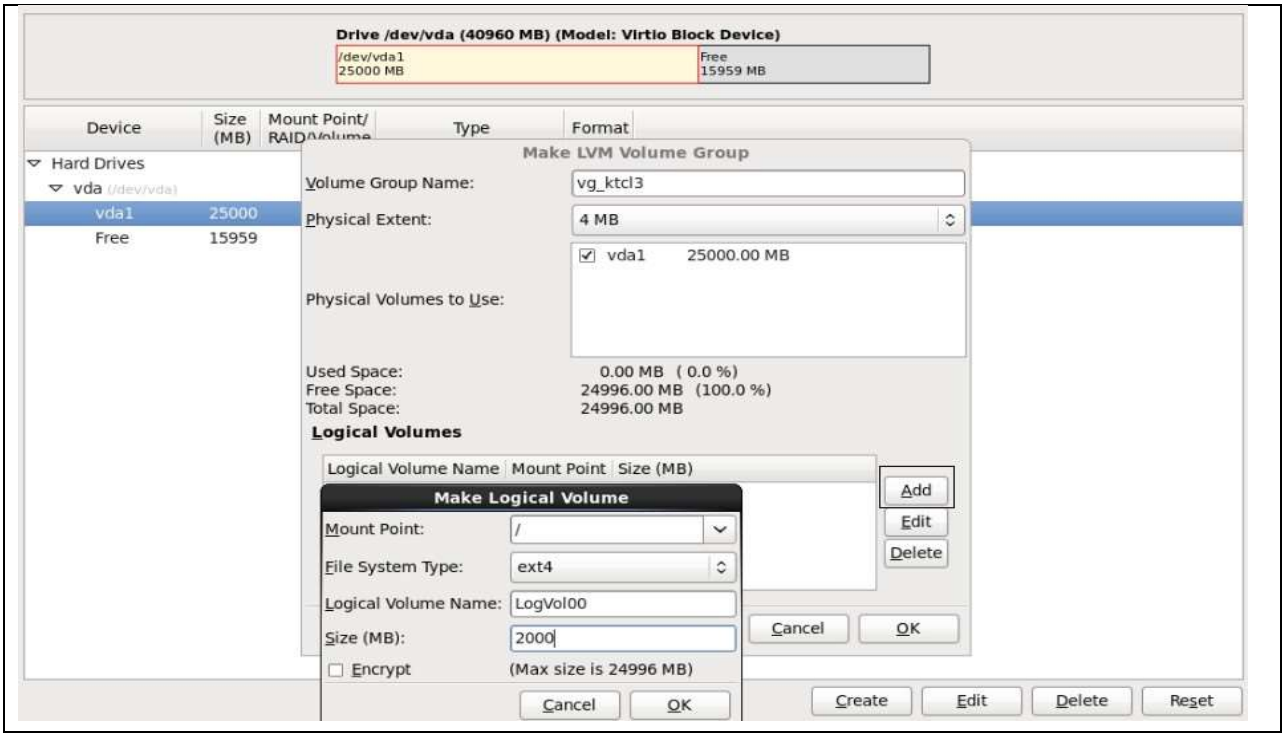
- Select the **Free** space and click on **Create,** then select **LVM Physical Volume** and click on **Create** to proceed.



- Give the maximum possible size to this **PV**, as all the partition has to be created inside it only.



- Select the created **PV, i.e., vda1** and this time check the box beside **LVM Volume Group** to create a volume group.

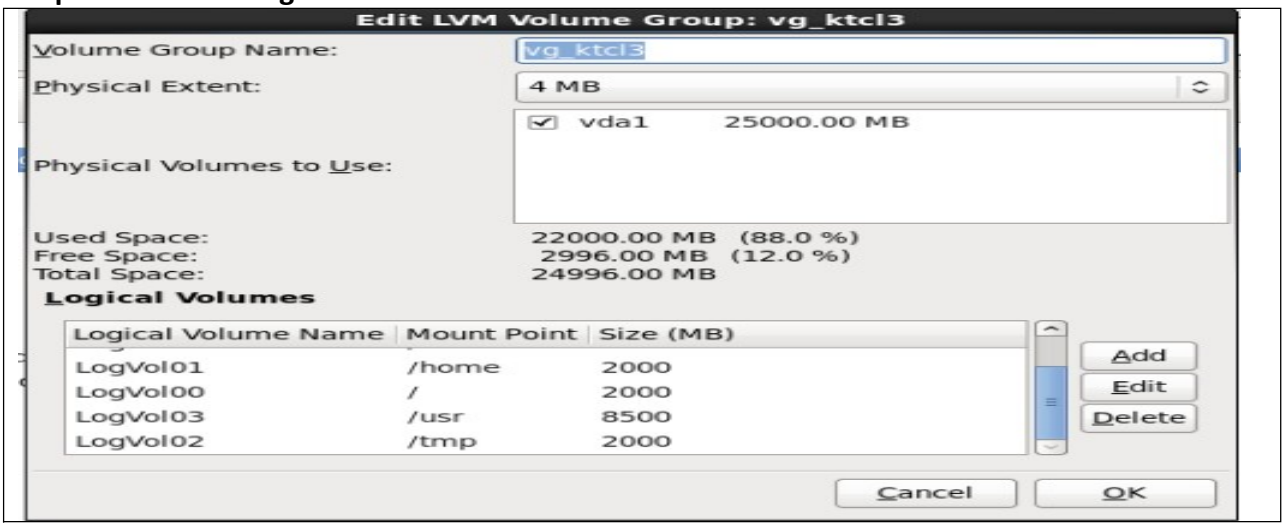


- Click on **Add** button to start adding **LVs**, Select a mount point and assign a size to it and click on **OK**

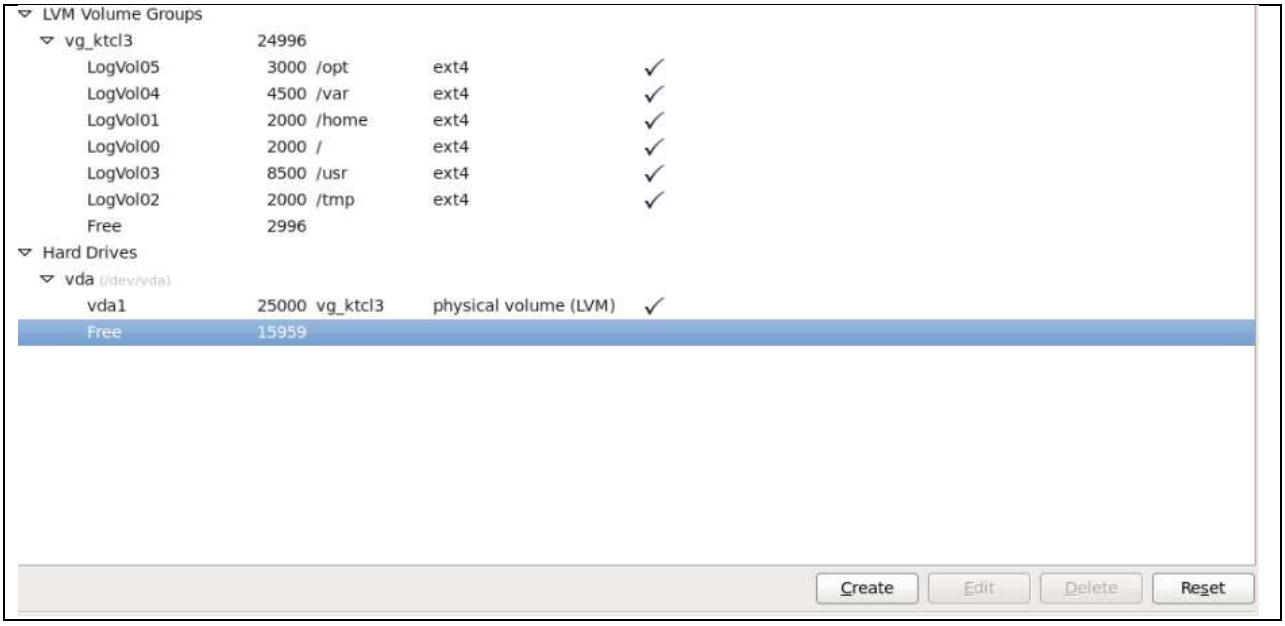
Repeat the above step and create the following partitions with the given sizes

- **/usr with 8.5 GB approx**
- **/var with 4.5 GB approx**
- **/opt with 2 GB approx**
- **/, /tmp, /home 2 GB each approx**
- **/opt 3 GB approx**

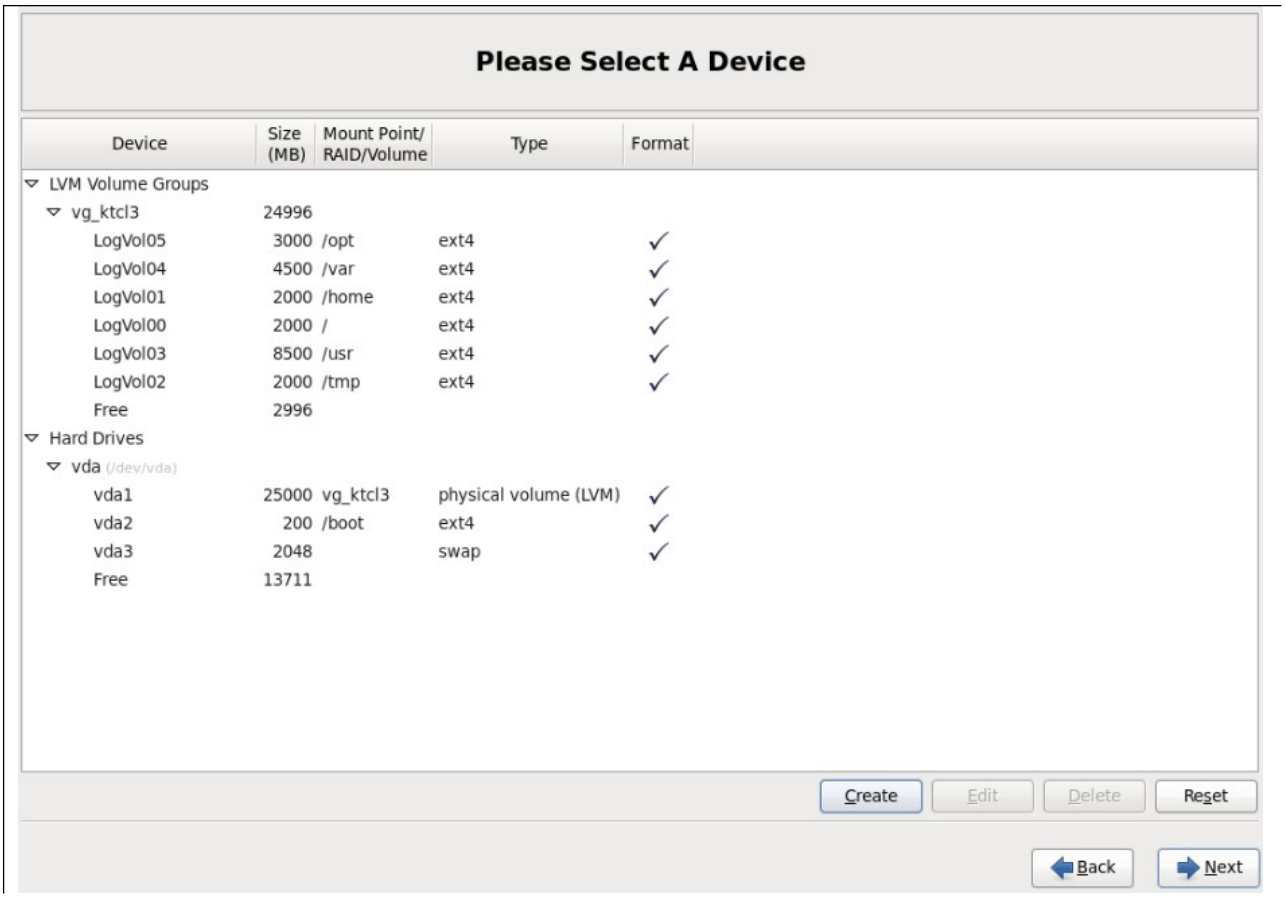
Note: All the sizes listed above are based on the availability of the space. It is no where a recommended or minimum sizes. The sizes can be based on your requirements. But /usr required the above given size if it is 64 bit architecture.



- Once done click on **OK** to conitinue



- Select the **Free** space under **Hard Drives** and create **/boot** with **200 MB** and **/Swap** with **2GB.** Make sure that you select **Standard Partition** this time, instead of **LVM.**



- Verify the sizes and click on **Next** to continue with the installation. Complete the installation as usual as we have done previously at the beginning of the course.

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 director**y 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 499**	**1 to 499**	**/var/ftp , etc**	**/sbin/nologin**
Normal User	**Visitor, ktuser,etc**	**500 to 60000**	**500 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 softwares 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)
- unique UID & GID are given to user

Linux uses UPG (User Private Group) scheme

- It means that whenever a user is created is has its own private group
- For Example if a user is created with the name **Rahul,** then a primary group for that user will be **Rahul** 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@ktlinux ~]# 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

The fields are as follows,

1. **root** = User name
2. **:$1fdsfsgsdfsdkffefje** = 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.

Password Complexity Requirements:

- A root user can change password of self and of any user in the system, there are no rules for root to assign a password. Root can assign any length of password either long or short, it can be alphabet or numeric or both. On the whole there is no limitation for root for assigning a password.

- A normal user can change only its password. Valid password for a normal user should adhere to the following rules
- It should be at least **7** characters but not more than **255** characters.
- At least one character should be **Upper case**
- At least one character should be **Lower case**
- At least one character should be a **symbol,** and one character should be a **number**.
- It should not match the previous password.
- It cannot have a sequence (ex: **123456** or **abcdef**)
- The **login name** and the **password** cannot be same.

Note: For security reasons don't keep the password based on **date of birth** because it can easily be hacked.

LAB WORK:-

Creating a user

- The syntax for creating a user in Linux is
- **# useradd <option> <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 ktusr**

```
[root@ktlinux ~]# useradd ktusr
[root@ktlinux ~]# tail /etc/passwd
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
ktuser:x:501:502::/home/ktuser:/bin/bash
named:x:25:25:Named:/var/named:/sbin/nologin
ktusr:x:502:503::/home/ktusr:/bin/bash
[root@ktlinux ~]#
```

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

Let's create a user with our own attributes

- Create a user with following attributes
- **Name** = **ktuser2**
- **uid=505**
- **home dir = /home/kernel**
- **comment =salesman**
- **#useradd ktuser2 –u 505 -g 505 –d /home/kernel –c salesman**

```
[root@ktlinux ~]# useradd ktuser2 -u 505 -d /home/kernel -c salesman
[root@ktlinux ~]# tail /etc/passwd
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
ktuser:x:501:502::/home/ktuser:/bin/bash
named:x:25:25:Named:/var/named:/sbin/nologin
ktusr:x:502:503::/home/ktusr:/bin/bash
ktuser2:x:505:505:salesman:/home/kernel:/bin/bash
[root@ktlinux ~]#
```

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@ktlinux ~]# passwd ktuser2
Changing password for user ktuser2.
New password:
BAD PASSWORD: it is based on a dictionary word
BAD PASSWORD: is too simple
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ktlinux ~]#
```

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 newname** to lock the user account
- **ex. # usermod -U newname** to unlock the user account
- **Note: - when an account is locked it will show! (Exclamation mark) in /etc/shadow file.**

Locking and unlocking a user account:

- To lock a user a/c use the following
- **#usermod –L < user name>**
- **#usermod –L ktuser2**
- Verify it in **/etc/shadow file**, it shows exclamation mark before user a/c or try login as ktuser2

```
[root@ktlinux ~]# usermod -L ktuser2
[root@ktlinux ~]# tail /etc/shadow
ntp:!!:14923::::::
pulse:!!:14923::::::
gdm:!!:14923::::::
sshd:!!:14923::::::
tcpdump:!!:14923::::::
visitor:$6$ONwZFaSl6WerWm2i$ULgPvVbt3.E8Ge.6jwTDQKTaQLvX5d
0Kiqj/6rq9DP1xelZIFyM6Mbwhy35GGem0:14923:0:99999:7:::
ktuser:$6$6jAEv8c9$j9VTJ1LPwDOuCuMIm6S2I7k3KdfAJktHGNB1akE
ecLDS3DAOvq074OFwIUXyWS2/:15234:0:99999:7:::
named:!!:15239::::::
ktusr:!!:15250:0:99999:7:::
ktuser2:!$6$sygiWqG7$uTphGmvQhScKQ8acThAMhbJuGiK9eRNBuBV4aI
.bhMNMZQA6GStxP1XSRyeTpKph.:15250:0:99999:7:::
[root@ktlinux ~]#
```

Unlocking a user a/c:

- Unlock the above a/c
- **#usermod –U < user name >**
- **#usermod –U ktuser2**
- Verify it in **/etc/shadow file**, it shows exclamation mark before user a/c or try login as ktuser2

```
[root@ktlinux ~]# usermod -U ktuser2
[root@ktlinux ~]# tail /etc/shadow
ntp:!!:14923::::::
pulse:!!:14923::::::
gdm:!!:14923::::::
sshd:!!:14923::::::
tcpdump:!!:14923::::::
visitor:$6$ONwZFaSl6WerWm2i$ULgPvVbt3.E8Ge.6jwTDQKTaQLvX5
0Kiqj/6rq9DP1xelZIFyM6Mbwhy35GGem0:14923:0:99999:7:::
ktuser:$6$6jAEv8c9$j9VTJ1LPwDOuCuMIm6S2I7k3KdfAJktHGNB1ak
ecLDS3DAOvq074OFwIUXyWS2/:15234:0:99999:7:::
named:!!:15239::::::
ktusr:!!:15250:0:99999:7:::
ktuser2:$6$XbHuW6gS$NO4lvd4XbZ76ZdvZFIUXxwBhQO80sshf664zg
asNnhz0zDKrt39Q5OJZnKaj6G1:15254:0:99999:7:::
[root@ktlinux ~]#
```

- **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 ktusr**

```
[root@ktlinux ~]# chage -l ktusr
Last password change                                    : Oct 03, 2011
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
[root@ktlinux ~]#
```

- **Last password change:** When the password was change 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 **"ktusr"** to
    - **Min password age : 2 days**
    - **Max password age: 7 days**
    - **Password expiration warnings: 2 days before password expires**
    - **Password inactive [-1]: 0 same day account is locked after password expiry.**
    - **A/C expiration date: 2011-12-31 (dec 31st 2011)**

- **#chage ktusr**

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

        Minimum Password Age [2]: 2
        Maximum Password Age [7]: 7
        Last Password Change (YYYY-MM-DD) [2011-10-03]:
        Password Expiration Warning [2]: 2
        Password Inactive [-1]: 0
        Account Expiration Date (YYYY-MM-DD) [1969-12-31]: 2011-12-31
[root@ktlinux ~]# chage -l ktusr
Last password change                                    : Oct 03, 2011
Password expires                                        : Oct 10, 2011
Password inactive                                       : Oct 10, 2011
Account expires                                         : Dec 31, 2011
Minimum number of days between password change          : 2
Maximum number of days between password change          : 7
Number of days of warning before password expires       : 2
[root@ktlinux ~]#
```

- 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.
    - **-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@ktlinux ~]# chage -E 2012-1-5 ktusr
[root@ktlinux ~]# chage -l ktusr
Last password change                                    : Oct 03, 2011
Password expires                                        : Oct 10, 2011
Password inactive                                       : Oct 10, 2011
Account expires                                         : Jan 05, 2012
Minimum number of days between password change          : 2
Maximum number of days between password change          : 7
Number of days of warning before password expires       : 2
[root@ktlinux ~]#
```

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

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 use the following command.
- **#userdel –r < user name >**
- **#userdel –r ktuser2**

```
[root@ktlinux ~]# userdel -r ktuser2
[root@ktlinux ~]# tree /home
/home
├── ktuser
│   ├── kernel1
│   ├── kernel2
│   ├── kernel3
│   ├── kernel4
│   └── kernel5
├── ktusr
├── lost+found
└── visitor

4 directories, 5 files
[root@ktlinux ~]#
```

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 ktgroup**

```
[root@ktlinux Desktop]# groupadd ktgroup
[root@ktlinux Desktop]# tail /etc/group
stapdev:x:491:
stapusr:x:490:
sshd:x:74:
tcpdump:x:72:
slocate:x:21:
visitor:x:500:
ktuser:x:501:
named:x:25:
ktusr:x:503:
ktgroup:x:504:
[root@ktlinux Desktop]#
```

Creating a group with user specified group id (GID)

- **#groupadd <option> <name for the group>**
- **#groupadd -g 595 ktgroup**
- Verify it in **/etc/group**

```
[root@ktlinux Desktop]# groupadd -g 595 ktgroup
[root@ktlinux Desktop]# tail /etc/group
stapdev:x:491:
stapusr:x:490:
sshd:x:74:
tcpdump:x:72:
slocate:x:21:
visitor:x:500:
ktuser:x:501:
named:x:25:
ktusr:x:503:
ktgroup:x:595:
[root@ktlinux Desktop]#
```

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 600 ktgroup**
- Verify it in /**etc/group**

```
[root@ktlinux Desktop]# groupmod -g 600 ktgroup
[root@ktlinux Desktop]# tail  /etc/group
stapdev:x:491:
stapusr:x:490:
sshd:x:74:
tcpdump:x:72:
slocate:x:21:
visitor:x:500:
ktuser:x:501:
named:x:25:
ktusr:x:503:
ktgroup:x:600:
[root@ktlinux Desktop]#
```

Changing the name of the group

- The syntax for changing the group name is
- **#groupmod –n <new name > < existing name >**
- **#groupmod –n kernelgrp ktgroup**

```
[root@ktlinux Desktop]# groupmod -n kernelgrp ktgroup
[root@ktlinux Desktop]# tail -5  /etc/group
visitor:x:500:
ktuser:x:501:
named:x:25:
ktusr:x:503:
kernelgrp:x:600:
[root@ktlinux Desktop]#
```

Adding and Removing Members to a Group

- Adding the members to the group is to add users to the group. To add the members to the group the syntaxes are
- **To add single user to the group**
- **#usermod –G <group name > < user name>**
- **#usermod –G ktgroup ktuser**

```
[root@ktlinux Desktop]# usermod -G ktgroup ktuser
[root@ktlinux Desktop]# grep ktgroup /etc/group
ktgroup:x:600:ktuser
[root@ktlinux Desktop]#
```

- **Adding multiple 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 group Administrator**
- **-a for Adding a single user to a group**
- **-d removing a user from a group**

- **#gpasswd –M <user>,<user>,<user> <group>**
- **#gpasswd –M ktuser2,ktuser3,ktuser4 ktgroup**

```
[root@ktlinux Desktop]# gpasswd -M ktuser2,ktuser3,ktuser4 ktgroup
[root@ktlinux Desktop]# tail -5  /etc/group
ktusr:x:503:
ktuser2:x:601:
ktuser3:x:504:
ktuser4:x:505:
ktgroup:x:600:ktuser2,ktuser3,ktuser4
[root@ktlinux Desktop]#
```

Adding a single user using gpasswd

- **#gpasswd –a ktuser ktgroup (**verify it in **/etc/group)**

```
[root@ktlinux Desktop]# gpasswd -a ktuser ktgroup
Adding user ktuser to group ktgroup
[root@ktlinux Desktop]# grep ktgroup /etc/group
ktgroup:x:600:ktuser2,ktuser3,ktuser4,ktuser
[root@ktlinux Desktop]#
```

Making a user as a administrator

- **#gpasswd –A ktuser ktgroup (verify it in /etc/gshadow)**

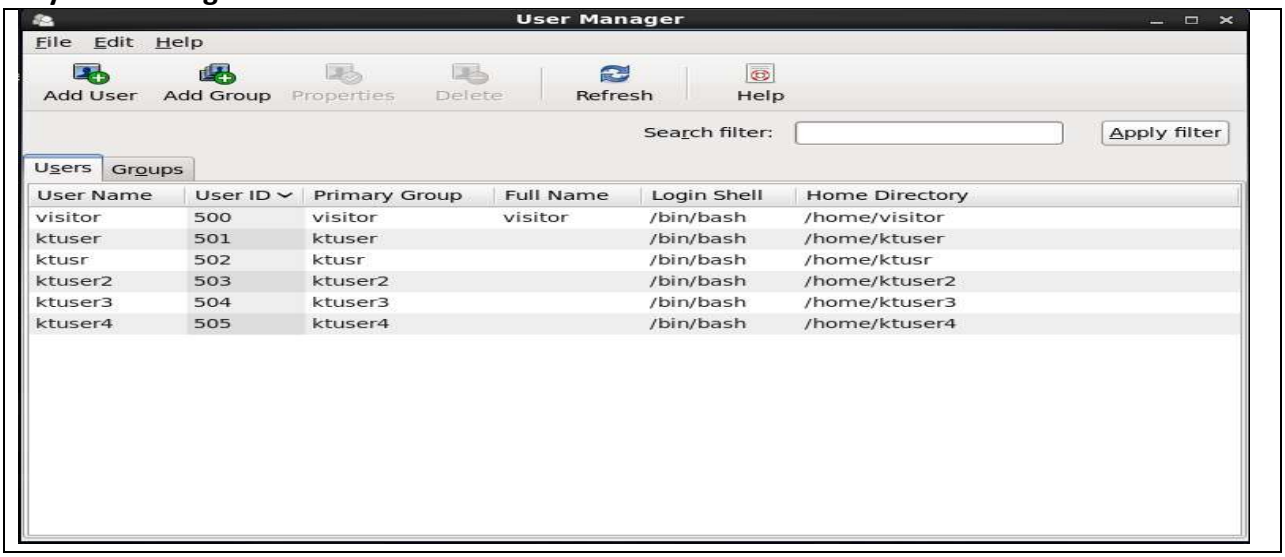
```
[root@ktlinux Desktop]# gpasswd -A ktuser ktgroup
[root@ktlinux Desktop]# grep ktgroup /etc/gshadow
ktgroup:!:ktuser:ktuser2,ktuser3,ktuser4,ktuser
[root@ktlinux Desktop]#
```

Removing a user from the group

- **#gpasswd –d ktuser2 ktgroup**

```
[root@ktlinux Desktop]# grep ktgroup /etc/group
ktgroup:x:600:ktuser2,ktuser3,ktuser4,ktuser
[root@ktlinux Desktop]# gpasswd -d ktuser2 ktgroup
Removing user ktuser2 from group ktgroup
[root@ktlinux Desktop]# grep ktgroup /etc/group
ktgroup:x:600:ktuser3,ktuser4,ktuser
[root@ktlinux Desktop]#
```

To add and remove groups use can also use the graphical tool in linux
#system-config-users &



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 ping command is having suid, so all users can run that command but if suid is removed and a normal user wants to user execute it, then it will show **'operation not permitted'**

```
[root@ktlinux Desktop]# which ping
/bin/ping
[root@ktlinux Desktop]# ls -l /bin/ping
-rwsr-xr-x. 1 root root 41432 Jul 27  2010 /bin/ping
[root@ktlinux Desktop]#
```

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

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

```
[root@ktlinux Desktop]# chmod u-s /bin/ping
[root@ktlinux Desktop]# su - ktuser2
[ktuser2@ktlinux ~]$ ping 192.168.10.95
ping: icmp open socket: Operation not permitted
[ktuser2@ktlinux ~]$
```

- **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@ktlinux /]# mkdir ktsdir
[root@ktlinux /]# chgrp ktgroup ktsdir
[root@ktlinux /]# ls -ld ktsdir
drwxr-xr-x 2 root ktgroup 4096 Oct  8 07:32 ktsdir
[root@ktlinux /]# chmod g+s ktsdir
[root@ktlinux /]# ls -ld ktsdir
drwxr-sr-x 2 root ktgroup 4096 Oct  8 07:32 ktsdir
[root@ktlinux /]# chmod go+w ktsdir
[root@ktlinux /]# su - ktuser3
[ktuser3@ktlinux ~]$ cd /ktsdir
[ktuser3@ktlinux ktsdir]$ touch file{1..5}
[ktuser3@ktlinux ktsdir]$ ls -l
total 0
-rw-rw-r-- 1 ktuser3 ktgroup 0 Oct  8 07:34 file1
-rw-rw-r-- 1 ktuser3 ktgroup 0 Oct  8 07:34 file2
-rw-rw-r-- 1 ktuser3 ktgroup 0 Oct  8 07:34 file3
-rw-rw-r-- 1 ktuser3 ktgroup 0 Oct  8 07:34 file4
-rw-rw-r-- 1 ktuser3 ktgroup 0 Oct  8 07:34 file5
[ktuser3@ktlinux ktsdir]$
```

Note: when a file is created by any user it will get the group as primary group of the owner which is usually owner's private group with same name.

- **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 the file or directory.

- Let see it practically.

```
[root@ktlinux /]# chmod o+t ktsdir
[root@ktlinux /]# ls -ld ktsdir
drwxrwsrwt 2 root ktgroup 4096 Oct  8 07:34 ktsdir
[root@ktlinux /]# su - wshr
[wshr@ktlinux ~]$ cd /ktsdir
[wshr@ktlinux ktsdir]$ ls
file1  file2  file3  file4  file5
[wshr@ktlinux ktsdir]$ rm file1
rm: remove write-protected regular empty file `file1'? y
rm: cannot remove `file1': Operation not permitted
[wshr@ktlinux ktsdir]$
```

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.
- ACL can be applied on ACL enabled partition that means you need to enable ACL while mounting the partition.

Steps to implement ACL

- Create a partition and format it with ext4 file system
- Mount a file system with ACL
- Apply ACL on it.

Let's implement it practically.

- Create a partition and format it with ext4 file system

```
[root@ktlinux ~]# parted -l /dev/sda
Model: VMware Virtual disk (scsi)
Disk /dev/sda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system     Flags
 1      1049kB  31.5GB  31.5GB  primary                   lvm
 2      31.5GB  31.7GB  210MB   primary   ext4            boot
 3      31.7GB  35.9GB  4194MB  primary   linux-swap(v1)
 4      35.9GB  53.7GB  17.8GB  extended
 5      35.9GB  36.4GB  534MB   logical
 6      36.4GB  36.9GB  535MB   logical   linux-swap(v1)
 7      36.9GB  37.5GB  535MB   logical
 8      37.5GB  38.0GB  535MB   logical

[root@ktlinux ~]# mkfs.ext4 /dev/sda7
mke2fs 1.41.12 (17-May-2010)
```

- Mount it with ACL option
- #mount –o acl /dev/sda7 /ktdir
- If the partition is already mounted and you want add acl on it use following command

```
[root@ktlinux ~]# mount -o acl /dev/sda7 /ktdir
[root@ktlinux ~]# mount
/var/named on /var/named/chroot/var/named type none (rw,bind)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/sda2 on /boot type ext4 (rw)
/dev/mapper/vg_ktlinux-homelv on /home type ext4 (rw)
/dev/mapper/ktpart on /kernel type ext4 (rw)
/dev/sda7 on /ktdir type ext4 (rw,acl)
[root@ktlinux ~]#
```

- To make it permanent make following entry in **/etc/fstab**

```
/dev/mapper/ktpart      /kernel                 ext4    defaults        0 0
/dev/sda6               swap                    swap    defaults        0 0
/dev/sda7               /ktdir                  ext4    defaults,acl    0 0
```

- If your partition already exists, then just add an **acl** after **defaults** as shown above and use the following command.
- **#mount –o remount /dev/sda7**

- Now check the default permission and acl permission on /ktdir
- #ls –ld /ktdir
- To check the acl permission syntax is
- **#getfacl <option> <dir/file name>**
- **Options:**
- **-d** Displays the default ACL
- **-R** Recurses into subdirectories

#getfacl /ktdir

```
[root@ktlinux ~]# ls -ld /ktdir
drwxr-xr-x 3 root root 1024 Oct 10 02:42 /ktdir
[root@ktlinux ~]# getfacl /ktdir
getfacl: Removing leading '/' from absolute path names
# file: ktdir
# 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@ktlinux ~]# chmod 777 /ktdir
[root@ktlinux ~]# ls -ld /ktdir
drwxrwxrwx 3 root root 1024 Oct 10 02:42 /ktdir
[root@ktlinux ~]#
```

- 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 ACL
- **-R** Recurses into subdirectories

- **The possible arguments are**
- **u: user**
- **g: group**
- **o: others**

Note: Whatever ACL permissions assigned to a user or group or others, it will be treated as Normal Permissions minus ACL

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 u:ktuser: rx ktdir**
- Verify it by using **getfacl** command

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

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

```
[root@ktlinux ~]# su - ktuser
[ktuser@ktlinux ~]$ cd /ktdir
[ktuser@ktlinux ktdir]$ touch file1
touch: cannot touch `file1': Permission denied
[ktuser@ktlinux ktdir]$ ls -ld /ktdir
drwxrwxrwx+ 3 root root 1024 Oct 10 02:42 /ktdir
[ktuser@ktlinux ktdir]$
```

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:ktgroup:rwx /ktdir**

```
[root@ktlinux /]# setfacl -m g:ktgroup:rwx /ktdir
[root@ktlinux /]# getfacl ktdir
# file: ktdir
# owner: root
# group: root
user::rwx
user:ktuser:r-x
group::rwx
group:ktgroup: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:ktuser:rx,g:ktgroup:rx /ktdir**

```
[root@ktlinux /]# setfacl -m u:ktuser:rx,g:ktgroup:rx /ktdir
[root@ktlinux /]# getfacl ktdir
# file: ktdir
# owner: root
# group: root
user::rwx
user:ktuser:r-x
group::rwx
group:ktgroup: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 u:ktuser /ktdir**

```
[root@ktlinux /]# setfacl -x u:ktuser /ktdir
[root@ktlinux /]# getfacl ktdir
# file: ktdir
# owner: root
# group: root
user::rwx
group::rwx
group:ktgroup:r-x
mask::rwx
other::rwx
```

Removing acl for a particular group

- **#setfacl –x g:<group name> <directory name>**
- **#setfacl –x g: ktgroup /ktdir**

```
[root@ktlinux /]# setfacl -x g:ktgroup /ktdir
[root@ktlinux /]# getfacl ktdir
# file: ktdir
# 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@ktlinux /]# setfacl -m u:ktuser:rx,g:ktgroup:rx /ktdir
[root@ktlinux /]# getfacl ktdir
# file: ktdir
# owner: root
# group: root
user::rwx
user:ktuser:r-x
group::rwx
group:ktgroup:r-x
mask::rwx
other::rwx

[root@ktlinux /]# setfacl -b /ktdir
[root@ktlinux /]# getfacl ktdir
# file: ktdir
# owner: root
# group: root
user::rwx
group::rwx
other::rwx

[root@ktlinux /]#
```

ACL can also be applied to a file in exactly similar fashion as we did for a directory.

This part confirms the end of USER ADMINISTRATION

Network Configuration & Trouble Shooting

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)**

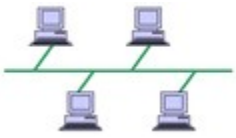
   A **network interface controller** (also known as a **network interface card**, **network adapter**, **LAN adapter** and by similar terms) is a computer hardware component that connects a computer to a computer network. Each NIC will be having a unique MAC addresses (**Media Access Control address)**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



Bus Network Topology

Ring Network Topology

Star Network Topology

Mesh Network Topology

Tree Network Topology

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.

Example 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.

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.

Netmask:	255.	255.	255.	0
Binary:	11111111	11111111	11111111	00000000
Netmask length	8	16	24	--

- **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.

```
[root@ktlinux network-scripts]# ls
ifcfg-eth0   ifdown-isdn    ifup-aliases  ifup-plusb     init.ipv6-global
ifcfg-lo     ifdown-post    ifup-bnep     ifup-post      net.hotplug
ifdown       ifdown-ppp     ifup-eth      ifup-ppp       network-functions
ifdown-bnep  ifdown-routes  ifup-ippp     ifup-routes    network-functions-ipv6
ifdown-eth   ifdown-sit     ifup-ipv6     ifup-sit
ifdown-ippp  ifdown-tunnel  ifup-isdn     ifup-tunnel
ifdown-ipv6  ifup           ifup-plip     ifup-wireless
```

#/etc/sysconfig/network is a file 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.

```
[root@ktlinux ~]# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=ktlinux.kt.com
[root@ktlinux ~]#
```

#/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.

```
[root@ktlinux ~]# cat /etc/hosts
192.168.10.98   ktlinux.kt.com  ktlinux # Added by NetworkManager
127.0.0.1       localhost.localdomain   localhost
::1     ktlinux.kt.com  ktlinux localhost6.localdomain6 localhost6
```

#/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.

```
[root@ktlinux ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search kt.com
nameserver 192.168.10.98
```

LAB WORK:-

- **To check the ip address assign to all the interfaces**
  #ifconfig

```
[root@ktlinux ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:3C:2F:1E
          inet addr:192.168.10.98  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3c:2f1e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:29686 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1866 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2102329 (2.0 MiB)  TX bytes:15481314 (14.7 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:252 errors:0 dropped:0 overruns:0 frame:0
          TX packets:252 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19656 (19.1 KiB)  TX bytes:19656 (19.1 KiB)
```

- **To chech the ip of a particular interface**
  #ifconfig < adapter name >
  #ifconfig eth0

```
[root@ktlinux ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:3C:2F:1E
          inet addr:192.168.10.98  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3c:2f1e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:36560 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3780 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2550422 (2.4 MiB)  TX bytes:24606368 (23.4 MiB)
```

- **To check the hostname of the system.**
  #hostname

```
[root@ktlinux ~]# hostname
ktlinux.kt.com
```

- **To check ip of the host**
  #hostname –i

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

  #host <hostname>
  #host ktlinux.kt.com

```
[root@ktlinux ~]# host ktlinux.kt.com.
ktlinux.kt.com has address 192.168.10.98
```

- **Same with "nslookup" command**
  #nslookup < ip address >
  #nslookup < hostname >

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

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

```
[root@ktlinux ~]# nlookup ktlinux.kt.com
bash: nlookup: command not found
[root@ktlinux ~]# nslookup ktlinux.kt.com
Server:         192.168.10.98
Address:        192.168.10.98#53

Name:   ktlinux.kt.com
Address: 192.168.10.98
```

- The most common command used to check DNS function is "**dig"**
  #dig <hostname>

```
[root@ktlinux ~]# dig ktlinux.kt.com

; <<>> DiG 9.7.0-P2-RedHat-9.7.0-5.P2.el6 <<>> ktlinux.kt.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:
;ktlinux.kt.com.                        IN      A

;; ANSWER SECTION:
ktlinux.kt.com.         10800   IN      A       192.168.10.98

;; AUTHORITY SECTION:
ktlinux.kt.com.         10800   IN      NS      ktlinux.kt.com.
```

With ip address
#dig –x <ip address>
#dig –x 192.168.10.98

```
[root@ktlinux ~]# 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.kt.com.
98.10.168.192.in-addr.arpa. 10800 IN    PTR     kt.com.

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

- **Checking network connectivity using ping command**
  #ping < ip address >
  #ping 192.168.10.95

```
[root@ktlinux ~]# 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@ktlinux ~]# 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 kernellinux.kt.com

```
[root@ktlinux ~]# hostname
ktlinux.kt.com
[root@ktlinux ~]# hostname kernellinux.kt.com
[root@ktlinux ~]# hostname
kernellinux.kt.com
[root@ktlinux ~]#
```

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** file and then logout and login to confirm the change.

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

```
NETWORKING=yes
HOSTNAME=kernellinux.kt.com
```

- Now logoff and logon and check the hostname

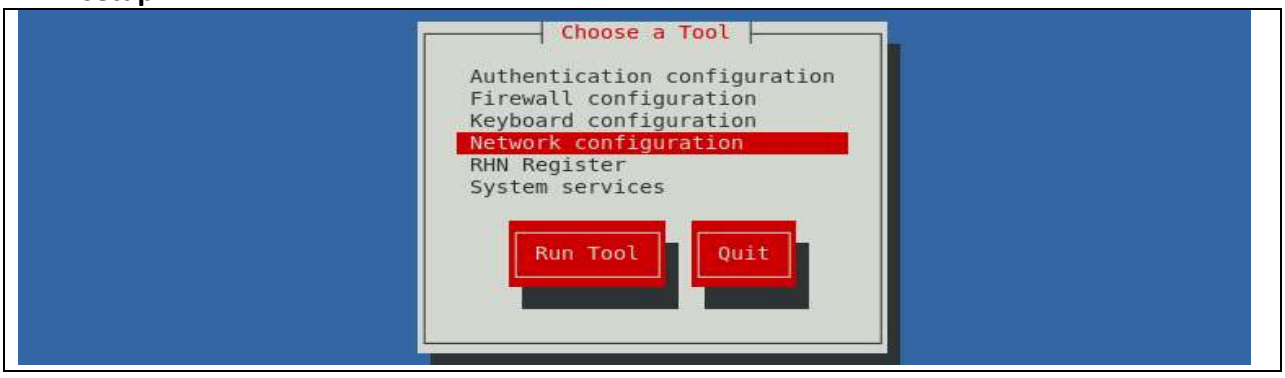
```
[root@kernellinux ~]# hostname
kernellinux.kt.com
[root@kernellinux ~]#
```

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

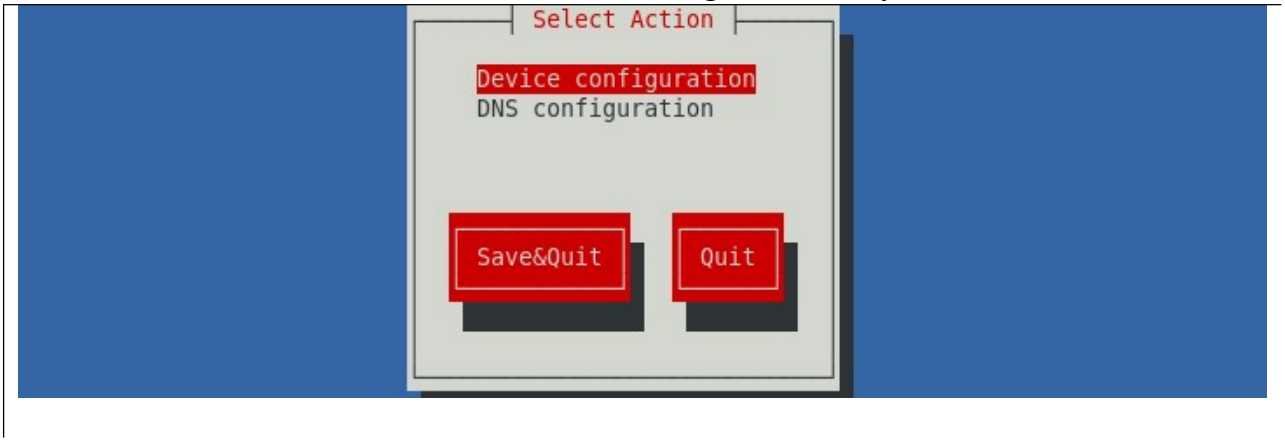
Assigning /Changing the IP Address

Steps for changing the IP Address.

- To change the IP Address use the following utilily
- **#setup** or **#system-config-network**
- It will open a text base utility follow the steps below and change the ip address
- Restart the network service to apply the changes
- Make the network service starts after reboot.
- Let's begin with **setup**
  #setup

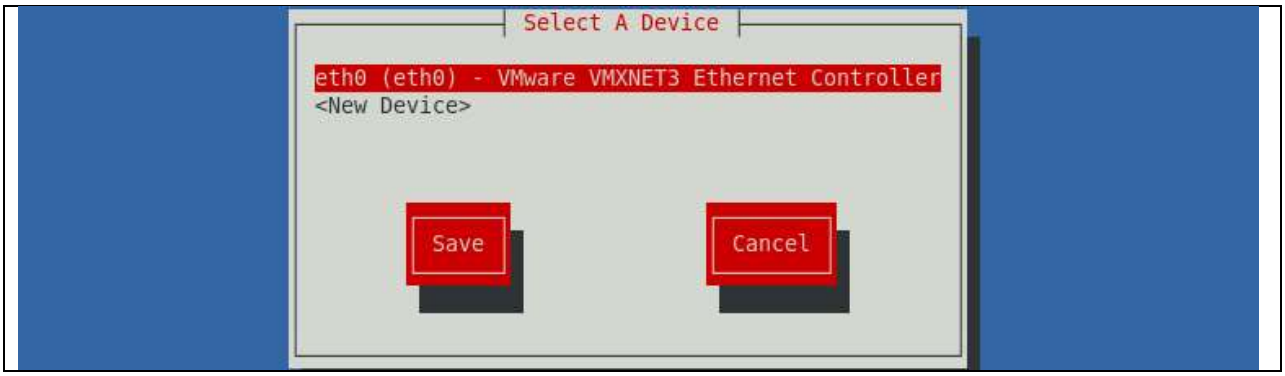


Move the cursor to Network configuration and press Enter

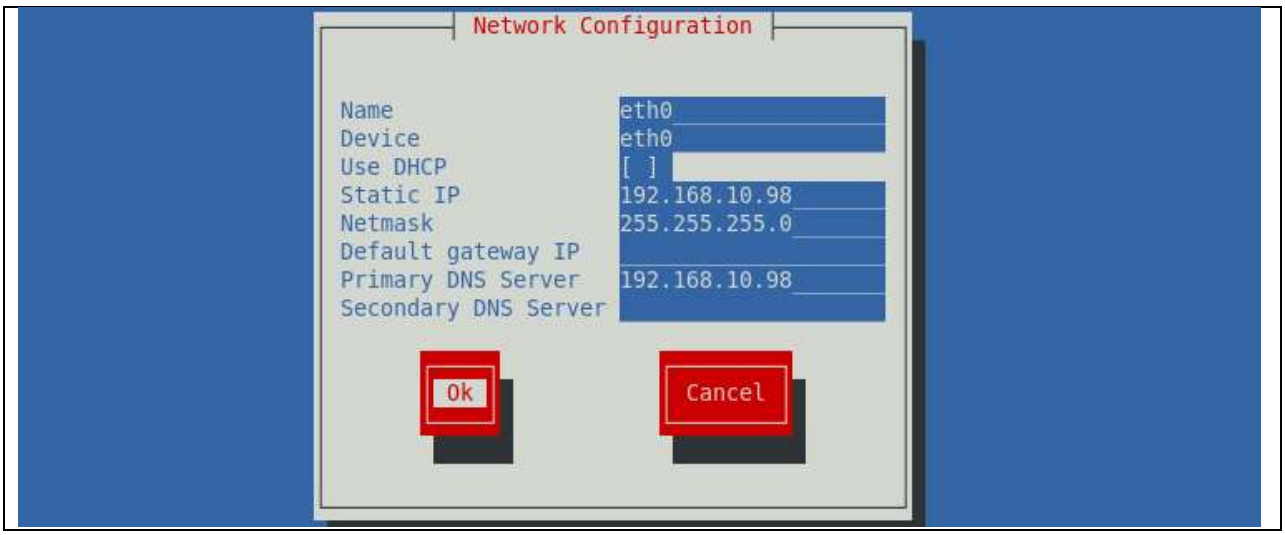


Move the cursor to Device configuration and press Enter

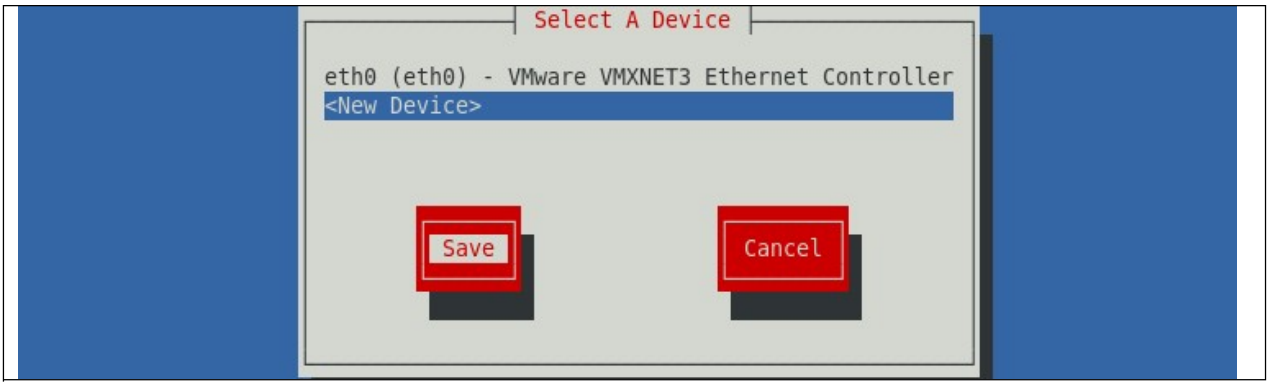
Note: If **system-config-network** command is used, it will directly take you to above position.



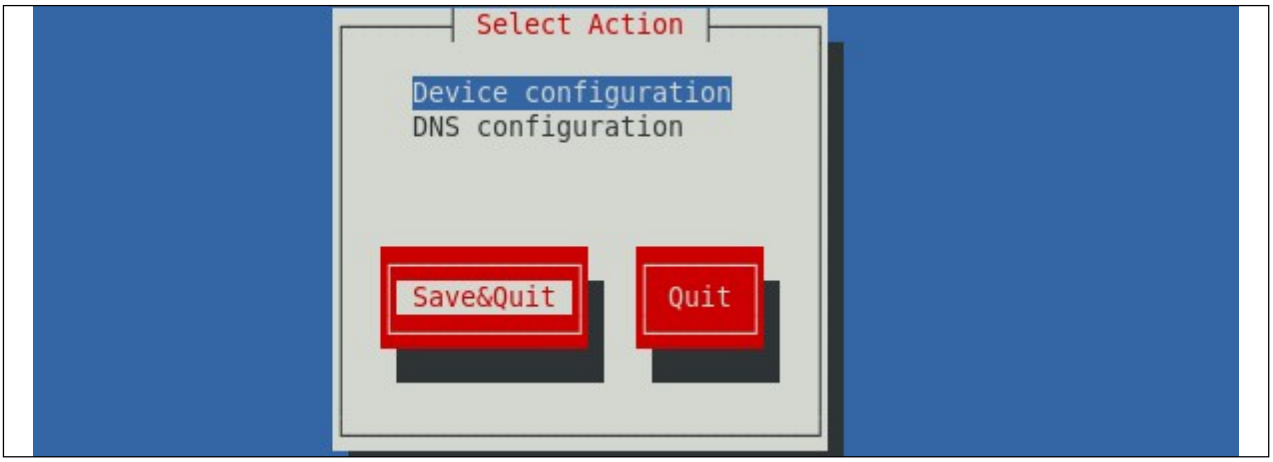
Now select the NIC adapter i.e. eth0 and press Enter



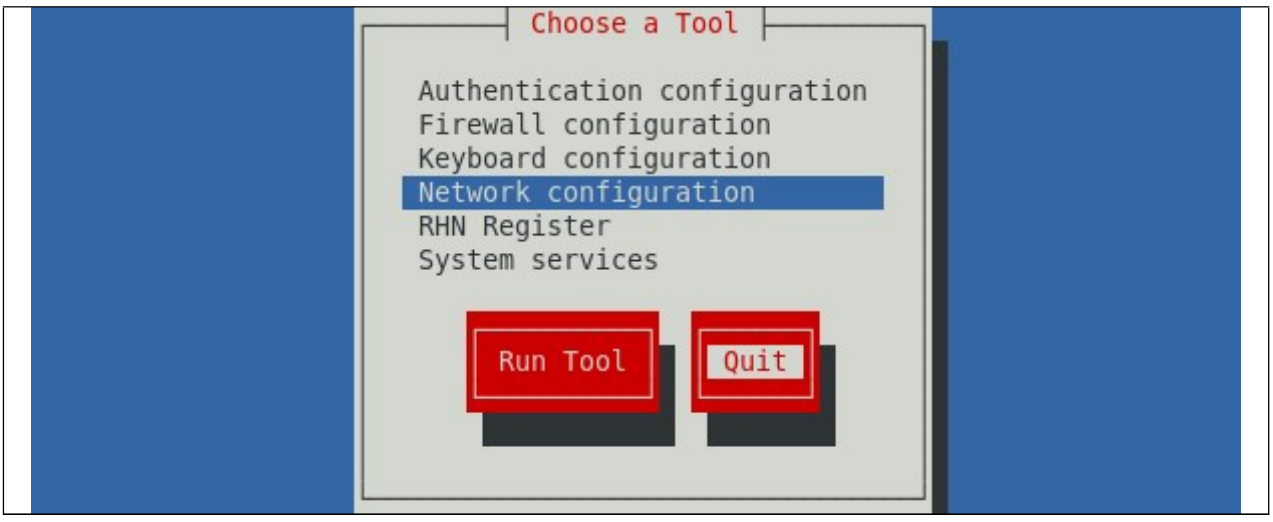
Assign the above ip address and other details as per your requirement, move the cursor to "OK" and press Enter



Move the cursor to “save” to save the changes in device configuration and press Enter.



Once again move the cursor to “Save&Quit” button and press Enter



Finally Move the cursor to “Quit” button and Press Enter to quit the utility.

- **Now restart the network service and check for the ip address**
  #service network restart
  If the change is not reflected with above service restart, restart the network manager **#service NetworkManager restart** (N and M are case sensitive)

```
[root@kernellinux Desktop]# service network restart
Shutting down interface eth0:  Device state: 3 (disconnected)
                                                           [  OK  ]
Shutting down loopback interface:                          [  OK  ]
Bringing up loopback interface:                            [  OK  ]
Bringing up interface eth0:  Active connection state: activated
Active connection path: /org/freedesktop/NetworkManager/ActiveConnection/1
                                                           [  OK  ]
[root@kernellinux Desktop]# service NetworkManager restart
Stopping NetworkManager daemon:                            [  OK  ]
Setting network parameters...                              [  OK  ]
Starting NetworkManager daemon:                            [  OK  ]
[root@kernellinux Desktop]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:3C:2F:1E
          inet addr:192.168.10.98  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3c:2f1e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:72158 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17208 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4877336 (4.6 MiB)  TX bytes:73419371 (70.0 MiB)
```

- The above picture confirms that we have successfully assigned an IP address to a machine.
- You can also check the functioning of newly assigned IP address by pinging it from other machines in the network.
- If it is not pinging from outside then check whether the cable is connected properly or not.
- If the server is in the remote location use **#mii-tool** to check whether the cable is connected or not

```
[root@kernellinux]# mii-tool
eth0: negotiated 100baseTx-FD, link ok
```

- To Know more about the NIC card/adapter use
  #ethtool <adapter name>

```
[root@kernellinux Desktop]# ethtool eth0
Settings for eth0:
        Supported ports: [ TP ]
        Supported link modes:   1000baseT/Full
                                10000baseT/Full
        Supports auto-negotiation: No
        Advertised link modes:  Not reported
        Advertised pause frame use: No
        Advertised auto-negotiation: No
        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
```

This sums up the Networking Chapter

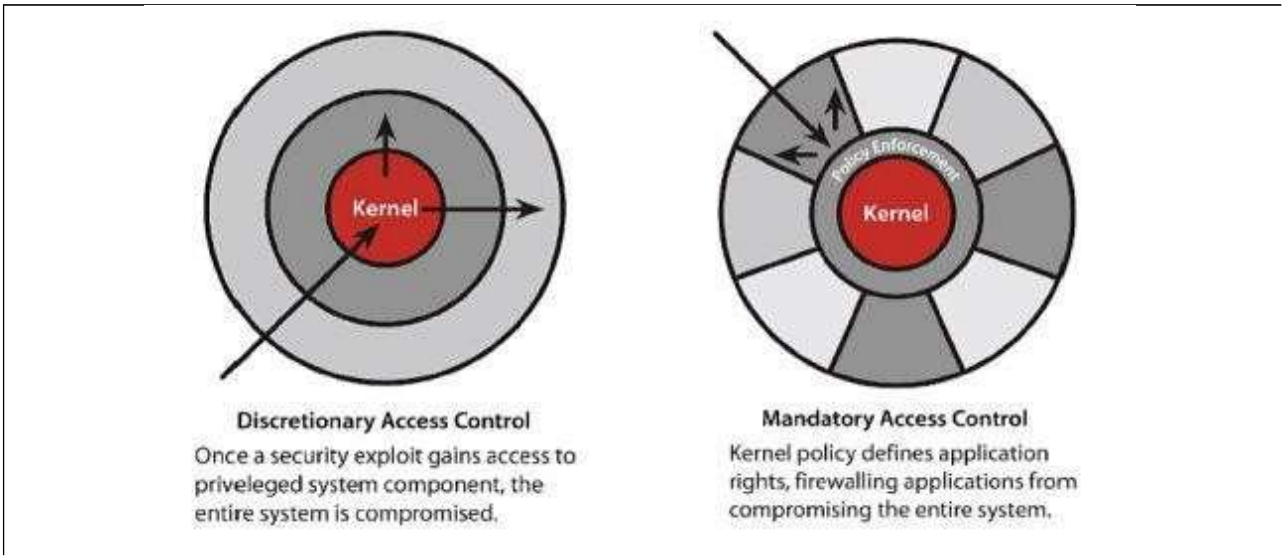
MANAGING SELINUX (BASICS OF SELINUX)

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.

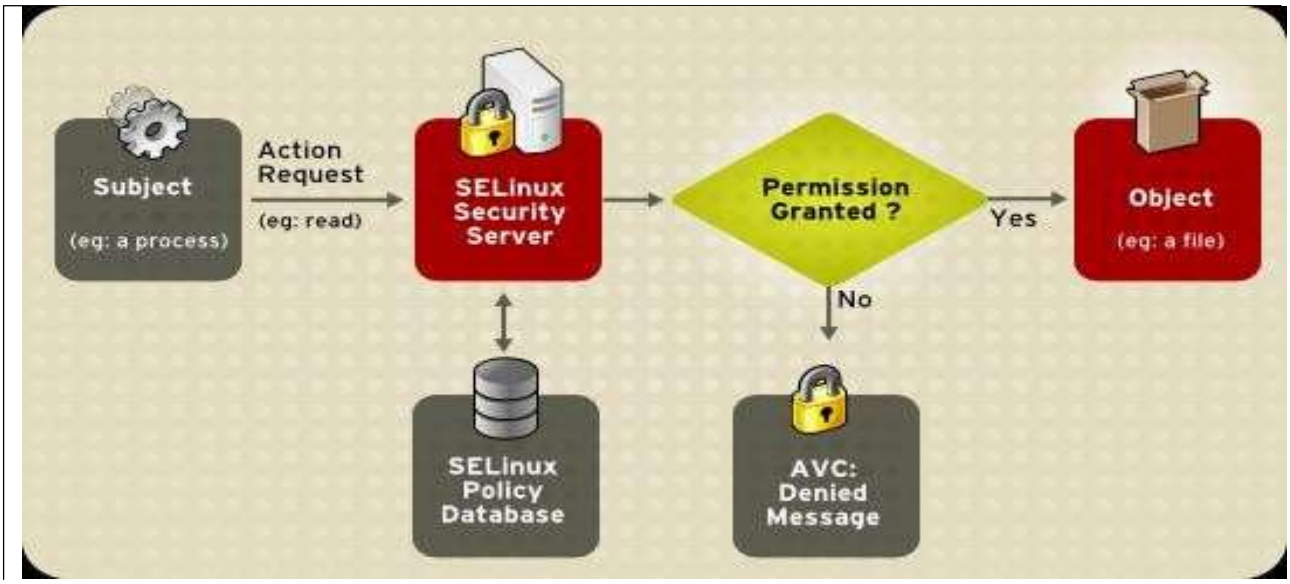
- SELinux also adds finer granularity to access controls. Instead of only being able to specify who can read, write or execute a file, for example, SELinux lets you specify who can unlink, append only, and move a file and so on. SELinux allows you to specify access to many resources other than files 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@ktlinux ~]# 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@ktlinux ~]# getenforce
Enforcing
[root@ktlinux ~]#
```

#sestatus

```
[root@ktlinux ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
[root@ktlinux ~]#
```

Display the SELinux context of a file or directory.

- To display the context of a file the syntax is

  #ls –Z <filename>

```
[root@ktlinux ~]# 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@ktlinux ~]# ls -ldZ Documents
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Documents
[root@ktlinux ~]#
```

Displaying the SELinux Context of a Process

- To display the context of a process running in the system, the syntax is

  #ps –efZ |grep <process name>

  #ps –efZ |grep http

```
[root@ktlinux ~]# ps -efZ |grep http
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 root 24940 10112  0 17:37 pts/0 00:00:00 gr
ep http
[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@ktlinux ~]# ls -ldZ ktfile
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 ktfile
```

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 ktfile

```
[root@ktlinux ~]# chcon -t public_content_t ktfile
[root@ktlinux ~]# ls -ldZ ktfile
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 ktfile
[root@ktlinux ~]#
```

- To change the context for a directory and its contents
- Check the context of both directory and its content**s**

```
[root@ktlinux ~]# ls -ldZ ktdir
drwxr-xr-x. root root system_u:object_r:admin_home_t:s0 ktdir
[root@ktlinux ~]# ls -lZ ktdir
-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@ktlinux ~]#
```

- 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 ktdir

```
[root@ktlinux ~]# chcon -R -t public_content_t ktdir
[root@ktlinux ~]# ls -ldZ ktdir
drwxr-xr-x. root root system_u:object_r:public_content_t:s0 ktdir
[root@ktlinux ~]# ls -lZ ktdir
-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@ktlinux ~]#
```

Restoring back the modified SELinux context to its default value

- To restore the modified/changed SELinux context of a file to its default form, the syntax is
  #restorecon –v <filename>
  #restorecon –v ktfile

```
[root@ktlinux ~]# ls -ldZ ktfile
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 ktfile
[root@ktlinux ~]# restorecon -v ktfile
restorecon reset /root/ktfile context unconfined_u:object_r:public_content_t:s0->system_u:object_
r:admin_home_t:s0
[root@ktlinux ~]# ls -ldZ ktfile
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 ktfile
[root@ktlinux ~]#
```

- To restore back the same of a directory with its contents, the syntax is
  #restorecon –Rv <dir name >
  #resotrecon –Rv ktdir

```
drwxr-xr-x. root root system_u:object_r:public_content_t:s0 ktdir
[root@ktlinux ~]# ls -lZ ktdir
-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@ktlinux ~]# restorecon -Rv ktdir
restorecon reset /root/ktdir context system_u:object_r:public_content_t:s0->system_u:object_r:adm
in_home_t:s0
restorecon reset /root/ktdir/file4 context system_u:object_r:public_content_t:s0->system_u:object
_r:admin_home_t:s0
restorecon reset /root/ktdir/file3 context system_u:object_r:public_content_t:s0->system_u:object
_r:admin_home_t:s0
restorecon reset /root/ktdir/file1 context system_u:object_r:public_content_t:s0->system_u:object
_r:admin_home_t:s0
restorecon reset /root/ktdir/file5 context system_u:object_r:public_content_t:s0->system_u:object
_r:admin_home_t:s0
restorecon reset /root/ktdir/file2 context system_u:object_r:public_content_t:s0->system_u:object
_r:admin_home_t:s0
[root@ktlinux ~]# ls -ldZ ktdir
drwxr-xr-x. root root system_u:object_r:admin_home_t:s0 ktdir
[root@ktlinux ~]# ls -lZ ktdir
-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
```

Note: For restoring the context of only the dir except its contents do not add "**R"** in the command.

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@ktlinux ~]# getenforce
Enforcing
[root@ktlinux ~]# setenforce 0
[root@ktlinux ~]# getenforce
Permissive
[root@ktlinux ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   permissive
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
[root@ktlinux ~]#
```

- To change the SELinux Mode back to Enforcing mode
  #setenforce 1
- Verify the change

```
[root@ktlinux ~]# getenforce
Permissive
[root@ktlinux ~]# setenforce 1
[root@ktlinux ~]# getenforce
Enforcing
[root@ktlinux ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 24
Policy from config file:        targeted
[root@ktlinux ~]#
```

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@ktlinux ~]# getenforce
Enforcing
[root@ktlinux ~]# 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
  #init 6 (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@ktlinux ~]# getenforce
Disabled
[root@ktlinux ~]# sestatus
SELinux status:                 disabled
[root@ktlinux ~]#
```

To Enable it back the procedure is exactly same as above, instead of **SELINUX=disabled** change it to **SELINUX=enforcing** or p**ermissive.** Don't forget to restart the system, unless the system is rebooted the changes will not take effect.

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@ktlinux ~]# getenforce
Enforcing
[root@ktlinux ~]# 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@ktlinux ~]#
```

- 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@ktlinux ~]# setsebool allow_ftpd_anon_write on
[root@ktlinux ~]# 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@ktlinux ~]#
```

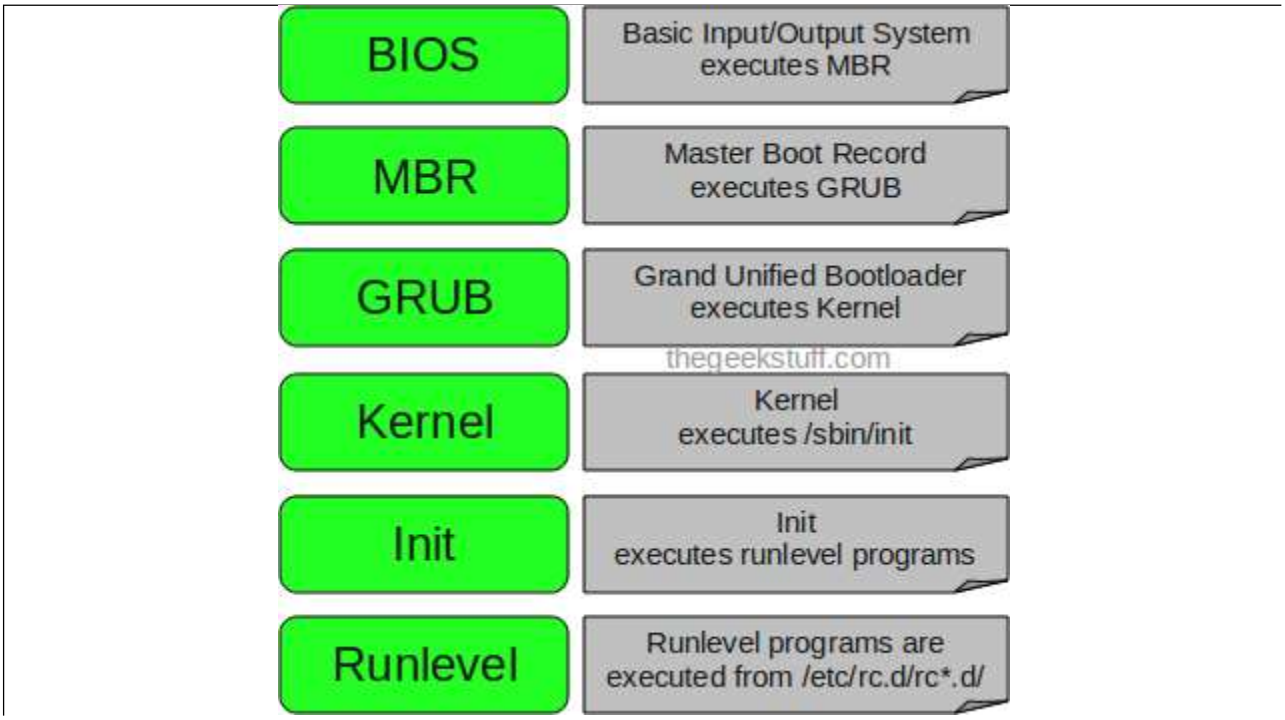
Hope you got the SELinux concept cleared by this time. Keep working on it

BOOTING PROCEDURE AND KERNEL PARAMETER

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 Linux boot process.



1. BIOS

- BIOS stands for Basic Input/Output System
- 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 of 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 loads and executes the MBR boot loader.

2. MBR

- MBR stands for Master Boot Record.
- 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 (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,1)
#          kernel /vmlinuz-version ro root=/dev/mapper/vg_ktadm-rootlv
#          initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,1)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux (2.6.32-131.0.15.el6.x86_64)
        root (hd0,1)
        kernel /vmlinuz-2.6.32-131.0.15.el6.x86_64 ro root=/dev/mapper/vg_ktad
rootlv rd_LVM_LV=vg_ktadm/rootlv rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
YSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us rhgb quiet
        initrd /initramfs-2.6.32-131.0.15.el6.x86_64.img
~
```

- As you notice from the above info, it contains kernel and initrd image.
- So, in simple terms GRUB just loads and executes Kernel and initrd images.

4. Kernel

- Mounts the root file system as specified in the "root=" in grub.conf
- Kernel executes the /sbin/init program

Since init was 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.

- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

1. Looks at the /etc/inittab file to decide the Linux run level.
2. Following are the available run levels
    - 0 – halt
    - 1 – Single user mode
    - 2 – Multiuser, without NFS
    - 3 – Full multiuser mode
    - 4 – unused
    - 5 – X11
    - 6 – reboot
3. Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
4. Execute 'grep initdefault /etc/inittab' on your system to identify the default run level
5. If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
6. Typically you would set the default run level to either 3 or 5.

6. Runlevel programs

- When the Linux system is booting up, you might see various services getting started. For example, it might say "starting sendmail …. OK". Those are the run level programs, executed from the run level directory as defined by your run level.
- Depending on your default init level setting, the system will execute the programs from one of the following directories.
    - Run level 0 – /etc/rc.d/rc0.d/
    - Run level 1 – /etc/rc.d/rc1.d/
    - Run level 2 – /etc/rc.d/rc2.d/
    - Run level 3 – /etc/rc.d/rc3.d/
    - Run level 4 – /etc/rc.d/rc4.d/
    - Run level 5 – /etc/rc.d/rc5.d/
    - Run level 6 – /etc/rc.d/rc6.d/

- Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.

LAB WORK:-

To check the default run level in linux

- To see the default run level in linux the command is
  #who –r

```
[root@ktadm ~]# who -r
         run-level 5  2011-11-01 12:39
[root@ktadm ~]#
```

Changing the default run level to some other like 3

- To change the run level edit the **/etc/inittab** and make the following changes
  #vim /etc/inittab

```
# System initialization is started by /etc/init/rcS.conf
#
# Individual runlevels are started by /etc/init/rc.conf
#
# Ctrl-Alt-Delete is handled by /etc/init/control-alt-delete.conf
#
# Terminal gettys are handled by /etc/init/tty.conf and /etc/init/serial.conf,
# with configuration in /etc/sysconfig/init.
#
# For information on how to write upstart event handlers, or how
# upstart works, see init(5), init(8), and initctl(8).
#
# Default runlevel. The runlevels used are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
```

- Now reboot the system and check in which runlevel it is.
  #init 6

```
Red Hat Enterprise Linux Server release 6.1 (Santiago)
Kernel 2.6.32-131.0.15.el6.x86_64 on an x86_64

ktcl3 login: root
Password:
[root@ktcl3 ~]# who -r
         run-level 3  2011-11-01 17:21
[root@ktcl3 ~]# _
```

- To start the graphical interface when you are in runlevel **3**, use the following command
  #startx
- Change it back to runlevel **5** and reboot the system.

To see the details regarding the kernel installed

- To see the version of the kernel use
  #uname –r

```
[root@ktcl3 ~]# uname -r
2.6.32-131.0.15.el6.x86_64
[root@ktcl3 ~]#
```

- To see the same thing with more details use
  #uname –a

```
[root@ktcl3 ~]# uname -a
Linux ktcl3.kt.com 2.6.32-131.0.15.el6.x86_64 #1 SMP Tue May 10 15:42:40 EDT 2011 x86_64 x86_64 x86_64 GNU/Linux
[root@ktcl3 ~]#
```

Note: The same information can be seen in /boot/grub/grub.conf

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,1)
#          kernel /vmlinuz-version ro root=/dev/mapper/vg_ktadm-rootlv
#          initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,1)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux (2.6.32-131.0.15.el6.x86_64)
        root (hd0,1)
        kernel /vmlinuz-2.6.32-131.0.15.el6.x86_64 ro root=/dev/mapper/vg_ktadm-
rootlv rd_LVM_LV=vg_ktadm/rootlv rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 S
YSFONT=latarcyrheb-sun16 KEYBOARDTYPE=pc KEYTABLE=us rhgb quiet
        initrd /initramfs-2.6.32-131.0.15.el6.x86_64.img
~
```

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

```
[root@ktadm Desktop]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 6.1 (Santiago)
[root@ktadm Desktop]#
```

Recovering the lost password in RHEL 6

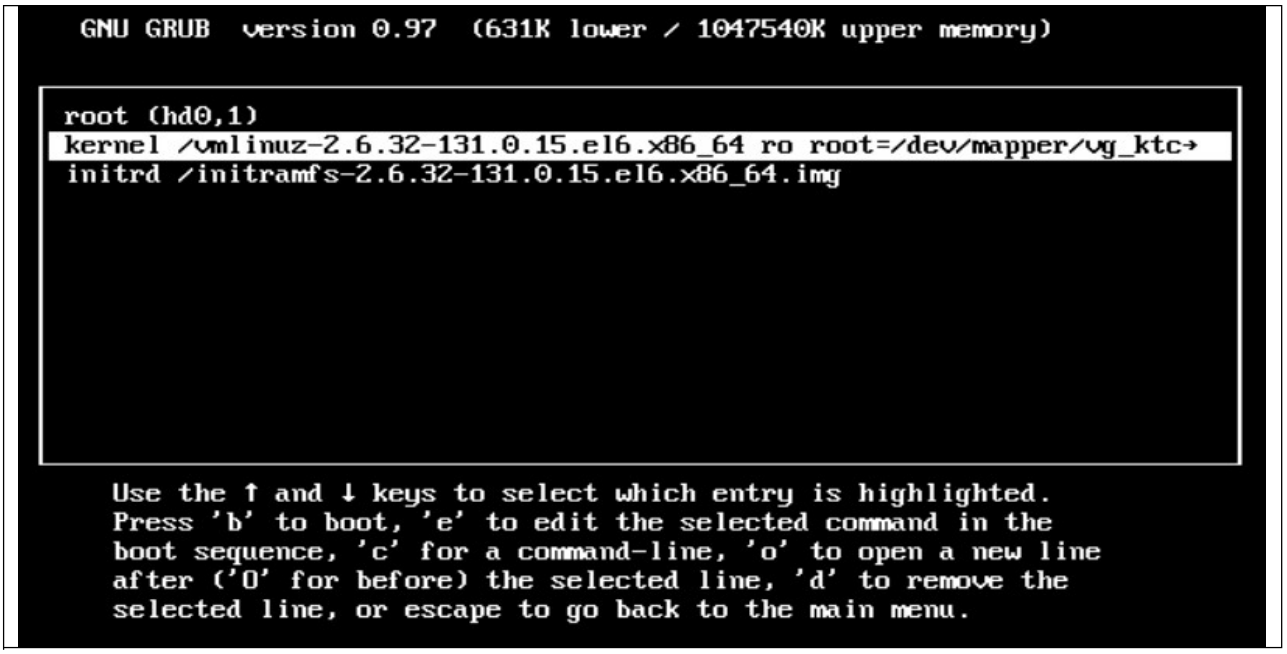
To recover the password the steps are

- **Distrub the normal boot by pressing any when RHEL 6 booting screen is displayed**

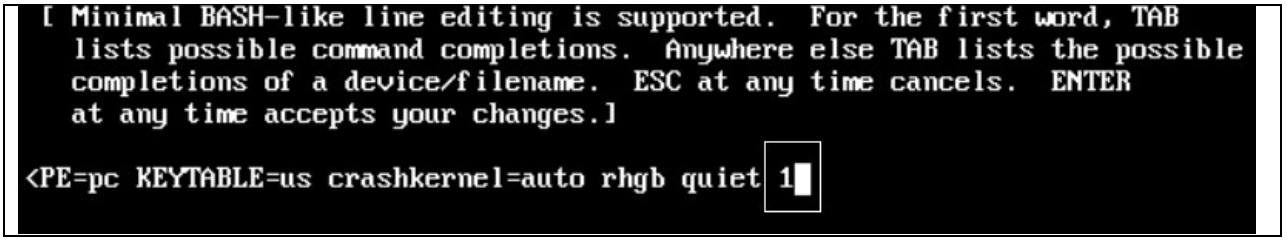
```
Press any key to enter the menu


Booting Red Hat Enterprise Linux (2.6.32-131.0.15.el6.x86_64) in 4 seconds.
```

- **You will be inside the menu like the following**

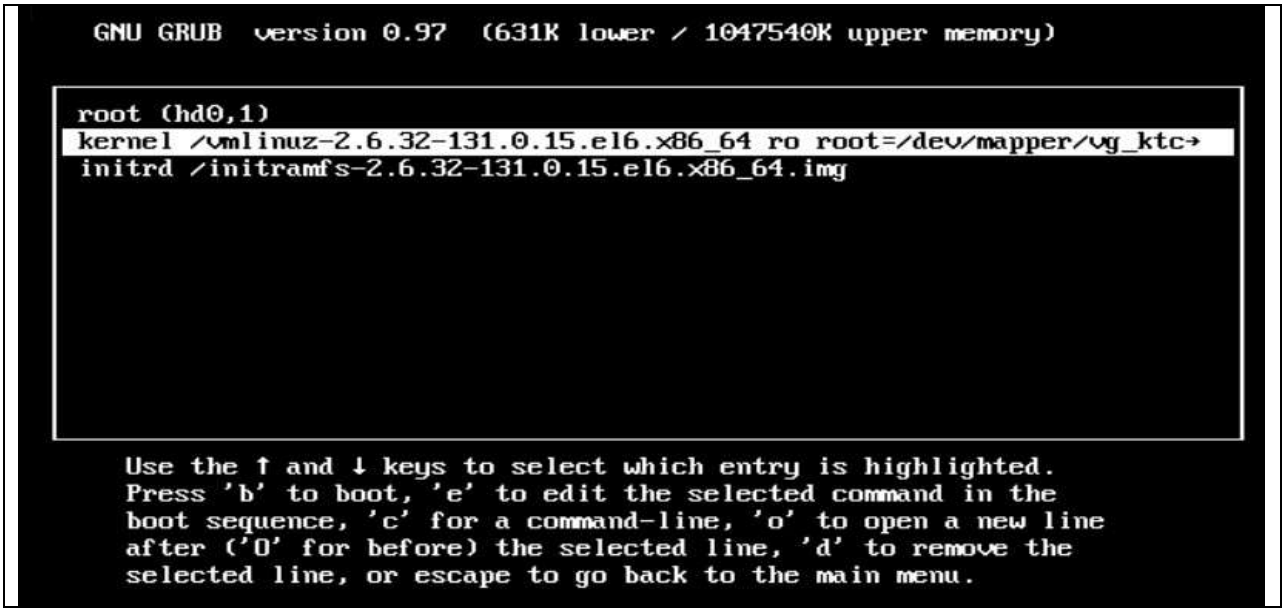
```
  GNU GRUB  version 0.97  (631K lower / 1047540K upper memory)

 Red Hat Enterprise Linux (2.6.32-131.0.15.el6.x86_64)




    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press enter to boot the selected OS, 'e' to edit the
    commands before booting, 'a' to modify the kernel arguments
    before booting, or 'c' for a command-line.
```



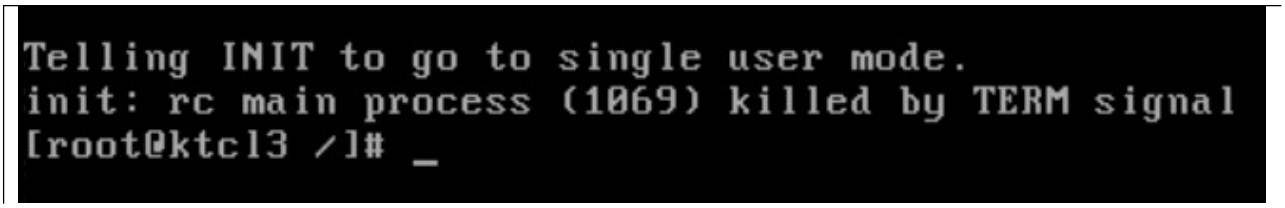
- **Move the cursor to 2nd line (line of Kernel) and press 'e' to edit the kernel parameter**



- **Type "1" after the line to boot in maintenance level and press enter to continue**



- **Now, type "b" to boot it in single user mode. Then you will be in single user mode**

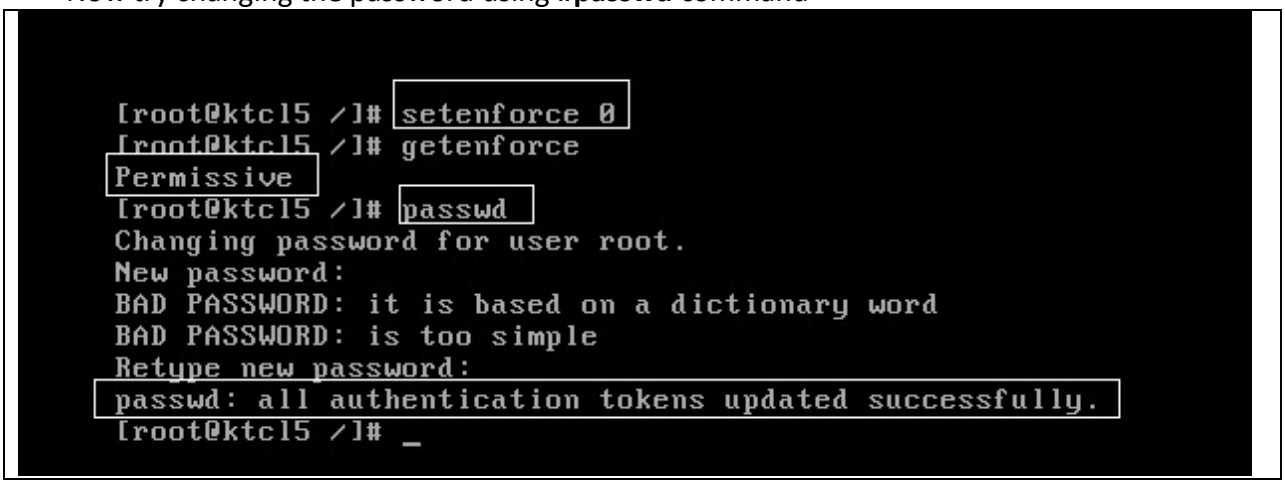


- **Now without being prompted for password you will be logged in the single user mode**

- To change the password use command **#passwd,** but as you can see it will not work because of SELinux.
- Check the **SELinux** mode by using **#getenforce** command

- As we can see that **SELinux** is in **Enforcing** mode, change it to **Permissive** and then try changing the password

- Change the **SELinux** Mode to **Permissive, using**
  #setenforce 0
- Now try changing the password using **#passwd** command



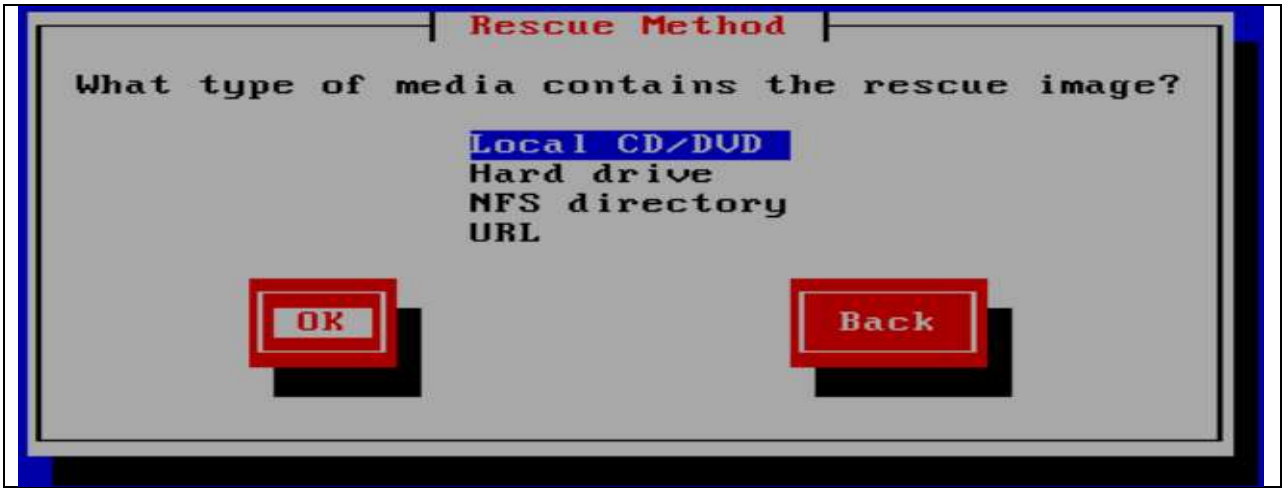
- Okay, Now we are successfully changed the password, now just type exit or reboot, to reboot the system and try the new password for root.

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 6 DVD**.
- To recover the grub the steps are:
  - Insert the **RHEL 6 DVD** and make the system boot from **CD/DVD**
  - Boot the system in **Rescue installed system** Mode.

- **Select the language with which you want to continue and move cursor on OK, press Enter.**

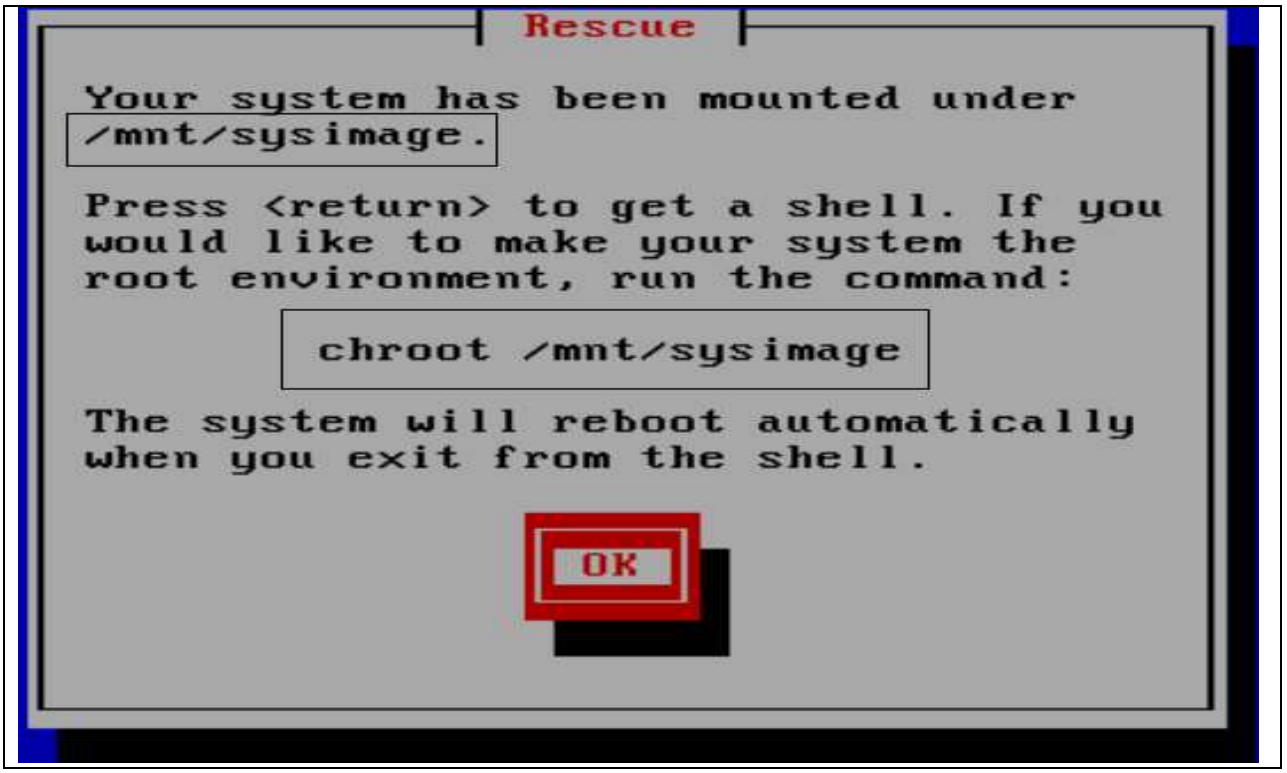
- **Select the keyboard Type and move cursor to OK and press Enter to continue.**



- **Select Local CD/DVD to make the sytem boot from it. As the system's bootloader is corrupt.**

- **Move cursor to NO to ignore the networking and also to continue**

- **Move the curor to Continue tab, to mount the 'root' from CD/DVD. Press Enter**



- **Observe from above pic, that now your system has been mounted on /mnt/sysimage. It means where our system root is residing**
- **Move the cursor to OK and press Enter to continue.**

- **Press Enter to continue.**

- **Select 'shell start shell' and move cursor to OK to start the shell**
- **You can observe that a shell prompt is displayed**

```
Starting shell...
bash-4.1# chroot /mnt/sysimage/
sh-4.1# _
```

- **Change the DVD root to system root by using following command**
  #chroot /mnt/sysimage

```
Disk /dev/vda: 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: 0x000150a1

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1               1        3188    25600000   8e  Linux LVM
/dev/vda2   *        3188        3213      204800   83  Linux
/dev/vda3            3213        3474     2097152   82  Linux swap
```

- **Check that on which partition does your '/boot' is using #fdisk –l or parted –l command**

```
sh-4.1# grub-install /dev/vda2
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

# this device map was generated by anaconda
(hd0)     /dev/vda
```

- **Install the grub on the /boot device i.e. /dev/vda2 by using following command**
  #grub-install <device name>
  #grub-install /dev/vda2
- **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**
- **Againg type "exit" or "reboot" to reboot the system.**

Listing out the Modules (Drivers) and blacklisting the USB and CD-ROM Driver:

- The drivers in Linux are known as Modules or Kernel Modules.
- These modules are assigned by kernel basing upon the hardware.
- Hardware can only be communicated and can work efficiently when the proper module is loaded in the kernel.

To find all the kernel Modules

- All the kernel modules will be residing in **/etc/lib/modules** directory
- Navigate to the directory **/etc/lib/modules** and check it with **uname –r**

```
[root@ktadm ~]# cd /lib/modules/
[root@ktadm modules]# ls
2.6.32-131.0.15.el6.x86_64
[root@ktadm modules]# uname -r
2.6.32-131.0.15.el6.x86_64
[root@ktadm modules]#
```

To search all the kernel modules in the system using find command

- All the kernel modules in the system will be ending with **.ko** extension, so let's search it using find command
  #find / -name *.ko

```
[root@ktadm modules]# find / -name *.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/libcrc32c.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/raid6/raid6_pq.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/crc-t10dif.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/zlib_deflate/zlib_deflate.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/reed_solomon/reed_solomon.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/crc-ccitt.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/lzo/lzo_compress.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/lzo/lzo_decompress.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/lib/ts_fsm.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/mm/hwpoison-inject.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/sound/drivers/snd-aloop.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/sound/drivers/snd-dummy.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/sound/drivers/snd-mtpav.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/sound/drivers/vx/snd-vx-lib.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/sound/drivers/snd-virmidi.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/sound/drivers/pcsp/snd-pcsp.ko
/lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/sound/drivers/mpu401/snd-mpu401-
```

Note:- Observe that all the modules listed are in /lib/modules only. All modules may be supported or currently loaded modules.

To list all the currently loaded modules

- **#lsmod**

```
[root@ktadm ~]# lsmod
Module                  Size  Used by
vfat                   10646  0
fat                    55054  1 vfat
usb_storage            49418  0
fuse                   66138  2
ip6table_filter         2855  0
ip6_tables             19424  1 ip6table_filter
ebtable_nat             1975  0
ebtables               18101  1 ebtable_nat
ipt_MASQUERADE          2400  3
iptable_nat             6124  1
nf_nat                 22788  2 ipt_MASQUERADE,iptable_nat
nf_conntrack_ipv4       9440  4 iptable_nat,nf_nat
nf_defrag_ipv4          1449  1 nf_conntrack_ipv4
xt_state                1458  1
nf_conntrack           79643  5 ipt_MASQUERADE,iptable_nat,nf_nat,nf_conntrack_i
pv4,xt_state
ipt_REJECT              2349  2
xt_CHECKSUM             1269  1
iptable_mangle          3283  1
iptable_filter          2759  1
ip_tables              17765  3 iptable_nat,iptable_mangle,iptable_filter
autofs4                27683  3
```

To check whether a particular module is loaded or not

- To see the particular module use
  #lsmod |grep –i module name
  #lsmod |grep –i fat
  #lsmod |grep –i cdrom

```
[root@ktadm ~]# lsmod |grep -i fat
vfat                    10646  0
fat                     55054  1 vfat
[root@ktadm ~]# lsmod |grep -i cdrom
cdrom                   39769  1 sr_mod
[root@ktadm ~]#
```

To remove the loaded module

- There might be a situation where your module is not working properly, in that case we need to remove the module and reinstall it. Let's see how to remove a module first
- From previous task we know that v**fat** module is installed let's remove it
  #modprobe –r < mod name>
  #modprobe –r vfat

```
[root@ktadm ~]# modprobe -r vfat
[root@ktadm ~]# lsmod |grep -i fat
[root@ktadm ~]# lsmod |grep -i vfat
```

To install/re-install a module

- To install a module use the following command
  #modprobe <mod name>
  #modprobe fat

```
[root@ktadm ~]# lsmod |grep -i fat
[root@ktadm ~]# modprobe vfat
[root@ktadm ~]# lsmod |grep -i fat
vfat                   10646  0
fat                    55054  1 vfat
```

To see the information about the module

- To see the information about a module the syntax is
  #modinfo < mod name>
  #modinfo cdrom

```
[root@ktadm ~]# modinfo cdrom
filename:       /lib/modules/2.6.32-131.0.15.el6.x86_64/kernel/drivers/cdrom/cdr
om.ko
license:        GPL
srcversion:     EA46535D273499C0A0C54A3
depends:
vermagic:       2.6.32-131.0.15.el6.x86_64 SMP mod_unload modversions
parm:           debug:bool
parm:           autoclose:bool
parm:           autoeject:bool
parm:           lockdoor:bool
parm:           check_media_type:bool
parm:           mrw_format_restart:bool
[root@ktadm ~]#
```

Disabling or Blacklisting a USB/CD-ROM driver

- **To disable a USB/CD-ROM drive driver, first check whether a driver is loaded or not**
- **#lsmod |grep –i usb**

```
[root@ktlinux ~]# lsmod |grep -i usb
[root@ktlinux ~]#
```

- **If it is not loaded, connect the USB drive to the system and wait for it to get loaded, then check it again whether the module is loaded or not**
  #lsmod |grep –i usb

```
[root@ktlinux ~]# lsmod |grep -i usb
usb_storage            39114  1
```

- **Also check where it is mounted**
  #mount

```
/dev/sdb1 on /media/E817-24C0 type vfat (rw,nosuid,nodev,uhelper=udisks,uid=0,gi
d=0,shortname=mixed,dmask=0077,utf8=1,flush)
```

- **You can navigate through /media/E817….. and verify whether it is correct device or not.**
- **Now as we know the module name just remove the module**
  #modprobe –r usb_storage

```
[root@ktlinux ~]# modprobe -r usb_storage
FATAL: Module usb_storage is in use.
```

The error showing above is because the drive is mounted, unmount it and try removing module again

```
[root@ktlinux ~]# umount /media/E817-24C0/
[root@ktlinux ~]# lsmod |grep -i usb
usb_storage            39114  0
[root@ktlinux ~]# modprobe -r usb_storage
[root@ktlinux ~]# lsmod |grep -i usb
[root@ktlinux ~]#
```

- **As we have successfully removed the module, now place the module name in /etc/modprobe.d/blacklist.conf file so that it may not be loaded in future.**
  #vim /etc/modprobe.d/blacklist.conf

```
# Listing a module here prevents the hotplug scripts from loading it.
# Usually that'd be so that some other driver will bind it instead,
# no matter which driver happens to get probed first.  Sometimes user
# mode tools can also control driver binding.
#
# Syntax: see modprobe.conf(5).
#
#USB
blacklist usb_storage

# watchdog drivers
blacklist i8xx_tco
```

Now save the file and quit the vim editor

- **Now again try connecting the drive and check whether the USB is loading or not**

```
[root@ktlinux ~]# lsmod |grep -i usb
[root@ktlinux ~]#
```

<u>Note</u>: - The procedure for black listing CD-ROM is exactly same. Try it yourselves

To remove the USB/CD-ROM from black list

- Remove the entry from **/etc/modprobe.d/blacklist.conf**
- Connect the drive and install the module for USB and check whether it is mounting or not

```
[root@ktlinux ~]# lsmod |grep -i usb
[root@ktlinux ~]# modprobe usb_storage
[root@ktlinux ~]# mount
/dev/sdb1 on /media/E817-24C0 type vfat (rw,nosuid,nodev,uhelper=udisks,uid=0,gi
d=0,shortname=mixed,dmask=0077,utf8=1,flush)
[root@ktlinux ~]#
```

JOB AUTOMATION

Automation with cron and at

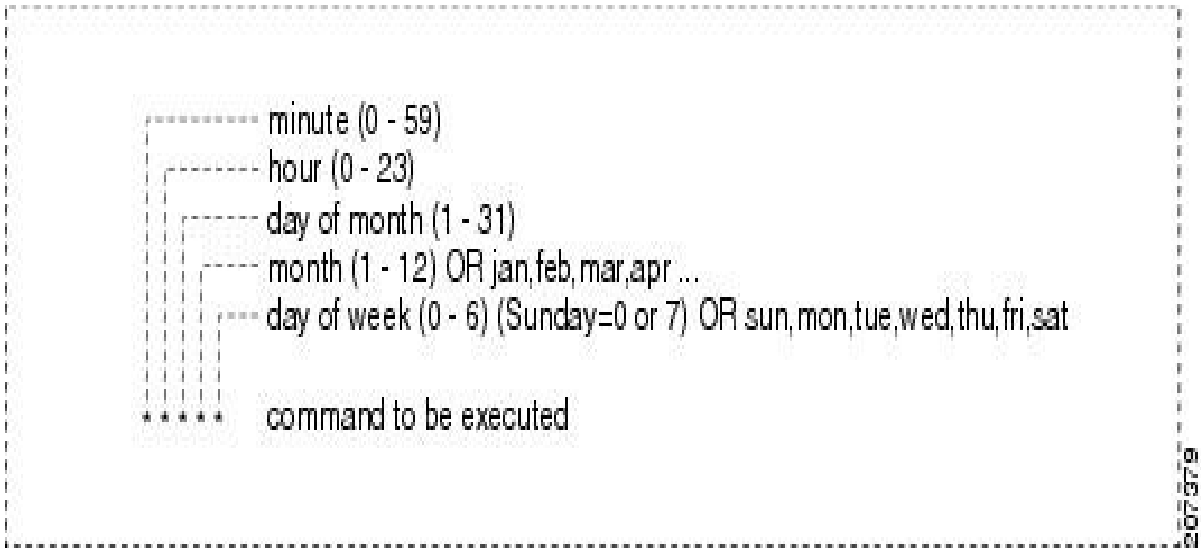
- In any operating system, it is possible to create jobs that you want to reoccur. This process, known as ***job scheduling***, is usually done based on user-defined jobs. For Red Hat or any other Linux, this process is handled by the cron service or a daemon called **crond**, which can be used to schedule 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 stores all scheduled jobs
- **/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)
- **/etc/at.deny** same as cron.deny for restricting at jobs
- **/etc/at.allow** same as cron.allow for allowing user to use at jobs.

Crontab format

- To assign a job in the Crontab file the format used is the following



Options	Explanation
*	**Is treated as a wild card. Meaning any possible value.**
*/5	**Is treated as ever 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:-

CRON JOBS:

To check the assigned cron jobs of currently logged in user

- To check the cron jobs the command is
  #crontab –l

```
[root@ktlinux ~]# crontab -l
no crontab for root
[root@ktlinux ~]#
```

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 ktuser

```
[root@ktlinux ~]# crontab -l -u ktuser
no crontab for ktuser
[root@ktlinux ~]# crontab -lu ktuser
no crontab for ktuser
[root@ktlinux ~]#
```

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@ktlinux ~]# tty
/dev/pts/1
[root@ktlinux ~]#
```

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@ktlinux ~]# crontab -e
crontab: installing new crontab
```

Note: where * means every possible value.

- Restart the cron services
  #service crond restart

```
[root@ktlinux ~]# service crond restart
Stopping crond:                                            [  OK  ]
Starting crond:                                            [  OK  ]
[root@ktlinux ~]#
```

- Wait for a minute and check whether time is displaying or not. Every min time will be displayed as below.

```
[root@ktlinux ~]# 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 "ktdir" 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/ktdir
~
:wq!
[root@ktlinux ~]# crontab -e
crontab: installing new crontab
[root@ktlinux ~]#
```

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 a 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@ktlinux ~]# ls
anaconda-ks.cfg  Desktop    Downloads    install.log.syslog
bkpscript.sh     Documents  install.log  ktdir
[root@ktlinux ~]# pwd
/root
[root@ktlinux ~]# ls -l bkpscript.sh
-rw-r--r--. 1 root root 0 Oct 13 15:47 bkpscript.sh
[root@ktlinux ~]# chmod a+x bkpscript.sh
[root@ktlinux ~]# ls -l bkpscript.sh
-rwxr-xr-x. 1 root root 0 Oct 13 15:47 bkpscript.sh
[root@ktlinux ~]#
```

- Apply the job in **crontab** and restart the service
  #crontab –l

```
30 1 22 10 0 mkdir /root/ktdir
30 12 * * 6  /root/bkpscript.sh
~
:wq!
[root@ktlinux ~]# crontab -e
crontab: installing new crontab
[root@ktlinux ~]# !ser
service crond restart
Stopping crond:                                            [  OK  ]
Starting crond:                                            [  OK  ]
```

Note: !ser is the command to restart the last restarted service

Schedule a job so that a user "ktuser" should get a mail regarding meeting on 24th, 29th and 31st 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 ktuser

```
25 14  27,29,31  10  *  echo "Meeting at 3:00 PM Today"
~
:wq!
[root@ktlinux ~]# crontab -e -u ktuser
no crontab for ktuser - using an empty one
crontab: installing new crontab
[root@ktlinux ~]# !ser
service crond restart
Stopping crond:                                            [  OK  ]
Starting crond:                                            [  OK  ]
[root@ktlinux ~]#
```

Schedule a job so that a user "ktuser" should get the mail from 15th to 20th and 25th to 30st 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 ktuser

```
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!
[root@ktlinux ~]# crontab -e -u ktuser
crontab: installing new crontab
[root@ktlinux ~]# service crond restart
Stopping crond:                                            [  OK  ]
Starting crond:                                            [  OK  ]
```

- There are still various method you can schedule the cron jobs, Do some **R&D** on it to find out more.

Restrict users "ktuser" "amit" "vivek" 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

```
ktuser
amit
vivek
~
~
~
:wq!
```

- Now login as one of those users and try to use crontab.

```
[root@ktlinux ~]# vim /etc/cron.deny
[root@ktlinux ~]# su - ktuser
[ktuser@ktlinux ~]$ crontab -l
You (ktuser) are not allowed to use this program (crontab)
See crontab(1) for more information
[ktuser@ktlinux ~]$ crontab -e
You (ktuser) are not allowed to use this program (crontab)
See crontab(1) for more information
[ktuser@ktlinux ~]$
```

- If again want to allow them to use cron job facilities just 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@ktlinux /]# vim /etc/cron.allow
[root@ktlinux /]# rm -f /etc/cron.deny
[root@ktlinux /]# su - vivek
[vivek@ktlinux ~]$ crontab  -l
You (vivek) are not allowed to use this program (crontab)
See crontab(1) for more information
[vivek@ktlinux ~]$ exit
logout
[root@ktlinux /]# su - musab
[musab@ktlinux ~]$ crontab -l
no crontab for musab
[musab@ktlinux ~]$ exit
logout
[root@ktlinux /]# su - rahul
[rahul@ktlinux ~]$ crontab -l
no crontab for rahul
[rahul@ktlinux ~]$ exit
logout
[root@ktlinux /]# su - ktuser
[ktuser@ktlinux ~]$ crontab -l
You (ktuser) are not allowed to use this program (crontab)
```

Note: To see man pages on cron job use #**man 4 crontabs** command

AT JOBS

- "at" is used to schedule the job for a particular time or interval, in other words it is used only for one time or only for one interval.

The disadvantages of at jobs are

- It can be modified like cron jobs
- It cannot be reused
- The content cannot to viewed in normal human readable format

Schedule at job to display current date on current console "now"

- To schedule above job using at first check the console
  #tty

```
[root@ktlinux ~]# tty
/dev/pts/1
[root@ktlinux ~]#
```

- The syntax to use at job for this task is
  #at <time>
  Task
  Ctrl+d to save it.
  #at now

```
[root@ktlinux /]# at now
at> date > /dev/pts/1
at> <EOT>
job 6 at 2011-10-13 18:43
[root@ktlinux /]# Thu Oct 13 18:43:21 IST 2011
```

Schedule at job to get a mail at 11:30 AM regarding meeting

- it is very similar to above task, use the following
  #at 11.30am

```
[root@ktlinux /]# at 10.30am
at> echo "Meeting today at 1 PM"
at> <EOT>
job 17 at 2011-10-14 10:30
[root@ktlinux /]#
```

Schedule at job to get a mail at 10.30 AM till three days from now for a meeting
#at 10.30am + 3days

```
[root@ktlinux /]# at 10.30am + 3days
at> echo "meeting today at 1 PM "
at> <EOT>
job 18 at 2011-10-16 10:30
[root@ktlinux /]#
```

Note: See man pages "man at" for more on at jobs and formats.

To check the list of at jobs

#at –l or **#atq**

```
[root@ktlinux /]# at -l
18      2011-10-16 10:30 a root
17      2011-10-14 10:30 a root
[root@ktlinux /]#
```

To check what is scheduled

#at –c < job id >

#at –c 18

Note: the output will not be in human readable format and also very lengthy.

```
[root@ktlinux /]# at -c 18
#!/bin/sh
# atrun uid=0 gid=0
# mail root 0
umask 22
ORBIT_SOCKETDIR=/tmp/orbit-root; export ORBIT_SOCKETDIR
HOSTNAME=ktlinux.kt.com; export HOSTNAME
IMSETTINGS_INTEGRATE_DESKTOP=yes; export IMSETTINGS_INTEGRATE_DESKTOP
SHELL=/bin/bash; export SHELL
```

To remove a job

- check the job id
- To remove a job the syntax is

  #atrm < job id >

  #atrm 17

  #atrm 18

```
[root@ktlinux /]# atq
18      2011-10-16 10:30 a root
17      2011-10-14 10:30 a root
[root@ktlinux /]# atrm 17
[root@ktlinux /]# atrm 18
[root@ktlinux /]# atq
[root@ktlinux /]#
```

<u>Restricting a user from using at jobs</u>

- To restrict a user from using at jobs it is exactly same like cron job
- Add user names to **/etc/at.deny ,** it will work like exactly like **/etc/cron.deny**
- To allow only few out of many users remove **at.deny** like we did for **cron.deny** and create **/etc/at.allow** and add user names who are allowed to use at jobs in it, like **cron.allow**
- If both **at.allow** and **at.deny** exists, then **at.allow** will have higer priority.
- If neither at.allow nor at.deny exits, then only root can use at jobs.

All the above are few examples to use cron jobs and at jobs, do some constant R&D's to know more about it.

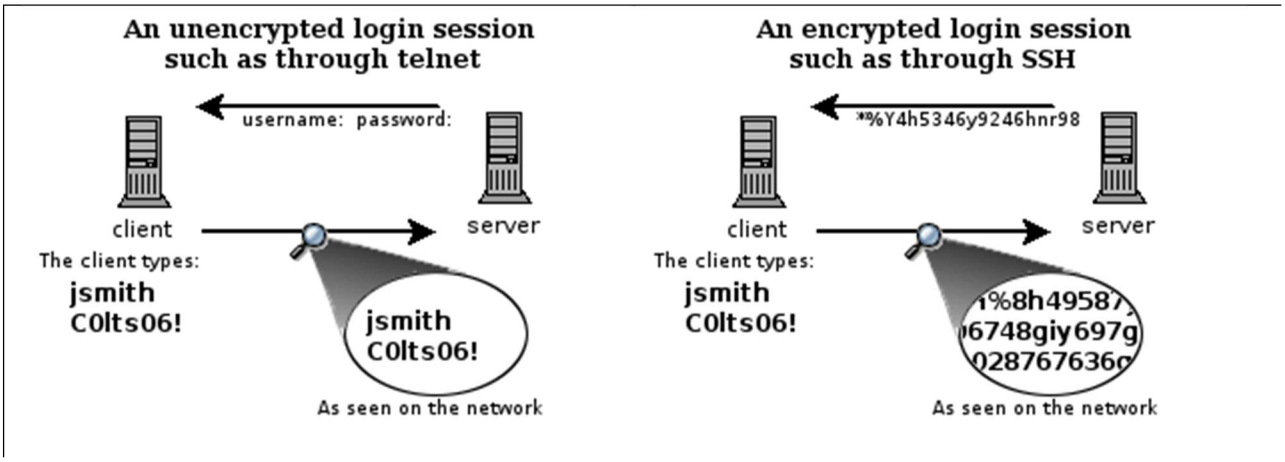
ADMINISTRATING REMOTE SYSTEM

- **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 C0lts06! 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@ktlinux .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@ktcl5 ~]# hostname
ktcl5.kt.com
[root@ktcl5 ~]# 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@ktcl5 ~]# hostname
ktcl5.kt.com
[root@ktcl5 ~]# exit
logout
Connection to 192.168.10.95 closed.
[root@ktlinux .ssh]# hostname
ktlinux.kt.com
[root@ktlinux .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
```

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.**

LAB WORK

Generating SSH key pair

- To generate the SSH key pair, the syntax is
  # ssh-keygen

```
[root@ktlinux ~]# 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@ktlinux ~]# 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 once per session. Enter your desired passphrase twice as shown on next page, 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]----+
|     .oo=        |
|     ..= +     . |
|      . * . . o  |
|       + . + o   |
|      * S . o    |
|     o = . o     |
|      + o . E    |
|       o         |
|                 |
+-----------------+
```

Okay now our keys are successfully generated, go to **/root/.ssh/** directory and check for the keys.

#cd /root/.ssh

```
[root@ktlinux ~]# cd /root/.ssh
[root@ktlinux .ssh]# ls
id_rsa  id_rsa.pub  known_hosts
[root@ktlinux .ssh]# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAv0sFIAZwmnB0KcBrRm71kze+RueIa0qx6HEPThdqdBpq0VG8IIBgKdLvrVZbm
PX1qmMnNDvJt7o8V9DSfHl2vwqEk8SoCuSQz53PwGJWSfmFYepkVF+0qpe3hsv2vFzFJmAoMInZZobkiwNH6Up9cQFPqMdpmP
J5cTe24dQLQuasFUQwg/IF15PK8o7dk0CUf+86Pxdb9XS3qGIZ7n6ABsIhE0MTQOF4uX/pnZNRWCZb7f8HFZ12x9Lsg2OV0SU
bbSELZmDfT0mLCP0ErUZ7oK8oTELcXl/sQ3Yddr5b09Caeb41OhKoNCUSiUOSIUmMrp3aSyaLIoSkTJ89IK35DQ== root@kt
linux.kt.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.

<u>Copying the public key on Client system</u>

- 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@ktlinux ~]# 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

Move to client system and check whether the key is copied properly or not

- To check the key navigate to **/root/.ssh/** directory and check for **authorzed_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@ktcl5 ~]# cd /root/.ssh/
[root@ktcl5 .ssh]# ls
authorized_keys
[root@ktcl5 .ssh]# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAv0sFIAZwmnB0KcBrRm71kze+RueIa0qx6HEPThdqdBpq0VG8IIBgKdLvrVZbm
PX1qmMnNDvJt7o8V9DSfHl2vwqEk8SoCuSQz53PwGJWSfmFYepkVF+0qpe3hsv2vFzFJmAoMInZZobkiwNH6Up9cQFPqMdpmP
J5cTe24dQLQuasFUQwg/IF15PK8o7dk0CUf+86Pxdb9XS3qGIZ7n6ABsIhE0MTQOF4uX/pnZNRWCZb7f8HFZ12x9Lsg2OV0SU
bbSELZmDfT0mLCP0ErUZ7oK8oTELcXl/sQ3Yddr5b09Caeb41OhKoNCUSiUOSIUmMrp3aSyaLIoSkTJ89IK35DQ== root@kt
linux.kt.com
[root@ktcl5 .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. But as we have assigned a passphrase it will ask us for it. Once you enter a passphrase it will last until you logged out of server's session. Let's see it practically.

  #ssh 192.168.10.95

  It will prompt for the passphrase as shown below, enter the passphrase and press Enter

```
[root@ktlinux ~]# ssh 192.168.10.95
```


```
Last login: Fri Oct 14 14:39:51 2011 from 192.168.10.98
[root@ktcl5 ~]# hostname
ktcl5.kt.com
[root@ktcl5 ~]#
```

- **Exit the client session and login again. Notice the change**
  While we logout the client session and re-login again it will not ask us for passphrase

```
[root@ktcl5 ~]# hostname
ktcl5.kt.com
[root@ktcl5 ~]# exit
logout
Connection to 192.168.10.95 closed.
[root@ktlinux ~]# hostname
ktlinux.kt.com
[root@ktlinux ~]# ssh 192.168.10.95
Last login: Fri Oct 14 14:43:41 2011 from 192.168.10.98
[root@ktcl5 ~]# hostname
ktcl5.kt.com
[root@ktcl5 ~]#
```

- **Logout of client session and completely logoff the server, login once again in server and connect the client using SSH.**
  As we log off completely from the server then login once again and try to connect the client, it will prompt for passphrase. Observe it below

```
[root@ktcl5 ~]# exit
logout
Connection to 192.168.10.95 closed.
[root@ktlinux ~]# exit
```

Note: if connected to the server via putty, above is complete log out just connect to the server once again as usual and try connecting to the client, but if connected graphically then go to **System -> logout <username>,** select it and click on **LOGOUT.** Login once more open terminal and reconnect to client and observe that it will prompt for passphrase.

```
[root@ktlinux ~]# ssh 192.168.10.95
```

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 **ktfile** in "**/"** directory, in the server **ktlinux.kt.com** who's IP is **192.168.10.98,** and we need to copy the same in **other** server's i.e. **ktcl5.kt.com** with an IP **192.168.10.95, /root directory. Then,**
- The syntax for SCP a file from source location.
  #scp <file name > <remote hosts IP >:/<location to copy the file >
  #scp /ktfile 192.168.10.95:/root/

```
[root@ktlinux ~]# hostname
ktlinux.kt.com
[root@ktlinux ~]# cat /ktfile
Welcome to Kernel Technologies
[root@ktlinux ~]# scp /ktfile 192.168.10.95:/root/
ktfile                                                    100%   31
```

- Now log in to destination system and check whether if the file is there.

```
[root@ktlinux ~]# ssh 192.168.10.95
Last login: Fri Oct 14 15:29:23 2011 from 192.168.10.98
[root@ktcl5 ~]# cd /root/
[root@ktcl5 ~]# ls
anaconda-ks.cfg  Documents  install.log         ktfile  Pictures
Desktop          Downloads  install.log.syslog  Music   Public
[root@ktcl5 ~]# cat ktfile
Welcome to Kernel Technologies
```

To copy a file using SCP from a remote machine being in destination's location

- Let's reverse the previous task, login to **ktcl5** machine whose **IP** is 192.168.10.95, and transfer a file from **ktlinux** machine whose **IP** is **192.168.10.98**
- Let's first remove the earlier copied file **ktfile,** 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@ktcl5 ~]# rm ktfile
rm: remove regular empty file `ktfile'? y
[root@ktcl5 ~]# scp 192.168.10.98:/ktfile /root/
ktfile                                                    100%   31
[root@ktcl5 ~]# cat ktfile
Welcome to Kernel 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 **ktdir** in"**/"** directory, in the server **ktlinux.kt.com** who's IP is **192.168.10.98,** and we need to copy the same in **other** server's i.e. **ktcl5.kt.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 /ktdir 192.168.10.95:/root/

```
[root@ktlinux ~]# tree /ktdir
/ktdir
├── ktfile1
├── ktfile2
├── ktfile3
├── ktfile4
└── ktfile5

0 directories, 5 files
[root@ktlinux ~]# scp -r /ktdir 192.168.10.95:/root/
ktfile1                                                   100%    0
ktfile4                                                   100%    0
ktfile5                                                   100%    0
ktfile2                                                   100%    0
ktfile3                                                   100%    0
```

To copy a directory using SCP from a remote machine being in destination's location

- Let's reverse the previous task, login to **ktcl5** machine who's **IP** is 192.168.10.95, and transfer a directory **ktdir** from **ktlinux** machine whose **IP** is **192.168.10.98**
- Let's first remove the earlier copied directory **ktdir,** 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:/ktdir /root/

```
[root@ktcl5 ~]# rm -rf /root/ktdir/
[root@ktcl5 ~]# scp -r 192.168.10.98:/ktdir /root/
ktfile1                                                   100%    0
ktfile4                                                   100%    0
ktfile5                                                   100%    0
ktfile2                                                   100%    0
ktfile3                                                   100%    0
[root@ktcl5 ~]#
```

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 suppose 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 **ktlinux** to **ktcl5** system, let' s use it for rsync.
- Update the directory with some files in **ktlinux** system

```
[root@ktlinux ~]# cd /ktdir
[root@ktlinux ktdir]# ls
ktfile1  ktfile2  ktfile3  ktfile4  ktfile5
[root@ktlinux ktdir]# touch ktfile{6..9}
[root@ktlinux ktdir]# ls
ktfile1  ktfile2  ktfile3  ktfile4  ktfile5  ktfile6  ktfile7  ktfile8  ktfile9
[root@ktlinux ktdir]#
```

- Check the content of same directory in **ktcl5**

```
[root@ktcl5 ~]# cd /root/ktdir
[root@ktcl5 ktdir]# ls
ktfile1  ktfile2  ktfile3  ktfile4  ktfile5
[root@ktcl5 ktdir]#
```

- Use rsync to sync the directory on **ktcl5** machine, with the one in **ktlinux** machine
- The syntax to rsycn a directory is

  #rsync <options> <encryption> <source dir> <destination IP>:/<location of destination dir>

  #rsync –rv -e ssh /ktdir 192.168.10.95:/root/

```
[root@ktlinux ~]# rsync -rv -e ssh /ktdir 192.168.10.95:/root/
sending incremental file list
ktdir/ktfile6
ktdir/ktfile7
ktdir/ktfile8
ktdir/ktfile9
```

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@ktlinux ~]# rsync -rv /ktdir 192.168.10.95:/root/
sending incremental file list
ktdir/ktfile1
ktdir/ktfile2
ktdir/ktfile3
ktdir/ktfile4
ktdir/ktfile5
ktdir/ktfile6
ktdir/ktfile7
ktdir/ktfile8
ktdir/ktfile9
```

- To compress the data and send it in archive mode use **-avz** instead of **-rv** in rsync

```
[root@ktlinux ~]# rsync -rv -e ssh /ktdir 192.168.10.95:/root/
sending incremental file list
ktdir/ktfile6
ktdir/ktfile7
ktdir/ktfile8
ktdir/ktfile9
```

Sync a file using rsync with ssh

- Let's check the file called ktfile1 on both **ktlinux** and **ktcl5** machines

```
[root@ktlinux ktdir]# cat ktfile1
Welcome to Kernel Technologies
```

```
[root@ktcl5 ktdir]# cat ktfile1
Welcome to Kernel Technologies
```

- Update the file **ktfile1** in **ktlinux**, sync it with rsync to **ktcl5** and check the file again.
- The syntax for syncing a file is
  #rsync –avz –e ssh <source file> <destination ip>:/<location of file >

```
[root@ktlinux ktdir]# vim ktfile1
[root@ktlinux ktdir]# cat ktfile1
Welcome to Kernel Technologies
        AMEERPET HYDERABAD
[root@ktlinux ktdir]# rsync -avz -e ssh /ktdir/ktfile1 192.168.10.95:/root/ktdir/
sending incremental file list
ktfile1

sent 123 bytes  received 37 bytes  106.67 bytes/sec
```

```
[root@ktlinux ktdir]# cat ktfile1
Welcome to Kernel Technologies
        AMEERPET HYDERABAD
```

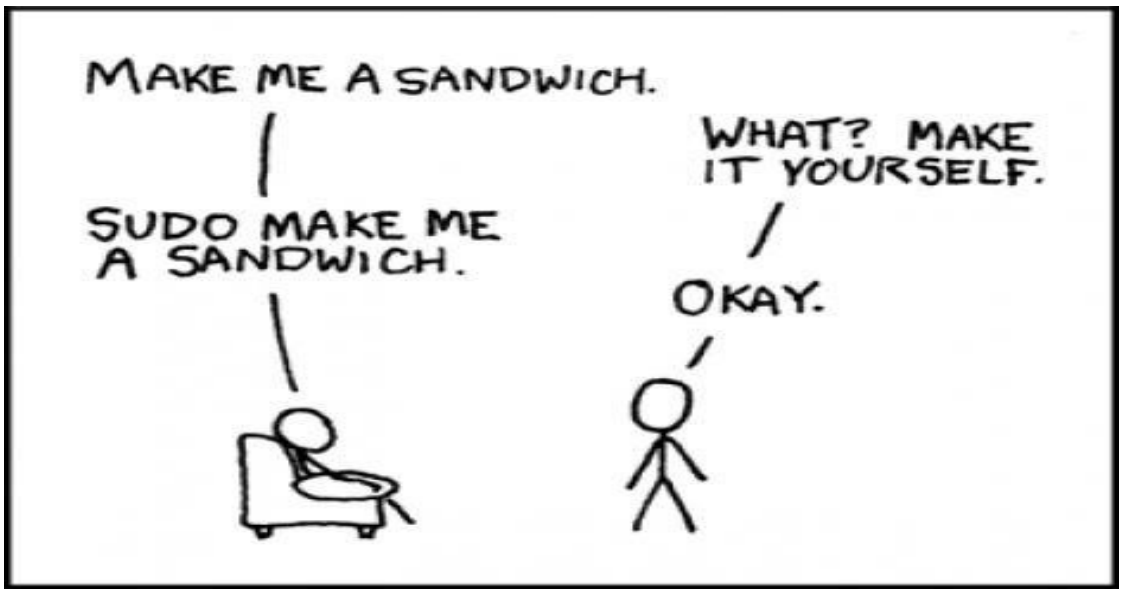
```
[root@ktcl5 ktdir]# cat ktfile1
Welcome to Kernel Technologies
        AMEERPET HYDERABAD
```

Like this you can use rsync in many ways to transfer the updated file or files/directory to other system.

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@ktcl5 ~]#
[root@ktcl5 ~]# visudo
>>> /etc/sudoers: syntax error near line 93 <<<
visudo: Warning: unused User_Alias KTADMIN
What now?
```

LAB WORK:-

Allow a user "ktuser" 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
ktuser  ALL=(ALL)       ALL
```

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

```
[root@ktcl5 ~]# su - ktuser
[ktuser@ktcl5 ~]$ fdisk -l
[ktuser@ktcl5 ~]$ fdisk /dev/sda

Unable to open /dev/sda
[ktuser@ktcl5 ~]$
```

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**

```
[ktuser@ktcl5 ~]$ sudo fdisk -l
[sudo] password for ktuser:

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

```
[ktuser@ktcl5 ~]$ 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 ktgroup, all root previleges.

- Let's first check the members of ktgroup and then apply root previleges.
  #tail /etc/gshadow

```
[root@ktcl5 ~]# tail /etc/gshadow
fuse:!::
stapdev:!::
stapusr:!::
gdm:!::
student:!!::
ktuser:!::
ktgroup:!::musab,rahul,sai
musab:!::
rahul:!::
sai:!::
[root@ktcl5 ~]#
```

- Okay as we know the users in ktgroup, let's assign it root previleges.
  #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
  %ktgroup      ALL=(ALL)       ALL
```

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

```
[root@ktcl5 ~]# su - musab
[musab@ktcl5 ~]$ 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 "ktuser2" 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
  %ktgroup      ALL=(ALL)       ALL
## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL
  ktuser2       ALL=(ALL)       NOPASSWD: ALL
```

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

```
[root@ktcl5 ~]# su - ktuser2
[ktuser2@ktcl5 ~]$ 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                  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 "ktuser" to run only two root commands.

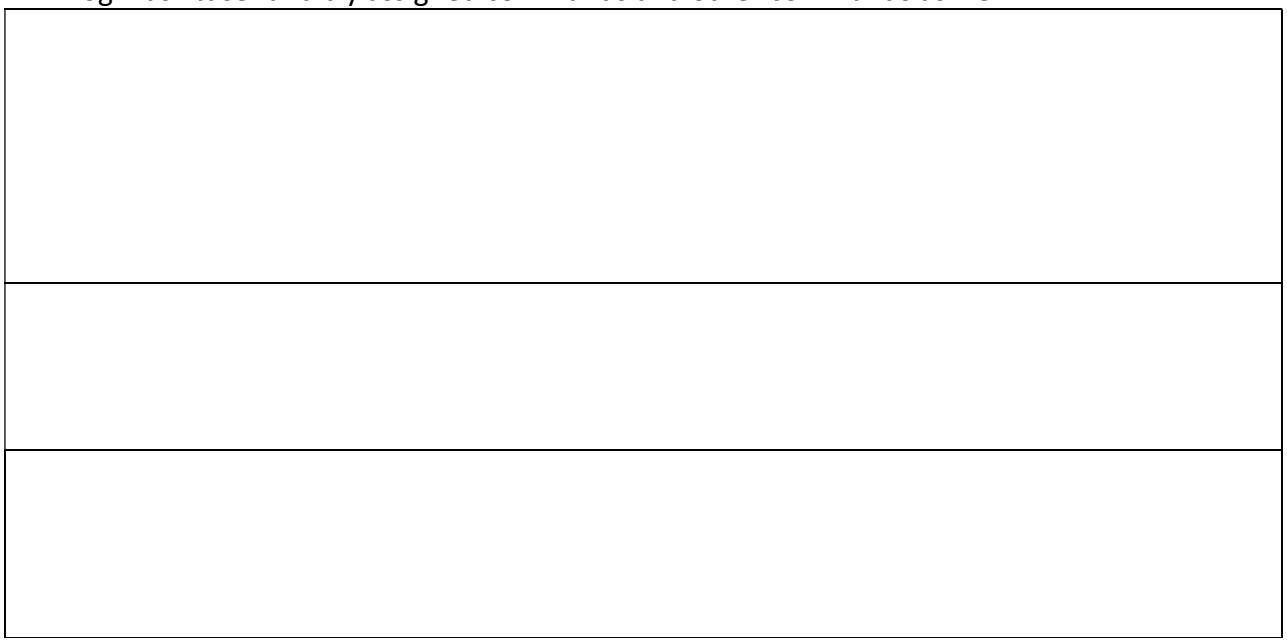
- This task is very simple, just modify the previous permissions assign to ktuser.
- Let's give ktuser to run only **#fdisk** and **#parted** command access.
- First check the complete path of those command by using following command
  #which fdisk
  #which parted

```
[root@ktcl5 ~]# which fdisk
/sbin/fdisk
[root@ktcl5 ~]# which parted
/sbin/parted
[root@ktcl5 ~]#
```

- Lets 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
ktuser  ALL=(ALL)       /sbin/fdisk, /sbin/parted
```

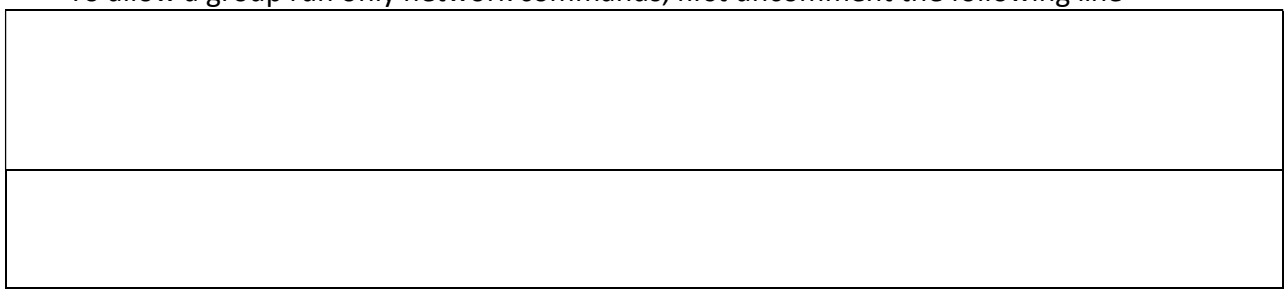
- Login as ktuser and try assigned commands and other commands as well



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

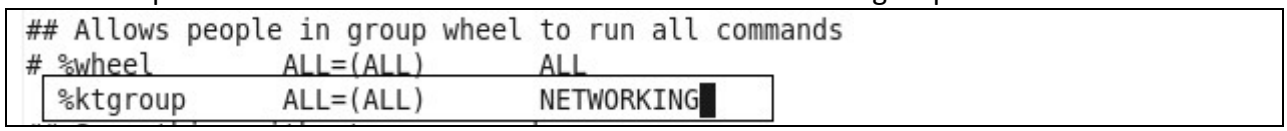
Allow a group " ktgroup" to run only network related commands as sudo user

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



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 **"NETWORKIG"** from the last field of ktgroup line.



NOTE: - NETWORKING is the name of the command alias where uncommented the line.

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

```
[root@ktcl5 ~]# su - rahul
[rahul@ktcl5 ~]$ sudo fdisk -l
[sudo] password for rahul:
Sorry, user rahul is not allowed to execute '/sbin/fdisk -l' as root on ktcl5.kt.com.
[rahul@ktcl5 ~]$ 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 ktgroup with network command.

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

```
[root@ktcl5 ~]# which service
/sbin/service
[root@ktcl5 ~]# which rpm
/bin/rpm
[root@ktcl5 ~]# which yum
/usr/bin/yum
[root@ktcl5 ~]#
```

- Okay, now let's create an alias for these commands and assign it to ktgroup
- **#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
  %ktgroup      ALL=(ALL)       NETWORKING, CUSTOM
```

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

```
[root@ktcl5 ~]# su - rahul
[rahul@ktcl5 ~]$ sudo rpm -q vsftpd
vsftpd-2.2.2-6.el6.i686
[rahul@ktcl5 ~]$ 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
erpriseLinux-201009221732.i386/6.0
```

Create a user alias and add some users from different groups and assign some root privileges

- This is very much similar to the previous task, instead of command alias we need to combine some user and give them some alias name which act as one user, but actually there are some users inside it. Then we can assign some privileges to them
- First we need available users and their groups.
  #tail /etc/passwd and **#tail /etc/gshadow.**

- From the above query, we can take **sai** from ktgroup, **student and ktuser** and can make an alias user. Search "**User Aliases",** under that create your own alias user as shown below.
  #visudo

```
## User Aliases
## These aren't often necessary, as you can use regular groups
## (ie, from files, LDAP, NIS, etc) in this file - just use %groupname
## rather than USERALIAS
# User_Alias ADMINS = jsmith, mikem
  User_Alias KTADMIN = sai, student, ktuser
```

- Now, let's assign some command to this alias user called **KTADMIN** and save the file.

```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
ktuser  ALL=(ALL)       /sbin/fdisk, /sbin/parted
KTADMIN ALL=(ALL)       CUSTOM, /sbin/umount
```

- Now login as one of those users and check the assigned commands for them.

```
[root@ktcl5 ~]# su - sai
[sai@ktcl5 ~]$ sudo route
[sudo] password for sai:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
192.168.10.0    *               255.255.255.0   U     0      0
link-local      *               255.255.0.0     U     1002   0
[sai@ktcl5 ~]$ sudo umount
Usage: umount -h | -V
       umount -a [-d] [-f] [-r] [-n] [-v] [-t vfstypes] [-O opts]
```

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

SOFTWARE MANAGEMENT

To manage the software in Linux, two utilities are used,

1. **RPM – REDHAT PACKAGE MANAGER**
2. **YUM – YELLOWDOG UPDATER MODIFIED**

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@ktlinux ~]# 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@ktlinux ~]# rpm -qa vsftpd
vsftpd-2.2.2-6.el6.i686
[root@ktlinux ~]# rpm -q vsftpd
vsftpd-2.2.2-6.el6.i686
[root@ktlinux ~]#
```

- 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@ktlinux ~]# 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@ktlinux ~]#
```

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@ktlinux ~]# cd /var/ftp/pub/rhel6/Packages/
[root@ktlinux 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.el6.i686.rpm

```
[root@ktlinux 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@ktlinux 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@ktlinux ~]# cd /var/ftp/pub/rhel6/Packages/
[root@ktlinux 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@ktlinux 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@ktlinux Packages]#
```

- To check whether it is installed or not
  #rpm –qa finger

```
[root@ktlinux Packages]# rpm -qa finger
finger-0.17-39.el6.i686
[root@ktlinux Packages]#
```

- Check the installed package by using it command, finger is used to check user's details.
  #finger <user name>
  #finger ktuser

```
[root@ktlinux Packages]# finger ktuser
Login: ktuser                           Name: ktuser
Directory: /home/ktuser                 Shell: /bin/bash
Never logged in.
No mail.
No Plan.
[root@ktlinux Packages]#
```

To remove a package or uninstall the package

- To remove a package the syntax is
  #rpm –e < package name>
  #rpm –e finger
  Verify it by **#rpm –q or rpm –qa command**

```
[root@ktlinux Packages]# rpm -e finger
[root@ktlinux Packages]# rpm -q finger
package finger is not installed
[root@ktlinux Packages]# rpm -qa finger
[root@ktlinux Packages]#
```

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@ktlinux ~]# cd /var/ftp/pub/rhel6/Packages/
[root@ktlinux 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@ktlinux 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@ktlinux ~]# 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@ktlinux ~]#
```

To check the package of a particular command

- To check the package of a particular command, first check the installed location of a command
  #which <command name>
  #which cat

```
[root@ktlinux ~]# which cat
/bin/cat
[root@ktlinux ~]#
```

- Now, use the following command,
  #rpm –qf <path of the command>
  #rpm –qf /bin/cat

```
[root@ktlinux ~]# rpm -qf /bin/cat
coreutils-8.4-9.el6.i686
[root@ktlinux ~]#
```

To install a package forcefully

- Before installing a package forcefully, first understand a situation where we need this force option.
- Let me 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
  #rpm –qf /bin/mount

```
[root@ktlinux ~]# which mount
/bin/mount
[root@ktlinux ~]# rpm -qf /bin/mount
util-linux-ng-2.17.2-6.el6.i686
[root@ktlinux ~]#
```

- 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 /bin/date /bin/mount

```
[root@ktlinux ~]# cp /bin/date /bin/mount
cp: overwrite `/bin/mount'? y
[root@ktlinux ~]#
```

- Now when you run mount command it will show date, that means it is corrupted.

```
[root@ktlinux ~]# mount
Sat Oct 15 03:25:34 IST 2011
[root@ktlinux ~]# date
Sat Oct 15 03:25:41 IST 2011
[root@ktlinux ~]#
```

- 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@ktlinux ~]# cd /var/ftp/pub/rhel6/Packages/
[root@ktlinux Packages]# rpm -ivh util-linux-ng-2.17.2-6.el6.i686.rpm
warning: util-linux-ng-2.17.2-6.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NO
KEY
Preparing...                ########################################### [100%]
        package util-linux-ng-2.17.2-6.el6.i686 is already installed
[root@ktlinux Packages]#
```

It says the package is already install, check by using mount command whether it is working fine.

```
[root@ktlinux Packages]# mount
Sat Oct 15 03:30:54 IST 2011
[root@ktlinux Packages]#
```

- Oops…!!! It isn't fixed yet, now in such to force the installation to be done, the syntax is

  #rpm –ivh <package name > - - force

  # rpm –ivh util-linux-ng 2.17.2-6.el6.i686 - -force

```
[root@ktlinux Packages]# rpm -ivh util-linux-ng-2.17.2-6.el6.i686.rpm --force
warning: util-linux-ng-2.17.2-6.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NO
KEY
Preparing...                ########################################### [100%]
   1:util-linux-ng          ########################################### [100%]
[root@ktlinux Packages]# mount
/dev/mapper/rootvg_ktlinux-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_ktlinux-LogVol01 on /home type ext4 (rw)
/dev/mapper/rootvg_ktlinux-LogVol04 on /opt type ext4 (rw)
/dev/mapper/rootvg_ktlinux-LogVol03 on /usr type ext4 (rw)
/dev/mapper/rootvg_ktlinux-LogVol02 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)
[root@ktlinux Packages]#
```

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@ktlinux 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@ktlinux ~]# 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@ktlinux ~]# vim /etc/vsftpd/vsftpd.conf
[root@ktlinux ~]# rpm -V vsftpd
S.5....T.  c /etc/vsftpd/vsftpd.conf
```

It is showing that on line 5 in config file, 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 Cantos 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**

LAB WORK:-

Configuring a YUM server and adding the info about it in at least one client

To configure a YUM server the steps are.

- **Make sure that vsftpd package is installed, if not install it.**
- **Copy entire RHEL6 DVD to "/var/ftp/pub/rhel6" directory, where rhel6 dir is to made by us only it is not default dir.**
- **Make a repo file as "kt.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@ktlinux ~]# rpm -qa vsftpd
vsftpd-2.2.2-6.el6.i686
[root@ktlinux ~]#
```

- 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
/dev/sr0 on /media/RHEL_6.0 i386 Disc 1 type iso9660
```

- As we know the mount point of dvd is **/media/RHEL_6**, move to its location and enter into **Packages** directory.

```
[root@ktlinux ~]# cd /media/RHEL_6.0\ i386\ Disc\ 1/
[root@ktlinux RHEL_6.0 i386 Disc 1]# cd Packages/
[root@ktlinux Packages]# ls |grep -i vsftpd
vsftpd-2.2.2-6.el6.i686.rpm
[root@ktlinux Packages]# rpm -ivh vsftpd-2.2.2-6.el6.i686.rpm
warning: vsftpd-2.2.2-6.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Preparing...                ########################################### [100%]
        package vsftpd-2.2.2-6.el6.i686 is already installed
[root@ktlinux Packages]#
```

As it is already installed, it is not being installed.

Copy entire RHEL6 DVD to "/var/ftp/pub/rhel6" directory, Where rhel6 dir is to be made by user only it is not a default dir

- First make an directory "rhel6" under **/var/ftp/pub**
  #mkdir /var/ftp/pub/rhel6

```
[root@ktlinux ~]# mkdir /var/ftp/pub/rhel6
[root@ktlinux ~]# cd /var/ftp/pub/
[root@ktlinux pub]# ls
rhel6
[root@ktlinux pub]#
```

- Now copy the RHEL6 DVD to /var/ftp/pub/rhel6 directory with its default permission
  #cp –rvfp /media/RHEL_6.0\I386\Disc\1/* /var/ftp/pub/rhel6

```
[root@ktcl5 ~]# cp -rvfp /media/RHEL_6.0\ i386\ Disc\ 1/* /var/ftp/pub/rhel6/
`/media/RHEL_6.0 i386 Disc 1/EULA' -> `/var/ftp/pub/rhel6/EULA'
`/media/RHEL_6.0 i386 Disc 1/GPL' -> `/var/ftp/pub/rhel6/GPL'
`/media/RHEL_6.0 i386 Disc 1/HighAvailability' -> `/var/ftp/pub/rhel6/HighAvailability'
`/media/RHEL_6.0 i386 Disc 1/HighAvailability/Packages' -> `/var/ftp/pub/rhel6/HighAvailability/P
ackages'
`/media/RHEL_6.0 i386 Disc 1/HighAvailability/TRANS.TBL' -> `/var/ftp/pub/rhel6/HighAvailability/
TRANS.TBL'
`/media/RHEL_6.0 i386 Disc 1/HighAvailability/repodata' -> `/var/ftp/pub/rhel6/HighAvailability/r
epodata'
`/media/RHEL_6.0 i386 Disc 1/HighAvailability/repodata/160a3cc3436b80ac6304f30c1330b0eaf299d2cd1a
6e06929151305498ef9ac0-other.sqlite.bz2' -> `/var/ftp/pub/rhel6/HighAvailability/repodata/160a3cc
3436b80ac6304f30c1330b0eaf299d2cd1a6e06929151305498ef9ac0-other.sqlite.bz2'
`/media/RHEL_6.0 i386 Disc 1/HighAvailability/repodata/6ea28d988fa906de8fe148bf9d15351ab59b34edb4
469ff5a05d26f6c3f665ec-other.xml.gz' -> `/var/ftp/pub/rhel6/HighAvailability/repodata/6ea28d988fa
906de8fe148bf9d15351ab59b34edb4469ff5a05d26f6c3f665ec-other.xml.gz'
`/media/RHEL_6.0 i386 Disc 1/HighAvailability/repodata/8e5b19b62dcb8bd3c5cafe6f785324175ac589f3f1
1217994d2a8ad677659306-primary.sqlite.bz2' -> `/var/ftp/pub/rhel6/HighAvailability/repodata/8e5b1
```

Note:- it will take around 5 minutes copy all the data, based on the DVD

- Check the directory after copying is finished.

```
[root@ktlinux ~]# cd /var/ftp/pub/rhel6/
[root@ktlinux rhel6]# ls
EULA                        RELEASE-NOTES-es-ES.html  RELEASE-NOTES-ru-RU.html
GPL                         RELEASE-NOTES-fr-FR.html  RELEASE-NOTES-si-LK.html
HighAvailability            RELEASE-NOTES-gu-IN.html  RELEASE-NOTES-ta-IN.html
images                      RELEASE-NOTES-hi-IN.html  RELEASE-NOTES-te-IN.html
isolinux                    RELEASE-NOTES-it-IT.html  RELEASE-NOTES-zh-CN.html
LoadBalancer                RELEASE-NOTES-ja-JP.html  RELEASE-NOTES-zh-TW.html
media.repo                  RELEASE-NOTES-kn-IN.html  repodata
Packages                    RELEASE-NOTES-ko-KR.html  ResilientStorage
README                      RELEASE-NOTES-ml-IN.html  RPM-GPG-KEY-redhat-beta
RELEASE-NOTES-as-IN.html    RELEASE-NOTES-mr-IN.html  RPM-GPG-KEY-redhat-release
RELEASE-NOTES-bn-IN.html    RELEASE-NOTES-or-IN.html  Server
RELEASE-NOTES-de-DE.html    RELEASE-NOTES-pa-IN.html  TRANS.TBL
RELEASE-NOTES-en-US.html    RELEASE-NOTES-pt-BR.html
```

Okay, then half of our configuration is completed.

Make a repo file as “kt.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.reops.d/kt.repo

```
[KTREPO]
name=Redhat.enterprise 6
baseurl=file:///var/ftp/pub/rhel6
enabled=1
gpgcheck=0
```

I guess there’s some explanation requires about the fields we have entered.

- **[KTREPO]** is the short name given to the repository.
- **name** is the complete name 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** With the gpgcheck option, all packages must be signed, and yum must be able to verify the signatures on packages from **red hat** . If gpgcheck=0, there will be no package signing by red hat and signature verification.

Clean the yum cache and check the package list using yum command

- To clear the cache use the following command

  #yum clean all

```
[root@ktlinux ~]# yum clean all
Loaded plugins: refresh-packagekit, rhnplugin
Cleaning up Everything
```

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 list (to list all the packages in repository)

```
[root@ktlinux ~]# yum list
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-RedHatEnterp
iseLinux-201009221732.i386/6.0
ConsoleKit-libs.i686                    0.4.1-3.el6                         @anaconda-RedHatEnterp
iseLinux-201009221732.i386/6.0
ConsoleKit-x11.i686                     0.4.1-3.el6                         @anaconda-RedHatEnterp
iseLinux-201009221732.i386/6.0
DeviceKit-power.i686                    014-1.el6                           @anaconda-RedHatEnterp
iseLinux-201009221732.i386/6.0
GConf2.i686                             2.28.0-6.el6                        @anaconda-RedHatEnterp
iseLinux-201009221732.i386/6.0
GConf2-gtk.i686                         2.28.0-6.el6                        @anaconda-RedHatEnterp
iseLinux-201009221732.i386/6.0
MAKEDEV.i686                            3.24-6.el6                          @anaconda-RedHatEnterp
```

If the above output is displayed then congratulation you have successfully configured the yum Server

Configure the yum client and check whether yum server is responding to it.

Configuring a yum client is very simple with just three steps.

- **Install ftp package , so that packages can be accessed from client**
- Make a repo file **/etc/yum.repo.d/ as "ktcl5.repo"**
- **Clean the cache and check whether yum server is responding or not**

Install ftp package, so that packages can be accessed from client

- Install the ftp package from rhel dvd in **Packages** directory.

```
[root@ktcl5 Packages]# cd /media/RHEL_6.0\ i386\ Disc\ 1/Packages/
[root@ktcl5 Packages]# ls |grep -i ftp
ftp-0.17-51.1.el6.i686.rpm
gvfs-obexftp-1.4.3-9.el6.i686.rpm
lftp-4.0.9-1.el6.i686.rpm
report-config-ftp-0.18-7.el6.i686.rpm
report-plugin-ftp-0.18-7.el6.i686.rpm
tftp-0.49-5.1.el6.i686.rpm
tftp-server-0.49-5.1.el6.i686.rpm
vsftpd-2.2.2-6.el6.i686.rpm
[root@ktcl5 Packages]# rpm -ivh ftp-0.17-51.1.el6.i686.rpm
warning: ftp-0.17-51.1.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Preparing...                ########################################### [100%]
   1:ftp                    ########################################### [100%]
[root@ktcl5 Packages]# rpm -q ftp
ftp-0.17-51.1.el6.i686
[root@ktcl5 Packages]#
```

Make a repo file /etc/yum.repo.d/ as "ktcl5.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/ktcl5.repo

```
[KTREPO]
name=redhat.enterprise6
baseurl=ftp://192.168.10.98/pub/rhel6
enabled=1
gpgcheck=0
```

Note:- baseurl =ftp://192.168.10.98/pub/rhel6 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 clients yum request.

  #yum list

```
[root@ktcl5 ~]# yum list
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
KTREPO                                                    | 3.7 kB     00:00
KTREPO/primary_db                                         | 2.3 MB     00:00
Installed Packages

ConsoleKit.i686                      0.4.1-3.el6                  @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
ConsoleKit-libs.i686                 0.4.1-3.el6                  @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
ConsoleKit-x11.i686                  0.4.1-3.el6                  @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
DeviceKit-power.i686                 014-1.el6                    @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
GConf2.i686                          2.28.0-6.el6                 @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
GConf2-gtk.i686                      2.28.0-6.el6                 @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
```

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.

- In case if yum list command is not listing the package then delete the following file in **/etc/yum.repos.d/** at client side only.

```
[root@ktcl5 ~]# cd /etc/yum.repos.d/
[root@ktcl5 yum.repos.d]# ls
ktcl5.repo  packagekit-media.repo
[root@ktcl5 yum.repos.d]# rm -f  packagekit-media.repo
```

- Clean the cache and list the packages, it will certainly solve the problem.

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 first command, second will also gives exactly same output. let us see the third command
  #yum list |more

```
[root@ktlinux ~]# 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
ConsoleKit-libs.i686                     0.4.1-3.el6                          @anaconda-RedHatEnterpr
NetworkManager.i686                      1:0.8.1-5.el6                        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
NetworkManager-glib.i686                 1:0.8.1-5.el6                        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
NetworkManager-gnome.i686                1:0.8.1-5.el6                        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
ORBit2.i686                              2.14.17-3.1.el6                      @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0
PackageKit.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

```
[root@ktlinux ~]# 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
ConsoleKit-libs.i686                     0.4.1-3.el6                          @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
ConsoleKit-x11.i686                      0.4.1-3.el6                          @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
DeviceKit-power.i686                     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>
  #yum list installed vsftpd

```
[root@ktlinux ~]# 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@ktlinux ~]#
```

To install a package using yum

- Installing a package using yum does not requires 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>
  #yum install finger* (where * means anything with name "finger")

```
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package finger.i686 0:0.17-39.el6 set to be updated
---> Package finger-server.i686 0:0.17-39.el6 set to be updated
--> Processing Dependency: xinetd for package: finger-server-0.17-39.el6.i686
--> Running transaction check
---> Package xinetd.i686 2:2.3.14-29.el6 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch            Version                 Repository      Size
================================================================================
Installing:
 finger                 i686            0.17-39.el6             KTREPO          22 k
 finger-server          i686            0.17-39.el6             KTREPO          16 k
Installing for dependencies:
 xinetd                 i686            2:2.3.14-29.el6         KTREPO         121 k

Transaction Summary
================================================================================
Install       3 Package(s)
Upgrade       0 Package(s)

Total download size: 158 k
Installed size: 294 k
Is this ok [y/N]: y
```

It will prompt you for y/n 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
  #yum install finger* -y

```
Installing for dependencies:
 xinetd                 i686            2:2.3.14-29.el6         KTREPO         121 k

Transaction Summary
================================================================================
Install       3 Package(s)
Upgrade       0 Package(s)

Total download size: 158 k
Installed size: 294 k
Downloading Packages:
--------------------------------------------------------------------------------
Total                                           437 kB/s | 158 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing     : 2:xinetd-2.3.14-29.el6.i686                                1/3
  Installing     : finger-server-0.17-39.el6.i686                             2/3
  Installing     : finger-0.17-39.el6.i686                                    3/3

Installed:
  finger.i686 0:0.17-39.el6                 finger-server.i686 0:0.17-39.el6

Dependency Installed:
  xinetd.i686 2:2.3.14-29.el6

Complete!
```

To remove the package with yum command

- To remove the package using yum command, the syntax is
  #yum remove <package name>
  #yum remove finger -y

```
================================================================================================
 Package              Arch              Version                   Repository             Size
================================================================================================
Removing:
 finger               i686              0.17-39.el6               @KTREPO                25 k

Transaction Summary
================================================================================================
Remove        1 Package(s)
Reinstall     0 Package(s)
Downgrade     0 Package(s)

Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing        : finger-0.17-39.el6.i686                                               1/1

Removed:
  finger.i686 0:0.17-39.el6

Complete!
```

To update the package using yum

- To update the package using yum command, the syntax is
  #yum update <package name>
  #yum update httpd

```
[root@ktlinux ~]# 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@ktlinux ~]#
```

As there are no updates available for it, it is not showing anything to update

To install a package locally from a folder, pendrive or cd rom

- Move to the package where you have stored the package to be installed

```
[root@ktcl5 ~]# cd /
[root@ktcl5 /]# 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 <packagename> -y
  #yum localinstall finger* -y (or) **#yum localinstall finger-0.17-39.el6.i686.rpm -y**

```
[root@ktcl5 /]# 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>
#yum info finger

```
[root@ktcl5 /]# yum info finger
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Available Packages
Name       : finger
Arch       : i686
Version    : 0.17
Release    : 39.el6
Size       : 22 k
Repo       : KTREPO
Summary    : The finger client
License    : BSD
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.
           :
           : You should install finger if you'd like to retrieve finger information
           : from other systems.
```

To list and install a group of packages using yum

- To list the group of package the syntax is
  #yum grouplist

```
[root@ktcl5 /]# yum grouplist
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Group Process
KTREPO/group_gz                                                | 190 kB     00:00
Installed Groups:
   Additional Development
   Arabic Support
   Armenian Support
Available Groups:
    Afrikaans Support
    Albanian Support
    Amazigh Support
    Azerbaijani Support
    Backup Client
    Backup Server
    Basque Support
   Print Server
   Printing client
   Punjabi Support
   SNMP Support
   Server Platform
   Sinhala Support
   System administration tools
   Tajik Support
   Tamil Support
   Telugu Support
   Thai Support
   Urdu Support
   Venda Support
   Web Server
   X Window System
```

- Let's try install package from group called "urdu support", the syntax is
  #yum groupinstall <package name> -y
  #yum groupinstall urdu support -y

```
[root@ktcl5 /]# yum groupinstall urdu support -y
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Group Process
Installing:
 hunspell-ur                noarch          0.64-2.1.el6          KTREPO          308 k
 nafees-web-naskh-fonts     noarch          1.2-5.el6             KTREPO           65 k

Installed:
  hunspell-ur.noarch 0:0.64-2.1.el6          nafees-web-naskh-fonts.noarch 0:1.2-5.el6

Complete!
```

Removing a Group package using yum

- To remove a group, the syntax is
  #yum groupremove <package name>
  #yum groupremove urdu support

```
[root@ktlinux ~]# yum groupremove urdu support
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Group Process
KTREPO/group_gz                                                          | 190 kB     00:00 ...
No group named support exists
Resolving Dependencies
--> Running transaction check
---> Package dejavu-sans-fonts.noarch 0:2.30-2.el6 set to be erased
---> Package dejavu-sans-mono-fonts.noarch 0:2.30-2.el6 set to be erased
---> Package ibus-m17n.i686 0:1.3.0-1.el6 set to be erased
---> Package m17n-contrib-urdu.noarch 0:1.1.10-3.el6 set to be erased
---> Package paktype-naqsh-fonts.noarch 0:2.0-8.el6 set to be erased
---> Package paktype-tehreer-fonts.noarch 0:2.0-8.el6 set to be erased
--> Finished Dependency Resolution

Dependencies Resolved


================================================================================================
 Package         Arch   Version      Repository                                          Size
================================================================================================
Removing:
 dejavu-sans-fonts
                 noarch 2.30-2.el6   @anaconda-RedHatEnterpriseLinux-201009221732.i386/6.0 4.4 M
 dejavu-sans-mono-fonts
                 noarch 2.30-2.el6   @anaconda-RedHatEnterpriseLinux-201009221732.i386/6.0 1.0 M
 ibus-m17n       i686   1.3.0-1.el6  @anaconda-RedHatEnterpriseLinux-201009221732.i386/6.0  47 k
 m17n-contrib-urdu
                 noarch 1.1.10-3.el6 @anaconda-RedHatEnterpriseLinux-201009221732.i386/6.0 3.5 k
 paktype-naqsh-fonts
                 noarch 2.0-8.el6    @anaconda-RedHatEnterpriseLinux-201009221732.i386/6.0 620 k
 paktype-tehreer-fonts
                 noarch 2.0-8.el6    @anaconda-RedHatEnterpriseLinux-201009221732.i386/6.0 298 k

Transaction Summary
================================================================================================
Remove        6 Package(s)
Reinstall     0 Package(s)
Downgrade     0 Package(s)
```

And hence a group will be removed.

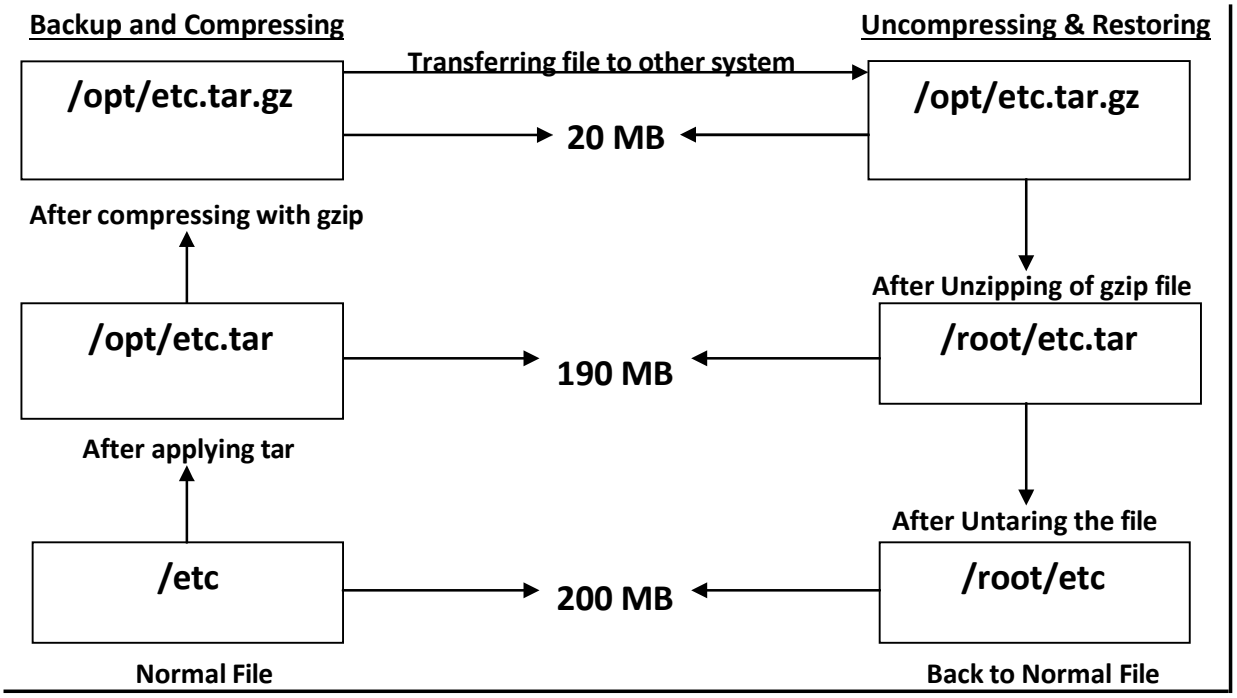
The software management can be learnt more by using manual pages like man yum and also man rpm etc.

If nothing works Google is yours…!!!

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@ktlinux ~]# 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@ktlinux ~]# du -h /opt/etc.tar
29M     /opt/etc.tar
[root@ktlinux ~]#
```

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@ktlinux ~]# gzip /opt/etc.tar
[root@ktlinux ~]#
```

- Now check the size of the file

```
[root@ktlinux ~]# cd /opt/
[root@ktlinux opt]# ls
etc.tar.gz  home  lost+found
[root@ktlinux opt]# du etc.tar.gz
7544    etc.tar.gz
[root@ktlinux 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@ktlinux ~]# scp /opt/etc.tar.gz 192.168.10.95:/root/
etc.tar.gz                                    100% 7544KB   7.4MB/s   00:01
[root@ktlinux ~]#
```

- 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@ktcl5 ~]# ls
anaconda-ks.cfg  Documents  etc.tar.gz  install.log.syslog  ktfile
Desktop          Downloads  install.log  ktdir              Music
[root@ktcl5 ~]# du -h etc.tar.gz
7.4M    etc.tar.gz
[root@ktcl5 ~]# gunzip etc.tar.gz
[root@ktcl5 ~]# ls
anaconda-ks.cfg  Documents  etc.tar     install.log.syslog  ktfile
Desktop          Downloads  install.log  ktdir              Music
[root@ktcl5 ~]# du -h etc.tar
29M     etc.tar
[root@ktcl5 ~]#
```

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@ktcl5 ~]# 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@ktcl5 ~]# ls
anaconda-ks.cfg  Downloads  install.log
Desktop          etc        install.log.syslog
Documents        etc.tar    ktdir
[root@ktcl5 ~]# du -h etc
8.0K    etc/avahi/etc
8.0K    etc/avahi/services
32K     etc/avahi
4.0K    etc/openldap/cacerts
12K     etc/openldap
```

That's it with backup and restore.

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 of 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.

chkconfig and service commands

There are 2 commands used to control services:

- **service -** This controls the starting and stopping of services during a session, these setting are not saved. If you start Apache this way but it is not set to start on boot using the above method then it will continue to run but on next boot will not start automatically.

- **chkconfig -** This controls which services are set to start on boot, by their nature these setting are saved and are applied at next boot. Changing these settings will not start the service immediately; it will just flag them to be started from the next boot.

- **The command use for maintaining a service is**

#service <name of the service> status	**---**	**T**o check the status of the service
#service <name of the service> start	**---**	To start the service
#service <name of the service > stop	**---**	To stop a service
#service <name of the service> reload	---	To reload the service
#service <name of the service> restart	---	To restart the service

- **The command use for service availability is**

#chkconfig - -list	**---**	To check the availability of service
#chkconfig <service> on	**---**	To make the service available after restart
#chkconfig <service> off	---	To make the service unavailable after restart

LAB WORK:-

Check the status of ftp service "vsftpd"

- To check the status of the above service
  #service vsftpd status

```
[root@ktlinux ~]# service vsftpd status
vsftpd is stopped
[root@ktlinux ~]#
```

Start the ftp services

- To start the ftp service, the command is
  #service vsftpd start

```
[root@ktlinux ~]# service vsftpd start
Starting vsftpd for vsftpd:                                [  OK  ]
[root@ktlinux ~]# service vsftpd status
vsftpd (pid 9947) is running...
[root@ktlinux ~]#
```

Reload the ftp services, may be required after doing some change in config file.

- To reload the service, the command is
  #service vsftpd reload

```
[root@ktlinux ~]# service vsftpd reload
Shutting down vsftpd:                                      [  OK  ]
Starting vsftpd for vsftpd:                                [  OK  ]
[root@ktlinux ~]#
```

To restart the ftp or any service, required when reload does not work

- To restart the ftp services, the command will be
  #service vsftpd restart

```
[root@ktlinux ~]# service vsftpd restart
Shutting down vsftpd:                                      [  OK  ]
Starting vsftpd for vsftpd:                                [  OK  ]
[root@ktlinux ~]#
```

Check the status of the all service availability.

- To check the status of all service availability, use
  #chkconfig - -list

```
[root@ktlinux ~]# chkconfig --list
NetworkManager  0:off   1:off   2:on    3:on    4:on    5:on    6:off
abrtd           0:off   1:off   2:off   3:on    4:off   5:on    6:off
acpid           0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
auditd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
```

Note: Where **0 1 2 3 4 5 6** are the run levels in Linux, The output shows that on which run level the service is available even after reboot.

Check the status of a particular service, say "vsftpd"

- To check the status of a vsftpd service, the command is
  #chkconfig - - list <name of the service>
  #chkconfig - - list vsftpd

```
[root@ktlinux ~]# chkconfig --list vsftpd
vsftpd          0:off   1:off   2:off   3:off   4:off   5:off   6:off
[root@ktlinux ~]#
```

Make the service availability on for vsftpd.

- To make the service availability on for vsftpd service,
  #chkconfig vsftpd on

```
[root@ktlinux ~]# chkconfig vsftpd on
[root@ktlinux ~]# chkconfig --list vsftpd
vsftpd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
[root@ktlinux ~]#
```

Make the service availability off for vsftpd

- To make the service availability off the command is
  #chkconfig vsftpd off

```
[root@ktlinux ~]# chkconfig vsftpd off
[root@ktlinux ~]# chkconfig --list vsftpd
vsftpd          0:off   1:off   2:off   3:off   4:off   5:off   6:off
[root@ktlinux ~]#
```

Make the service vsftpd availablily on only on runlevel 5

- **To make the service availablilty on, on a particular runlevel , the syntax is**
  #chkconfig - - level <1-6> <service> <on/off>
  #chkconfig - - level 5 vsftpd on

```
[root@ktlinux ~]# chkconfig --level 5 vsftpd on
[root@ktlinux ~]# chkconfig --list vsftpd
vsftpd          0:off   1:off   2:off   3:off   4:off   5:on    6:off
```

The same can be done for making service unavailable in a particular run level.

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** (day-mon). 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** process.
- The **init** 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 will be **1.**
- Only after init 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@ktlinux ~]# ps
  PID TTY          TIME CMD
10951 pts/0    00:00:00 bash
11523 pts/0    00:00:00 ps
[root@ktlinux ~]#
```

To see total number of processes running in the system

- The possible options which can be used with ps command are

  #ps –a

```
[root@ktlinux ~]# ps -a
  PID TTY          TIME CMD
11545 pts/0    00:00:00 ps
[root@ktlinux ~]#
```

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@ktlinux ~]# ps -u musab
  PID TTY          TIME CMD
11566 pts/1    00:00:00 bash
11591 pts/1    00:00:00 vim
[root@ktlinux ~]#
```

```
[root@ktlinux ~]# ps
  PID TTY          TIME CMD
10951 pts/0    00:00:00 bash
11523 pts/0    00:00:00 ps
[root@ktlinux ~]#
```

To see which process are attached with some terminals (tty) and which are not

- **#ps –x**

```
[root@ktlinux ~]# 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.

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@ktlinux ~]# ps -G musab
  PID TTY          TIME CMD
11566 pts/1    00:00:00 bash
11591 pts/1    00:00:00 vim
[root@ktlinux ~]# pgrep -G musab
11566
11591
```

To see the offline process of the system, already executed

- **#ps –aux**

```
[root@ktlinux ~]# ps -aux
root     10949  0.0  2.3  52144 12712 ?        Rl   16:49   0:02 /usr/bin/gnome-terminal
root     10950  0.0  0.1   2032   616 ?        S    16:49   0:00 gnome-pty-helper
root     10951  0.0  0.3   5212  1668 pts/0    Ss   16:49   0:00 /bin/bash -l
postfix  11471  0.0  0.3  10244  2084 ?        S    20:05   0:00 pickup -l -t fifo -u
root     11555  0.0  0.2   5084  1608 pts/1    Ss   20:37   0:00 bash
root     11564  0.0  0.3   7068  1648 pts/1    S    20:37   0:00 su - musab
musab    11566  0.0  0.2   5084  1592 pts/1    S    20:37   0:00 -bash
musab    11591  0.0  0.5  10776  2768 pts/1    S+   20:37   0:00 vim
root     11706  0.0  0.0      0     0 ?        S    21:11   0:00 [flush-253:0]
root     11710  2.0  0.1   4764   996 pts/0    R+   21:12   0:00 ps -aux
```

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 sen by
  #kill –l

```
[root@ktlinux ~]# 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
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at tty
SIGTTIN	21,21,26	Stop	tty input for background process
SIGTTOU	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 signal 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@ktlinux ~]# ps -u musab
  PID TTY          TIME CMD
11566 pts/1    00:00:00 bash
11591 pts/1    00:00:00 vim
[root@ktlinux ~]# kill -9 11591
[root@ktlinux ~]# ps -u musab
  PID TTY          TIME CMD
11566 pts/1    00:00:00 bash
[root@ktlinux ~]#
```

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@ktlinux ~]$ cat > hello
```

- Check its pid and kill it by using **20, #ps –u musab**
  #kill -20

```
[root@ktlinux ~]# ps -u musab
  PID TTY          TIME CMD
11566 pts/1    00:00:00 bash
12041 pts/1    00:00:00 cat
[root@ktlinux ~]# kill -20 12041
```

- check its effect at the user's console

```
[musab@ktlinux ~]$ cat > hello

[1]+  Stopped                 cat > hello
```

- Restart the process continue working
  #fg <pid>
  #fg 1

```
[musab@ktlinux ~]$ 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 19**. **The lower the number the more priority that task gets. If the niceness value is high number like 19 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 19)> <command>
  #nice –n 5 cat > ktfile

```
[root@ktlinux ~]# nice -n 5 cat > ktfile
Hello World
Welcome to Kernel Technologies
```

- Log in to other terminal and check the nice value for the above command/ process.
  #ps –elf

```
[root@ktlinux ~]# 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@ktlinux ~]# renice 2 13155
13155: old priority 5, new priority 2
[root@ktlinux ~]#
```

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	**Process ID**
USER	**Effective User ID**
PR	**Dynamic priority**
NI	**Nice value, also known as base priority**
VIRT	**Virtual Size of the task. This includes the size of process's executable binary, the data area and all the loaded shared libraries.**
RES	**The size of RAM currently consumed by the task. Swapped out portion of the task is not included.**
SHR	**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.**
S	**Task status**
%CPU	**The percentage of CPU time dedicated to run the task since the last top's screen update.**
%MEM	**The percentage of RAM currently consumed by the task.**
TIME+	**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.**
Command	**Showing program names**

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.

Just press the following key while running top and the output will be sorted in real time.

- **M – Sort by memory usage**
- **P – Sort by CPU usage**
- **T – Sort by cumulative time**
- **z – Color display**
- **k – Kill a process**
- **q – quit**
- **r – to renice a process**
- **h - help**

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

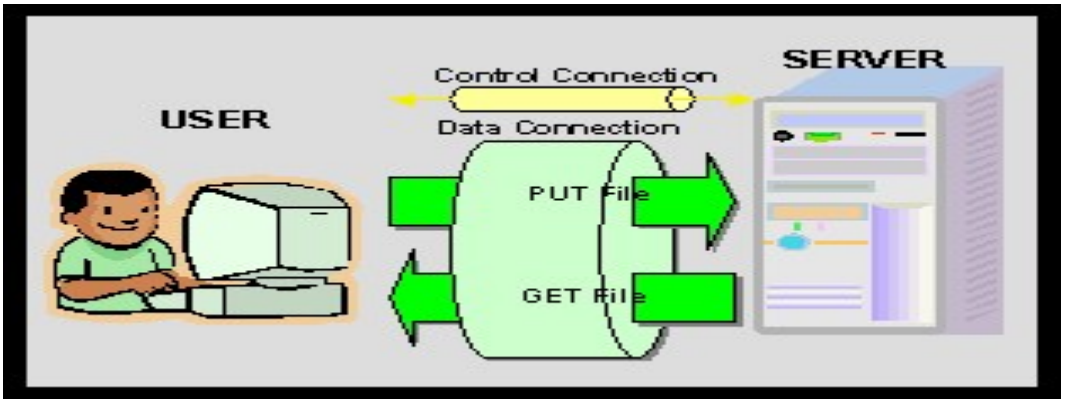
To renice a process with PID 4, then press "r" and a prompt will ask you for PID enter 4 and press enter. When prompted for renice value give any value .

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?

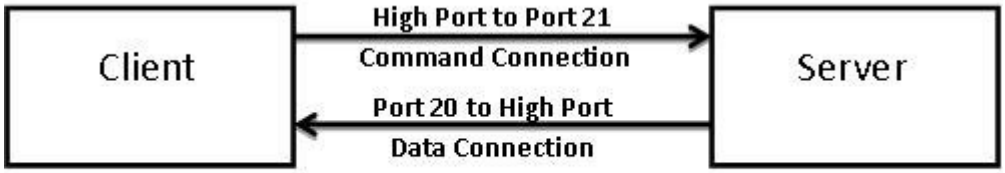


Figure 01 – Active FTP Connection

- 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?

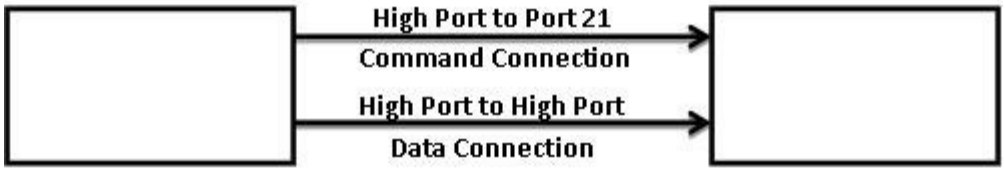


Figure 02 – Passive FTP Connection

- 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.

Profile of ftp server

- **Use : Ftp is used for uploading and downloading the files.**
- **Disadvantage : Directory cannot be uploaded or downloaded.**
- **Package : vsftpd**
- **Daemon : vsftpd (Very Secure Ftp daemon)**
- **Script : /etc/initd/vsftpd**
- **Port no : 21 (Tcp) > 1024 (Udp, Random)**
- **Configuration files : /etc/vsftpd/vsftpd.conf**
  /etc/vsftpd/user_list
  /etc/vsftpd/ftpuser
  /etc/pam.d/vsftpd
- **Document Root : /var/ftp**
- **Home directory : /var/ftp (which is created only when the package is installed.**

Steps to configuring ftp server for downloading a file:

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@ktcl3 ~]# 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                                         | 1.7 kB     00:00
duration: 117(ms)
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@ktcl3 ~]# 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 6 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@ktcl3 ~]# cd /var/ftp/pub/
[root@ktcl3 pub]# ls
[root@ktcl3 pub]# touch file{1..5}
[root@ktcl3 pub]# ls
file1  file2  file3  file4  file5
[root@ktcl3 pub]#
```

Step3: Restart the ftp service

- **#service vsftpd restart**

```
[root@ktcl3 pub]# service vsftpd restart
Shutting down vsftpd:                                      [  OK  ]
Starting vsftpd for vsftpd:                                [  OK  ]
[root@ktcl3 pub]#
```

Step4: Make the service enable even after reboot of the system

- **To make a service enable use the following command**
  #chkconfig vsftpd on

```
[root@ktcl3 pub]# chkconfig vsftpd on
[root@ktcl3 pub]# chkconfig --list vsftpd
vsftpd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
[root@ktcl3 pub]#
```

Step5: 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@ktcl1 ~]# rpm -q ftp
package ftp is not installed
[root@ktcl1 ~]#
```

- **To install ftp package either download it from redhat website or install it from RHEL6 DVD**
- **Move to the package folder and installed it.**
  #rpm –ivh <package name>

```
[root@ktcl1 ~]# 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@ktcl1 ~]# rpm -q ftp
ftp-0.17-51.1.el6.x86_64
```

- **Now connect to ftp server using its IP**
  To connect to ftp server use the following command
  #ftp <ftp server's IP>
  #ftp 192.168.10.93
  Use "ftp or anonymous" as login name
  Press enter without giving any password

```
[root@ktcl1 ~]# 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>
```

<u>Note:</u> 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**
  To download files use the following 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

```
ftp> quit
221 Goodbye.
[root@ktcl1 ~]# 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               Videos
Documents        gana                           Pictures
Downloads        install.log                    Public
[root@ktcl1 ~]#
```

To connect to the ftp server graphically open 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@ktlinux ~]# cd /var/ftp
[root@ktlinux ftp]# ls
pub
[root@ktlinux ftp]# mkdir upload
[root@ktlinux ftp]# ls
pub  upload
[root@ktlinux 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@ktlinux ftp]# ls -ld upload
drwxr-xr-x. 2 root root 4096 Nov  4 19:12 upload
[root@ktlinux ftp]# chgrp ftp upload
[root@ktlinux ftp]# ls -ld upload
drwxr-xr-x. 2 root ftp 4096 Nov  4 19:12 upload
[root@ktlinux ftp]#
```

- **Adding the write permission to upload directory**
  #chmod g+w upload

```
[root@ktlinux ftp]# ls -ld upload
drwxr-xr-x. 2 root ftp 4096 Nov  4 19:12 upload
[root@ktlinux ftp]# chmod g+w upload
[root@ktlinux ftp]# ls -ld upload
drwxrwxr-x. 2 root ftp 4096 Nov  4 19:12 upload
[root@ktlinux 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@ktcl5 ~]# cd sample
[root@ktcl5 sample]# ls
ktf1  ktf2  ktf3  ktf4  ktf5
[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
ktf1  ktf2  ktf3  ktf4  ktf5
ftp> prompt
Interactive mode off.
ftp> mput ktf1 ktf2 ktf3
local: ktf1 remote: ktf1
227 Entering Passive Mode (192,168,10,98,138,43).
550 Permission denied.
local: ktf2 remote: ktf2
227 Entering Passive Mode (192,168,10,98,87,254).
550 Permission denied.
local: ktf3 remote: ktf3
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.
# has an effect if the above global write enable is activated. Al
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES
```

```
# Uncomment this to allow the anonymous FTP user to upload files. Th
# has an effect if the above global write enable is activated. Also,
# obviously need to create a directory writable by the FTP user.
anon_upload_enable=YES
```

- **Restart the ftp service**
  #service vsftpd restart

```
[root@ktlinux ~]# service vsftpd restart
Shutting down vsftpd:                                      [  OK  ]
Starting vsftpd for vsftpd:                                [  OK  ]
[root@ktlinux ~]#
```

Step4: Again login to client system and try again to upload the files into ftp server

```
[root@ktcl5 sample]# ls
ktf1  ktf2  ktf3  ktf4  ktf5
[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 ktf1
local: ktf1 remote: ktf1
227 Entering Passive Mode (192,168,10,98,192,136).
553 Could not create file.
ftp>
```

If the 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

- **Check the Booleans for ftp using following command**
  #getsebool –a |grep ftp

```
[root@ktlinux ~]# 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
```

- **Make the above Boolean value as "on"**
- **To make it on use the following command**
  #setsebool –P allow_ftpd_anon_write on

```
[root@ktlinux ~]# setsebool -P allow_ftpd_anon_write on
[root@ktlinux ~]# 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
```

- **Add read write permission in context of upload directory using following command**
  #chcon –t public_content_rw_t

```
[root@ktlinux ftp]# ls -ldZ upload
drwxrwxr-x. root ftp unconfined_u:object_r:public_content_t:s0 upload
[root@ktlinux ftp]# chcon -t public_content_rw_t upload
[root@ktlinux ftp]# ls -ldZ upload
drwxrwxr-x. root ftp unconfined_u:object_r:public_content_rw_t:s0 upload
[root@ktlinux ftp]#
```

- **Finally login into client machine, access the ftp server and try uploading the files in it.**

```
[root@ktcl5 sample]#
[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> !ls
ktf1  ktf2  ktf3  ktf4  ktf5
ftp> prompt
Interactive mode off.
ftp> mput ktf1 ktf2
local: ktf1 remote: ktf1
227 Entering Passive Mode (192,168,10,98,121,220).
150 Ok to send data.
226 Transfer complete.
local: ktf2 remote: ktf2
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 ktf1
-rw-------    1 14       50              0 Nov 04 16:28 ktf2
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@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): 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 #service ftp restart

- **Now try login from client into ftp server as root**

```
[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): root
331 Please specify the password.
Password:
500 OOPS: cannot change directory:/root
Login failed.
ftp>
```

Though everything right, but still it is not allowing us to login as root because the home dir is not able to change. It is again because of SELinux.

- **To solve the above problem, login to ftp server and change the following Boolean for ftp**

```
[root@ktlinux ~]# 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
```

- **Change the Boolean value to on for ftp_home_dir by following command #setsebool –P ftp_home_dir on**

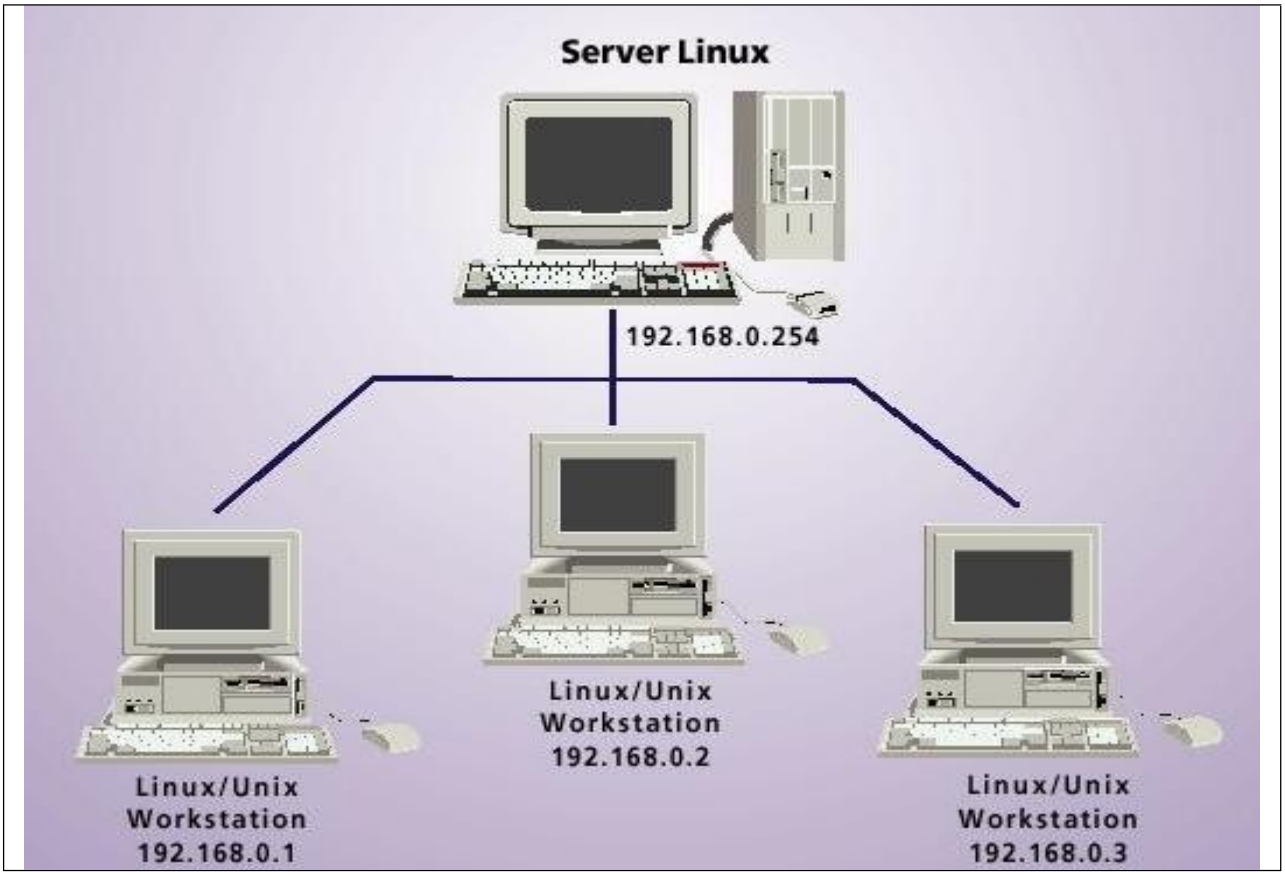
```
[root@ktlinux ~]# setsebool -P ftp_home_dir on
[root@ktlinux ~]# getsebool -a |grep ftp
allow_ftpd_use_cifs --> off
allow_ftpd_use_nfs --> off
ftp_home_dir --> on
```

- **Try logging again as root in ftp server**

```
[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>
```

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 window 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 : rpc.nfsd, rpc.mountd, rpc.statd, rpc.lockd, rpc.rquotad**
- **Script : /etc/init.d/nfs**
- **Port number : 2049**
- **Configuration File : /etc/exports**
- **Other imp files : /var/lib/nfs/etab, /var/lib/nfs/rmtab**

Workflow of NFS and its Daemon

Workflow of NFS

Daemons on client as well as Server

Steps to configure NFS server:

Step1: Install the NFS package using yum or rpm.
Step2: Create a dir or directory on partition 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@ktcl1 /]# rpm -q nfs-utils
nfs-utils-1.2.3-7.el6.x86_64
[root@ktcl1 /]#
```

- **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/vda11           3812        3927      931738+  8e  Linux LVM
/dev/vda12           3928        3979      417658+  8e  Linux LVM
/dev/vda13           3980        4056      618471   83  Linux
```

- **Update the partition table and format it**
  #partx –a /dev/vda
  #mkfs.ext4 /dev/vda13
- **Create a directory and mount the partition over it and also make it permanent in /etc/fstab**

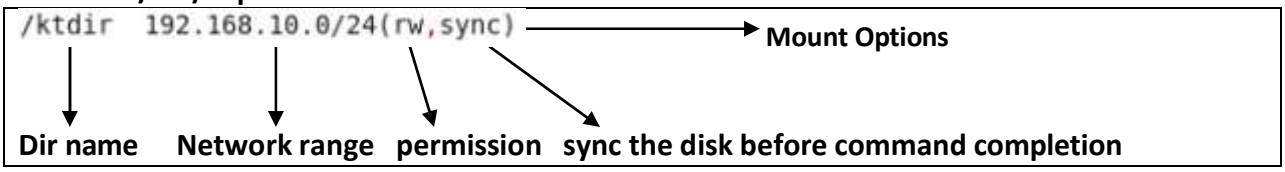
```
[root@ktcl1 ~]# mkdir /ktdir
[root@ktcl1 ~]# vim /etc/fstab
[root@ktcl1 ~]# mount -a
[root@ktcl1 ~]# mount
/dev/mapper/vg_ktcl1-rootlv on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/vda2 on /boot type ext4 (rw)
/dev/mapper/vg_ktcl1-homelv on /home type ext4 (rw)
/dev/mapper/vg_ktcl1-optlv on /opt type ext4 (rw)
/dev/mapper/vg_ktcl1-usrlv on /usr type ext4 (rw)
/dev/mapper/vg_ktcl1-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 /ktdir type ext4 (rw)
[root@ktcl1 ~]#
```

- **Access the mount point and add some data in it**

```
[root@ktcl1 ~]# cd /ktdir
[root@ktcl1 ktdir]# touch ktfile{1..5}
[root@ktcl1 ktdir]# ls
ktfile1  ktfile2  ktfile3  ktfile4  ktfile5  lost+found
[root@ktcl1 ktdir]#
```

<u>Step3</u>: Export the directory by editing /etc/exports file and using exportfs command

- **Edit the /etc/exports file**
  #vim /etc/exports



/ktdir	**:**	Name of the directory to be exported
192.168.10.0/24	**:**	Range of network where directory can be mounted To give permission to only one node, just give the IP ADDR Of that node (ex: 192.168.10.93)
(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 users

- **Now run the exportfs command to export the directory**
  #exportfs –avr

```
[root@ktcl1 ~]# exportfs -avr
exporting 192.168.10.0/24:/ktdir
[root@ktcl1 ~]#
```

<u>Options:</u>

-a **Exports or un-exports all directories**
-r **Reexport all directories**
-u **Unexports one or more directories**
-v **Provides verbose output**

Step4: Restart the services and make it permanent.

- **#service nfs restart**

```
[root@ktcl1 ~]# service nfs restart
Shutting down NFS mountd:                                  [  OK  ]
Shutting down NFS daemon:                                  [  OK  ]
Shutting down NFS quotas:                                  [  OK  ]
Shutting down NFS services:                                [  OK  ]
Starting NFS services:                                     [  OK  ]
Starting NFS quotas:                                       [  OK  ]
Starting NFS daemon:                                       [  OK  ]
Starting NFS mountd:                                       [  OK  ]
[root@ktcl1 ~]#
```

Check the directories which is exported in /var/lib/nfs/etab and /var/lib/nfs/rmtab

```
[root@ktcl1 ~]# cat /var/lib/nfs/etab
/ktdir  192.168.10.0/24(rw,sync,wdelay,hide,nocrossmnt,secure,root_squash,no_all
_squash,no_subtree_check,secure_locks,acl,anonuid=65534,anongid=65534)
[root@ktcl1 ~]# cat /var/lib/nfs/rmtab
[root@ktcl1 ~]#
```

- **Note: stop the iptables services by using #service iptables stop and chkconfig iptables off.**

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@ktcl1 ~]# rpm -q nfs-utils
nfs-utils-1.2.3-7.el6.x86_64
[root@ktcl1 ~]#
```

It is already installed, if it is not installed use yum install nfs-utils* -y

Step2: check and start the NFS services and make it permanent.

#service nfs start

#chkconfig - -list nfs

```
[root@ktcl3 ~]# service nfs start
Starting NFS services:                                     [  OK  ]
Starting NFS quotas:                                       [  OK  ]
Starting NFS daemon:                                       [  OK  ]
Starting NFS mountd:                                       [  OK  ]
[root@ktcl3 ~]# chkconfig nfs on
[root@ktcl3 ~]#
```

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@ktcl2 ~]# showmount -e 192.168.10.91
Export list for 192.168.10.91:
/ktdir 192.168.10.0/24
[root@ktcl2 ~]#
```

Step4: Make a directory and mount NFS over it.

- **#mkdir /ktnfs**
- **#mount –t nfs 192.168.10.91:/ktdir /ktnfs**

```
[root@ktcl2 ~]# mkdir /ktnfs
[root@ktcl2 ~]# mount -t nfs 192.168.10.91:/ktdir /ktnfs
[root@ktcl2 ~]# cd /ktnfs/
[root@ktcl2 ktnfs]# ls
ktfile1  ktfile2  ktfile3  ktfile4  ktfile5  lost+found
[root@ktcl2 ktnfs]#
```

Step5: Add some data to it and check the same is updated on server side.

```
[root@ktcl2 ktnfs]# touch nfs{1..4}
touch: cannot touch `nfs1': Permission denied
touch: cannot touch `nfs2': Permission denied
touch: cannot touch `nfs3': Permission denied
touch: cannot touch `nfs4': Permission denied
[root@ktcl2 ktnfs]#
```

- **Note that it is showing permission error because on server side the directory does not have write permissions neither for group nor for others.**
- **Log into server and add write permission to NFS directory**

```
[root@ktcl1 /]# ls -ld ktdir
drwxr-xr-x 3 root root 4096 Nov  8 13:25 ktdir
[root@ktcl1 /]# chmod go+w ktdir
[root@ktcl1 /]# ls -ld ktdir
drwxrwxrwx 3 root root 4096 Nov  8 13:25 ktdir
[root@ktcl1 /]#
```

- **Now , Again move back to client machine and try uploading some files**

```
[root@ktcl2 ktnfs]# touch nfs{1..4}
[root@ktcl2 ktnfs]# ls
ktfile1  ktfile2  ktfile3  ktfile4  ktfile5  lost+found  nfs1  nfs2  nfs3  nfs4
[root@ktcl2 ktnfs]#
```

- **To make it permanent mount edit /etc/fstab file as follows**

```
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
192.168.10.91:/ktdir    /ktnfs                  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**

Steps to configure 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@ktcl1 /]# rpm -q autofs
autofs-5.0.5-31.el6.x86_64
[root@ktcl1 /]#
```

- 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
/ktnfs          /etc/auto.ktnfs --timeout 10
```

Standby time

Dir to mount NFS **auto mount configuration file for this mount point**

Step3: Create /etc/auto.ktnfs file and /ktnfs directory if not created earlier

- **#vim /etc/auto.ktnfs**

```
ktnfs   -rw     192.168.10.91:/ktdir
```

Name for autofs **Permissions** **NFS directory name**

Step4: Stop and start the autofs service

- **#service autofs stop**
- **#service autofs start**
- **#chkconfig autofs on**

```
[root@ktcl1 /]# service autofs stop
Stopping automount:                                        [  OK  ]
[root@ktcl1 /]# service autofs start
Starting automount:                                        [  OK  ]
[root@ktcl1 /]# chkconfig autofs on
[root@ktcl1 /]#
```

Step5: log into the given directory in /etc/auto.master i.e. /ktnfs and check that if NFS is mounted by mount command

```
[root@ktcl1 /]# cd /ktnfs
[root@ktcl1 ktnfs]# mount
```

Note: Still NFS will not be mounted

Step6: change the directory to the name given in /etc/auto.ktnfs i.e. ktnfs and then auto mounting will be done.

#cd ktnfs

#mount

```
[root@ktcl1 ktnfs]# cd ktnfs
[root@ktcl1 ktnfs]# ls
ktfile1  ktfile2  ktfile3  ktfile4  ktfile5  lost+found  nfs1  nfs2  nfs3  nfs4
[root@ktcl1 ktnfs]#
```

Steps for removing NFS

Step1: Remove all autofs details from all config files like /etc/auto.master and /etc/auto.ktnfs

Step2: un-export all the directory which was exported earlier using following command

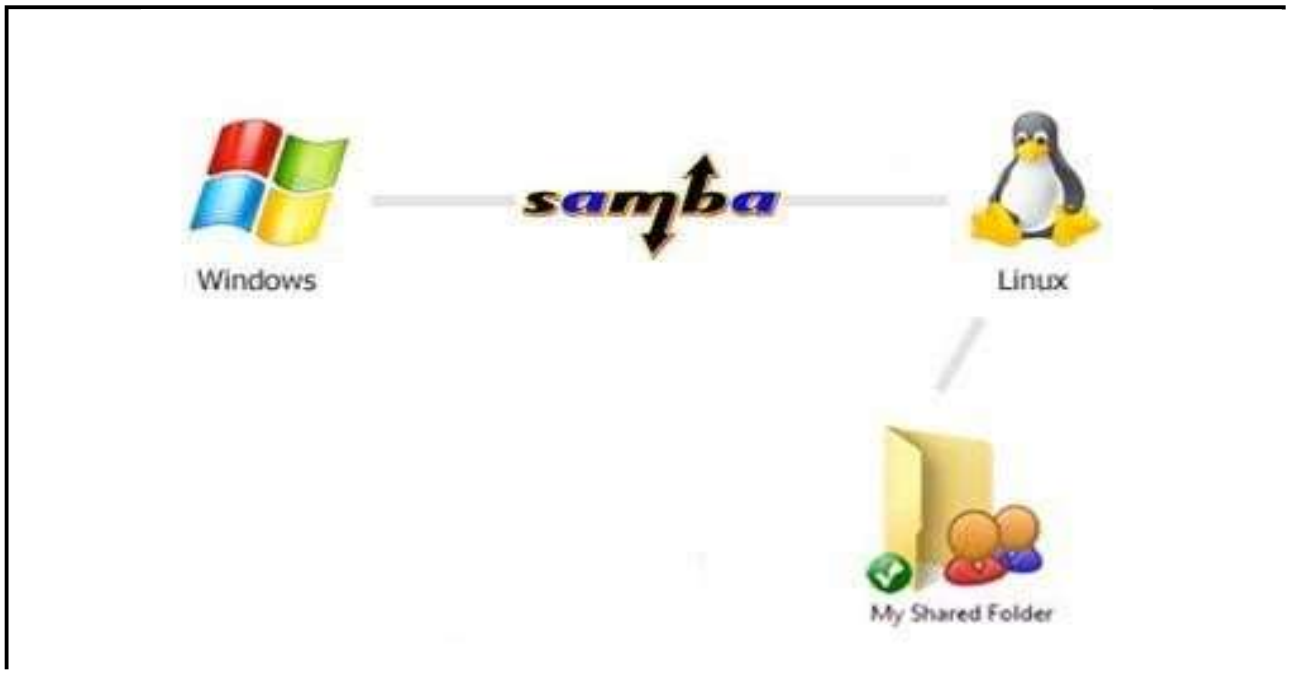
- **# exportfs –auv**

```
[root@ktcl1 ~]# cat /var/lib/nfs/etab
/ktdir  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@ktcl1 ~]# exportfs -auv
[root@ktcl1 ~]# cat /var/lib/nfs/etab
[root@ktcl1 ~]#
```

Note:- if you don't have DNS and you don't want use IP but want to use hostname instead, update hostname with its ip in /etc/hosts file and then you can use hostname instead of IP

Okay now finally we've done with all NFS activities.

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**
Script	**:**	**/etc/init.d/smb, /etc/init.d/nmb**
Portno	**:**	**187 (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**

Steps to configure SAMBA server

Step1: Check and Install the SAMBA package, if not installed

#rpm –q samba

```
[root@ktcl1 ~]# rpm -q samba
package samba is not installed
[root@ktcl1 ~]#
```

- **Install the package using yum**
  #yum install samba* -y

```
[root@ktcl1 ~]# 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           RHEL6
 samba-winbind          x86_64              3.5.6-86.el6           RHEL6

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 /ktsamba**
- **#chmod 777 /ktsamba**

```
[root@ktcl1 ~]# mkdir /ktsamba
[root@ktcl1 ~]# chmod 777 /ktsamba
[root@ktcl1 ~]#
```

.

Step3: Check the context of the directory and change it according to samba

- **#ls –ldZ /ktsamba**
- **#chcon –t samba_share_t /ktsamba**

```
[root@ktcl1 ~]# ls -ldZ /ktsamba/
drwxrwxrwx. root root unconfined_u:object_r:default_t:s0 /ktsamba/
[root@ktcl1 ~]# chcon -t samba_share_t /ktsamba/
[root@ktcl1 ~]# ls -ldZ /ktsamba/
drwxrwxrwx. root root unconfined_u:object_r:samba_share_t:s0 /ktsamba/
[root@ktcl1 ~]#
```

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 "ktuser", let's just make it samba user**
  #smbpasswd –a <username>
  #smbpasswd –a ktuser
  Give password twice and wait till it add the user

```
[root@ktcl1 ~]# smbpasswd -a ktuser
New SMB password:
Retype new SMB password:
Added user ktuser.
[root@ktcl1 ~]#
```

Note: To delete a user from samba use #smbpasswd –x <user name>

- **To check all the samba user use**
  #pdbedit –L

```
[root@ktcl1 ~]# pdbedit -L
ktuser:515:
[root@ktcl1 ~]#
```

Step5: Go to the configuration file i.e. /etc/samba/smb.conf and make the following changes

- **Open the /etc/samba/smb.conf and copy the last seven lines shown below and paste it at the last to 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**

```
############################################
        [ktshare]
        comment = Public Stuff
        path = /ktsamba
        public = no
        valid users = ktuser
        writable = yes
        printable = no
        hosts allow = 192.168.10.
```

Explanation about the above fields

- **[ktshare] : Share Name**
- **Comment = Public Stuff : Comment**
- **Path = /ktsamba : Share Directory**
- **Public = no : Public Access (Every user in network)**
- **Valid user = ktuser : Authorized user**
- **Writable = yes : Write Permission**
- **Printable = yes : Print permission**
- **Host allow= 192.168.10. : Network Range or host range**

Note: 192.168.10. Represent entire 192.168.10 network

Step5: Test the samba parameters andrestart the service and make it enable after reboot

- **To test the parameters us the following command**
  #testparm

```
[root@ktcl2 /]# 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 "[ktshare]"
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

[ktshare]
        comment = Public Stuff
        path = /ktsamba
        valid users = ktuser
        read only = No
        hosts allow = 192.168.10.
[root@ktcl2 /]#
```

- **We have to restart two services here**
- **#smb and #nmb and make it add to enable after reboot**
  #service smb restart
  #service nmb restart
  #chkconfig smb on
  #chkconfig nmb on

```
[root@ktcl1 ~]# service smb restart
Shutting down SMB services:                                [  OK  ]
Starting SMB services:                                     [  OK  ]
[root@ktcl1 ~]# chkconfig smb on
[root@ktcl1 ~]# service nmb restart
Shutting down NMB services:                                [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@ktcl1 ~]# chkconfig nmb on
[root@ktcl1 ~]#
```

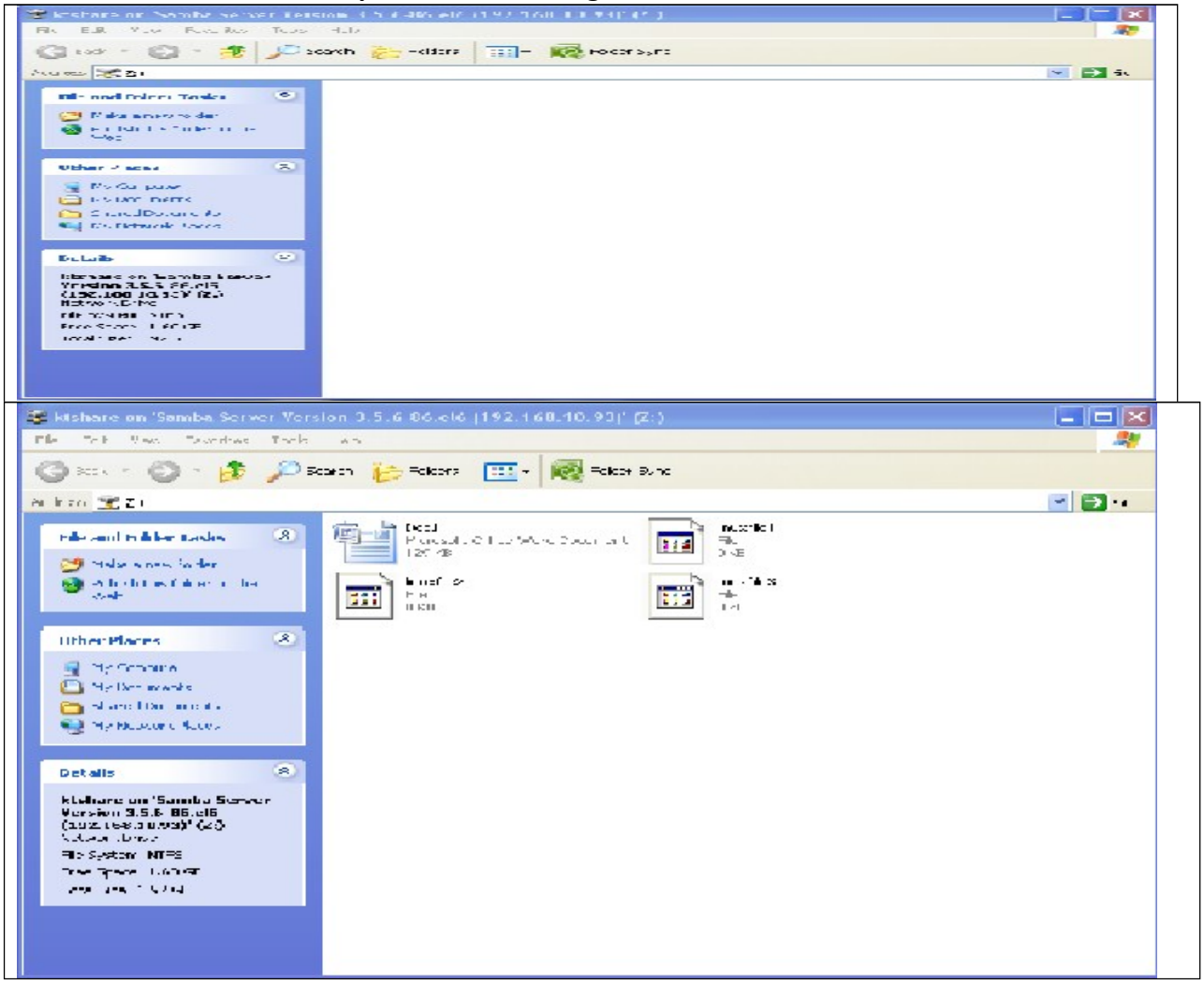
<u>Windows as a samba client</u>:

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**

- **Now a window will be opened, start adding some data to it from both sides**



Linux as client of SAMBA

- **Log into Linux machine and check how many samba servers are there in your network**
  #findsmb

```
[root@ktcl5 ~]# 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   KTCL2            [MYGROUP] [Unix] [Samba 3.5.6-86.el6]
192.168.10.98   KTLINUX         +[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@ktcl5 ~]# 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
)
        ktshare         Disk      Public Stuff
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-86.el6]

        Server               Comment
        ---------            -------
        KTCL2                Samba Server Version 3.5.6-86.el6
        KTLINUX              Samba Server Version 3.5.4-68.el6

        Workgroup            Master
        ---------            -------
        CL                   CL3
        KTS                  KTADS
        MYGROUP              KTLINUX
        WORKGROUP            KTLINUX
```

- **To connect to the samba server use the following syntax**
  #smbclient //<server IP>/<share name> -U <User name>
  #smbclient //192.168.10.93/ktshare –U ktuser

```
[root@ktcl5 ~]# smbclient //192.168.10.93/ktshare -U ktuser
Enter ktuser's password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-86.el6]
smb: \> ls
  .                                   D        0  Wed Nov  9 20:40:18 2011
  ..                                 DR        0  Wed Nov  9 20:00:43 2011
  linuxfile3                          0  Wed Nov  9 20:40:18 2011
  Doc1.docx                           A   131602  Sat Oct  1 15:26:29 2011
  linuxfile1                          0  Wed Nov  9 20:40:18 2011
  linuxfile2                          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**
- **The syntax is**
  #mount –t <type of fs> //<server IP address>/<share name> /<mount point> -O user= <user name>.
  #mount –t cifs //192.168.10.93/ktshare /mnt –o user=ktuser

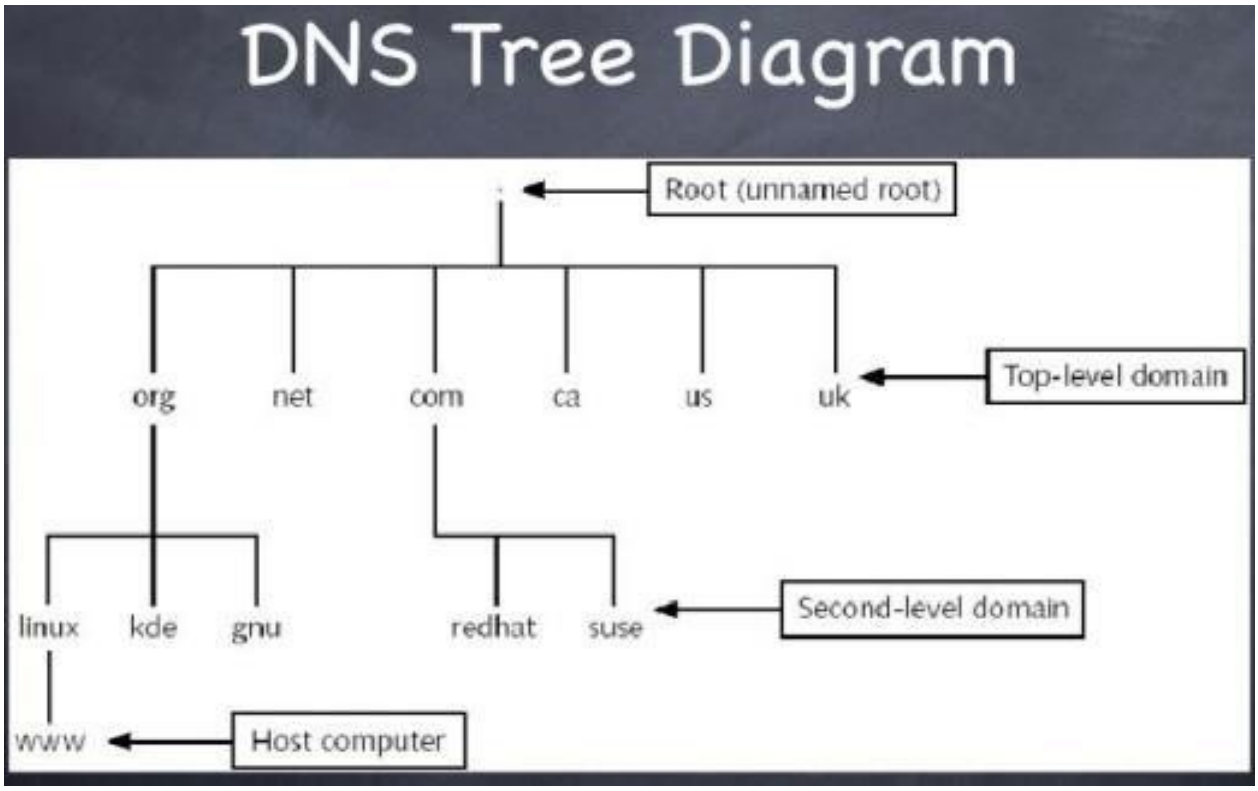
```
[root@ktcl5 ~]# mount -t cifs //192.168.10.93/ktshare /mnt -o user=ktuser
Password:
[root@ktcl5 ~]# mount
/dev/mapper/rootvg_ktcl5-LogVol00 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda2 on /boot type ext4 (rw)
/dev/mapper/rootvg_ktcl5-LogVol01 on /home type ext4 (rw)
/dev/mapper/rootvg_ktcl5-LogVol04 on /opt type ext4 (rw)
/dev/mapper/rootvg_ktcl5-LogVol05 on /tmp type ext4 (rw)
/dev/mapper/rootvg_ktcl5-LogVol02 on /usr type ext4 (rw)
/dev/mapper/rootvg_ktcl5-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/ktshare/ on /mnt type cifs (rw,mand)
[root@ktcl5 ~]#
```

That’s up with SAMBA Server and Client configuration; keep working on in to learn more about it.

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:

House name:	**Ameerpet Road**
Town:	**Hyderabad**
County:	**Hyderabad**
Country:	**India**

An Internet domain name looks like:

Host name	**www**
Domain	**google**
Second level domain	**co**
Top-level domain	**In**

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**
Script	**:**	**/etc/init.d/named**
Port	**:**	**53**
Configuration File	**:**	**/etc/named.conf**
Document root	**:**	**/var/named/**
Daemon	**:**	**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@ktadm ~]# rpm -q bind
package bind is not installed
[root@ktadm ~]#
```

- **To install the package use yum or rpm command**

```
[root@ktadm ~]# 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**
  #hostname ktadm.kt.com (where kt.com is the FQDN) and Make it permanent in /etc/sysconfig/network file

```
[root@ktadm ~]# hostname ktadm.kt.com
[root@ktadm ~]# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=ktadm.kt.com
[root@ktadm ~]#
```

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

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.10.95   ktadm.kt.com
```

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; 192.168.10.95; };
        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; 192.168.10.0/24; };
        recursion yes;
```

Where 192.168.10.95 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

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
Once pasted, edit the fields as follows **Where "kt.com" is the name of our domain** **And "10.168.192.in-addr.arpa" is the reverse order of our domain network.** **"kt.flz" is the name of the forward lookup zone file and...** **"kt.rlz" is the name of the reverse lookup zone file.**

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 kt.flz and edit it.**
  #cd /var/named
  #cp –p named.localhost kt.flz

```
[root@ktadm ~]# cd /var/named/
[root@ktadm named]# ls
chroot  dynamic   named.empty      named.loopback
data    named.ca  named.localhost  slaves
[root@ktadm named]# cp -p named.localhost kt.flz
[root@ktadm named]# vim kt.flz
```

- **Edit kt.flz file as follows**

```
$TTL 86400
@       IN SOA  ktadm.kt.com.   root.kt.com. (
                                        201111091       ; serial
                                        1D              ; refresh
                                        1H              ; retry
                                        1W              ; expire
                                        3H )            ; minimum
        NS      ktadm.kt.com.
ktadm   A       192.168.10.95
ktcl1   A       192.168.10.91
ktcl2   A       192.168.10.92
ktcl3   A       192.168.10.93
        A       127.0.0.1
```

Details about the fields used above:

- **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

- **Copy again named.localhost, this time as kt.rlz and edit it as shown below.**
  #cp –p named.localhost kt.rlz
  #vim kt.rlz

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 kt.com kt.flz (if you are not in named dir give absolute path)

```
[root@ktadm named]# named-checkzone kt.com /var/named/kt.flz
zone kt.com/IN: loaded serial 201111091
OK
```

#named-chkzone kt.com kt.rlz

```
[root@ktadm named]# named-checkzone kt.com /var/named/kt.rlz
zone kt.com/IN: loaded serial 2011110951
OK
```

Step7: Add the address of DNS server in /etc/resolv.conf

- **Edit the /etc/resolv.conf and add the IP of DNS server**
  #vim /etc/resolv.conf

```
# Generated by NetworkManager
search kt.com
nameserver 192.168.10.95
```

Step7: Restart the appropriate services

- **Restart the named service**
  #service named restart

```
[root@ktadm named]# service named restart
Stopping named: .                                          [  OK  ]
Starting named:                                            [  OK  ]
[root@ktadm named]#
```

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 ktadm.kt.com

```
[root@ktadm named]# dig ktadm.kt.com

; <<>> DiG 9.7.3-RedHat-9.7.3-2.el6 <<>> ktadm.kt.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10356
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;ktadm.kt.com.                  IN      A

;; ANSWER SECTION:
ktadm.kt.com.           86400   IN      A       192.168.10.95

;; AUTHORITY SECTION:
kt.com.                 86400   IN      NS      ktadm.kt.com.

;; Query time: 0 msec
;; SERVER: 192.168.10.95#53(192.168.10.95)
;; WHEN: Fri Nov 11 18:28:49 2011
;; MSG SIZE  rcvd: 60
```

- **Check with giving IP of hostname**
  #dig –x 192.168.10.95

```
[root@ktadm ~]# dig -x 192.168.10.95

; <<>> DiG 9.7.3-RedHat-9.7.3-2.el6 <<>> -x 192.168.10.95
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34843
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

;; ANSWER SECTION:
95.10.168.192.in-addr.arpa. 86400 IN    PTR     ktadm.kt.com.

;; AUTHORITY SECTION:
10.168.192.in-addr.arpa. 86400  IN      NS      ktadm.kt.com.

;; ADDITIONAL SECTION:
ktadm.kt.com.           86400   IN      A       192.168.10.95

;; Query time: 1 msec
;; SERVER: 192.168.10.95#53(192.168.10.95)
;; WHEN: Fri Nov 11 18:33:49 2011
;; MSG SIZE  rcvd: 100
```

- **Check the same with client's IP and Host name**
  #dig ktcl1.kt.com

```
[root@ktadm ~]# dig ktcl1.kt.com

; <<>> DiG 9.7.3-RedHat-9.7.3-2.el6 <<>> ktcl1.kt.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34114
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;ktcl1.kt.com.                  IN      A

;; ANSWER SECTION:
ktcl1.kt.com.           86400   IN      A       192.168.10.91

;; AUTHORITY SECTION:
kt.com.                 86400   IN      NS      ktadm.kt.com.

;; ADDITIONAL SECTION:
ktadm.kt.com.           86400   IN      A       192.168.10.95
```

- **With IP address:**
  #dig –x 192.168.10.91

```
[root@ktadm ~]# dig -x 192.168.10.91

; <<>> DiG 9.7.3-RedHat-9.7.3-2.el6 <<>> -x 192.168.10.91
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20434
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

;; ANSWER SECTION:
91.10.168.192.in-addr.arpa. 86400 IN    PTR     ktcl1.kt.com.

;; AUTHORITY SECTION:
10.168.192.in-addr.arpa. 86400  IN      NS      ktadm.kt.com.

;; ADDITIONAL SECTION:
ktadm.kt.com.           86400   IN      A       192.168.10.95
```

Using ping to test the resolution

- **Try pinging with hostname both server and client**
  #ping –c2 ktadm
  #ping –c2 ktcl1 or any other client

```
[root@ktadm ~]# ping -c2 ktadm
PING ktadm.kt.com (192.168.10.95) 56(84) bytes of data.
64 bytes from ktadm.kt.com (192.168.10.95): icmp_seq=1 ttl=64 time=0.051 ms
64 bytes from ktadm.kt.com (192.168.10.95): icmp_seq=2 ttl=64 time=0.034 ms
```

```
[root@ktadm ~]# ping -c2 ktcl1
PING ktcl1.kt.com (192.168.10.91) 56(84) bytes of data.
64 bytes from ktcl1.kt.com (192.168.10.91): icmp_seq=1 ttl=64 time=0.601 ms
64 bytes from ktcl1.kt.com (192.168.10.91): icmp_seq=2 ttl=64 time=0.260 ms
```

Using host command to check resolution

- **Checking the DNS resolution with host command for both server as well as clients**
  #host <hostname>
  #host ktadm
  #host ktcl2

- **Using host command with IP address of server as well as client**
  #host 192.168.10.95
  #host 192.168.10.92 (or) 91, 93 any client

Using nslookup command to check the DNS resolution

- **Use nslookup command with server and clients hostname and check it**
  #nslookup ktadm
  #nslookup ktcl3
- **Check the same thing with IP addresses**
  #nslookup 192.168.10.95
  #nslookup 192.168.10.93

Client side configuration for DNS

- **Log into client machine and add the DNS server's information in /etc/resolv.conf file**
  #vim /etc/resolv.conf

```
# Generated by NetworkManager
search kt.com
nameserver 192.168.10.95
```

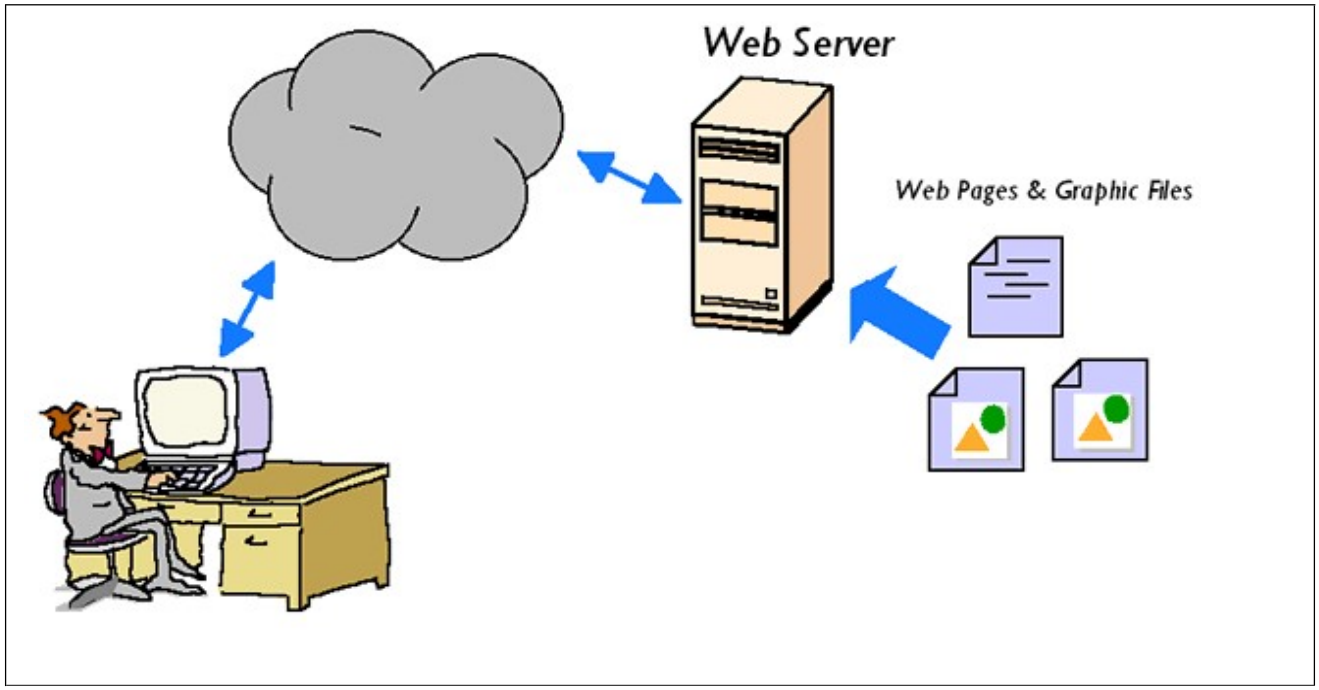
- **Now check with any of the options used previously like dig, ping, host or nslookup for dns resolution**
-

```
[root@ktcl1 ~]# nslookup ktadm
Server:         192.168.10.95
Address:        192.168.10.95#53

Name:   ktadm.kt.com
Address: 192.168.10.95

[root@ktcl1 ~]# nslookup ktcl3
Server:         192.168.10.95
Address:        192.168.10.95#53

Name:   ktcl3.kt.com
Address: 127.0.0.1
Name:   ktcl3.kt.com
Address: 192.168.10.93
```

```
[root@ktcl1 ~]# nslookup 192.168.10.95
Server:         192.168.10.95
Address:        192.168.10.95#53

95.10.168.192.in-addr.arpa      name = ktadm.kt.com.

[root@ktcl1 ~]# nslookup 192.168.10.93
Server:         192.168.10.95
Address:        192.168.10.95#53

93.10.168.192.in-addr.arpa      name = ktcl3.kt.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.

WEB SERVER (APACHE)



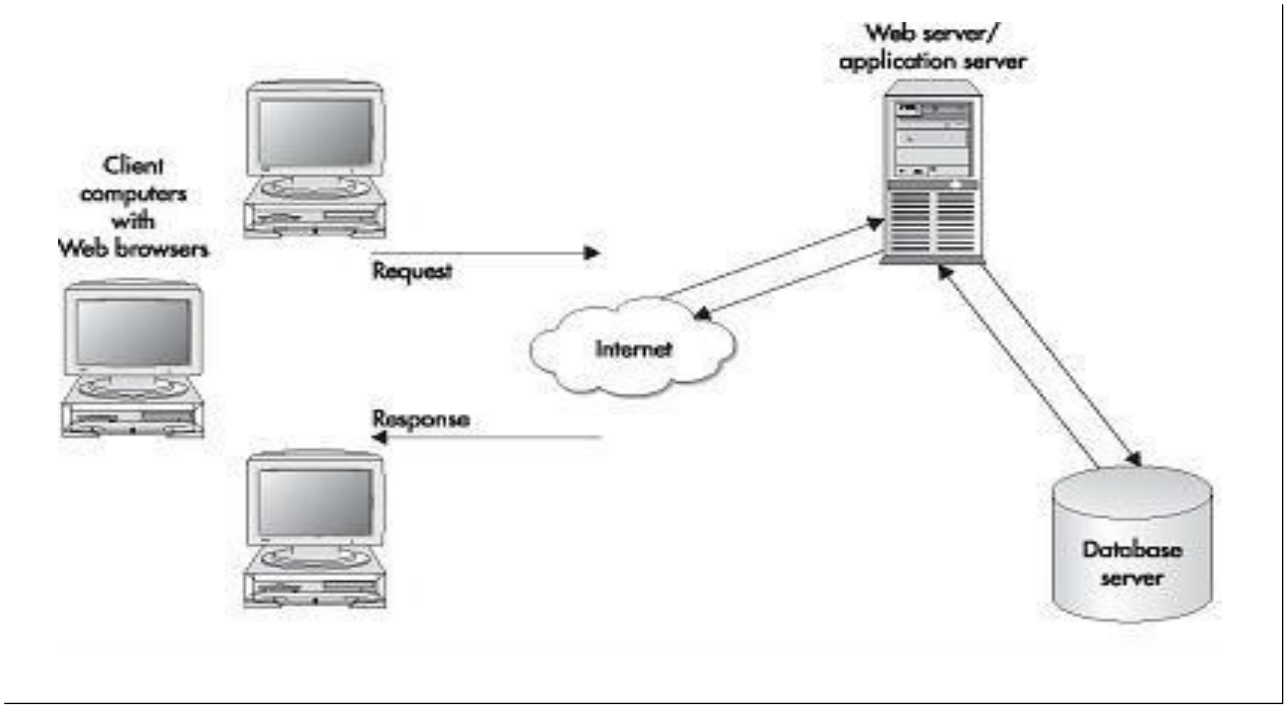
- Every Web site sits on a computer known as a Web server. This server is always connected to the internet. Web servers are computers that deliver (serves up) Web pages. Every Web server has an IP address and possibly a domain name.

- 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 in to 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 in to 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 in to HTML and displays the results to the user.

Profile for Apache Server

Use	**:**	**Hosting a web site.**
Package	**:**	**httpd**
Port	**:**	**80/http, 443/https**
Configuration Files	**:**	**/etc/httpd/conf/httpd.conf**
		/etc/httpd/conf.d/ssl.conf (https)
Document Root	**:**	**/var/www/html**
Daemon	**:**	**httpd**
Script	**:**	**/etc/initd/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@ktadm ~]# 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.el
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: Navigate to /etc/httpd/conf/httpd.conf and edit it.

- **Navigate to the configuration file for http i.e. /etc/httpd/conf/httpd.conf and copy the last 7 lines as shown below**
  #vim /etc/httpd/conf/httpd.conf

```
#<VirtualHost *:80>
#    ServerAdmin webmaster@dummy-host.example.com
#    DocumentRoot /www/docs/dummy-host.example.com
#    ServerName dummy-host.example.com
#    ErrorLog logs/dummy-host.example.com-error_log
#    CustomLog logs/dummy-host.example.com-access_log common
#</VirtualHost>
```

Copy these lines and paste it at the end of the page, then edit it with your preferences.

- **Edit the pasted lines as below**

```
<VirtualHost 192.168.10.95:80>
    ServerAdmin root@ktadm.kt.com
    DocumentRoot /var/www/html/
    ServerName ktadm.kt.com
    ErrorLog logs/ktadm.kt.com-error_log
    CustomLog logs/ktadm.kt.com-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> Kernel Technologies </h1>
###### Linux is Freedom ######
```

Step3: Restart the Service and enable it in boot configuration

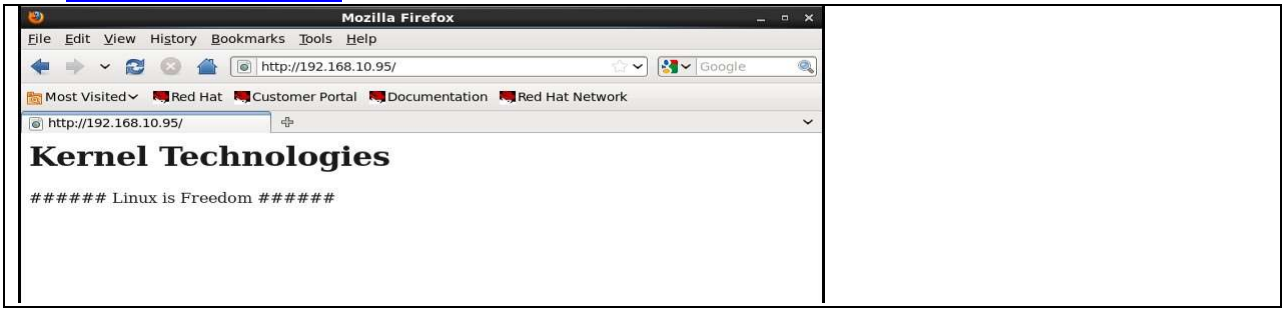
#service httpd restart

#chkconfig httpd on

```
[root@ktadm ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
[root@ktadm ~]# chkconfig httpd on
[root@ktadm ~]#
```

Step4: Now open the Firefox web browser and try connecting the web server.

- **Open Firefox web browser and type the IP Address of the web server**
  http://192.168.10.95



- **To open the website from command line use the following command**
  #curl <IP/HOSTNAME of web server>
  #curl 192.168.10.95

```
[root@ktadm ~]# curl 192.168.10.95
<h1> Kernel Technologies </h1>
###### Linux is Freedom ######
```

Also Try

#elinks --dump 192.168.10.95 and check the output

DNS configuration if you don't want to use IP address.

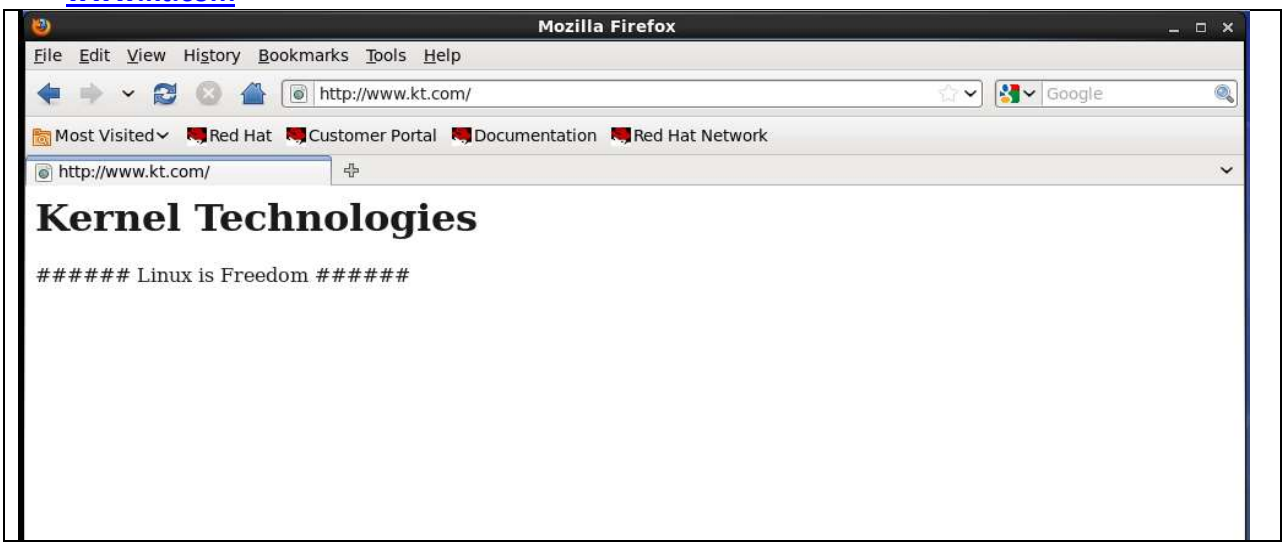
- **Open the DNS configuration file and add the canonical name as "www", so that we can use our domain as full fledge website.**
  #vim /var/named/kt.rlz

```
[root@ktadm ~]# vim /var/named/kt.flz
$TTL 86400
@       IN SOA  ktadm.kt.com.   root.kt.com. (
                                        201111171       ; serial
                                        1D              ; refresh
                                        1H              ; retry
                                        1W              ; expire
                                        3H )            ; minimum
        NS      ktadm.kt.com.
ktadm   A       192.168.10.95
ktcl1   A       192.168.10.91
ktcl2   A       192.168.10.92
ktcl3   A       192.168.10.93
www     CNAME   ktadm
```

- **Restart the DNS services**
  #service named restart

```
[root@ktadm port]# !ser
service named restart
Stopping named:                                            [  OK  ]
Starting named:                                            [  OK  ]
```

- **Okay now we are ready just open the web browser like Firefox and this type address as follows**
  www.kt.com



Note: This will only work in your DNS range, for others in your network use ip address

To create an Alias Web Site

- **Navigate to the document root i.e. /var/www/html/ and create a folder**

```
[root@ktadm ~]# cd /var/www/html/
[root@ktadm html]# mkdir vcs
[root@ktadm html]# cd vcs
[root@ktadm vcs]# vim index.html
```

- **Create an index.html**

```
<h1> Kernel Technologies </h1>
###### VCS On LINUX ######
```

- **Navigate to configuration file /etc/httpd/conf/httpd.conf and add a line as alias**
  #vim /etc/httpd/conf/httpd.conf

```
###################################
<VirtualHost 192.168.10.95:80>
    ServerAdmin root@ktadm.kt.com
    DocumentRoot /var/www/html/
    ServerName ktadm.kt.com
    alias       /vcs    /var/www/html/vcs
    ErrorLog logs/ktadm.kt.com-error_log
    CustomLog logs/ktadm.kt.com-access_log common
</VirtualHost>
```

- **Restart the service**

```
[root@ktadm ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

- **Open the Firefox web browser and type the following url**
  http://192.168.10.95/vcs

To redirect the website:

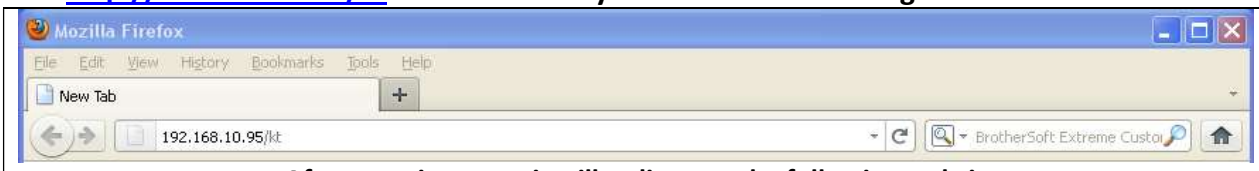
- **Redirecting means whenever the name of a particular website is given it should take us on some other website.**
- **To redirect a website, navigate and open the configuration file of http i.e. /etc/httpd/conf/httpd.conf and add the following line in it at the end.**
  #vim /etc/httpd/conf/httpd.conf

```
####################################
<VirtualHost *:80>
    ServerAdmin root.kt.com
    DocumentRoot /var/www/html/
    ServerName ktadm.kt.com
    Alias    /vcs  /var/www/html/vcs
    ErrorLog logs/ktadm.kt.com-error_log
    CustomLog logs/dummy-ktadm.kt.com-access log common
    Redirect  /kt "http://www.kerrneltech.com"
</VirtualHost>
```

- **Restart the services**
  #service http restart

```
[root@ktadm ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

- **Open the Firefox browser and type the following website address http://192.168.10.95/kt and it will take you to Kernel Technologies' website**



After pressing enter it will redirect to the following website

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@ktadm ~]# cd /var/www/port/
[root@ktadm port]# vim index.html
<h1> PORT BASED WEB HOSTING </h1>
###### Welcome to Kernel Tech #######
```

Step3: edit the configuration file i.e. /etc/httpd/conf/httpd.conf add the configuration for port based hosting in configuration file.

#vim /etc/httpd/conf/httpd.conf. Copy the same 7 lines and paste it at end edit it

```
########### Port Based Hosting #############
<VirtualHost 192.168.10.95:8080>
    ServerAdmin root@kt.com
    DocumentRoot /var/www/port/
    ServerName ktadm.kt.com
    ErrorLog logs/ktadm.kt.com-error_log
    CustomLog logs/dummy-ktadm.kt.com-access_log common
</VirtualHost>
```

- **Also search for the "Listen 80" by using "/" and paste your port under it**

```
#Listen 12.34.56.78:80
Listen 80
Listen 8080
```

Step4: Restart the service, open web browser and search for your website with port no.
#service httpd restart

```
[root@ktadm port]# !ser
service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
```

Open Firefox and type http://192.168.10.95:8080

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

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/ktname

```
[root@ktadm ~]# mkdir /var/www/ktname
```

- **Add an index page in it**
  #vim /etc/var/www/ktname/index.html

```
<h1> NAME BASED WEB HOSTING </h1>
###### Welcome to Kernel Tech #######
```

Step2: Edit the configuration file as shown below

- **Open the configuration file and add the name host information , search for NameVirtualHost and your host details**
  #vim /etc/httpd/conf/httdp.conf

```
# Use name-based virtual hosting.
#
#NameVirtualHost *:80
NameVirtualHost 192.168.10.95:80
#
# NOTE: NameVirtualHost cannot be used without a port specifier
# (e.g. :80) if mod_ssl is being used, due to the nature of the
# SSL protocol.
#
```

- **Copy the same <VirtualHost> 7 lines, paste it at the last of the page and edit it as follows**

```
########### Name  Based Hosting #############
<VirtualHost 192.168.10.95:80>
    ServerAdmin root@kt.com
    DocumentRoot /var/www/ktname/
    ServerName ktadm.kt.com
    ErrorLog logs/ktadm.kt.com-error_log
    CustomLog logs/dummy-ktadm.kt.com-access_log common
</VirtualHost>
```

Step3: Restart the server and open the web page from Firefox
#service httpd restart

```
[root@ktadm ~]# service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
[root@ktadm ~]#
```

http://192.168.10.95 now it will navigate to our name based web page

KICKSTART
AND
NETWORK INSTALLATIONS OF RHEL6

- 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 by hand or using the GUI system-config-kickstart tool. 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).

   To create a kickstart file using GUI, first install the package *"system-config-kickstart"*

Check and install package for kickstart

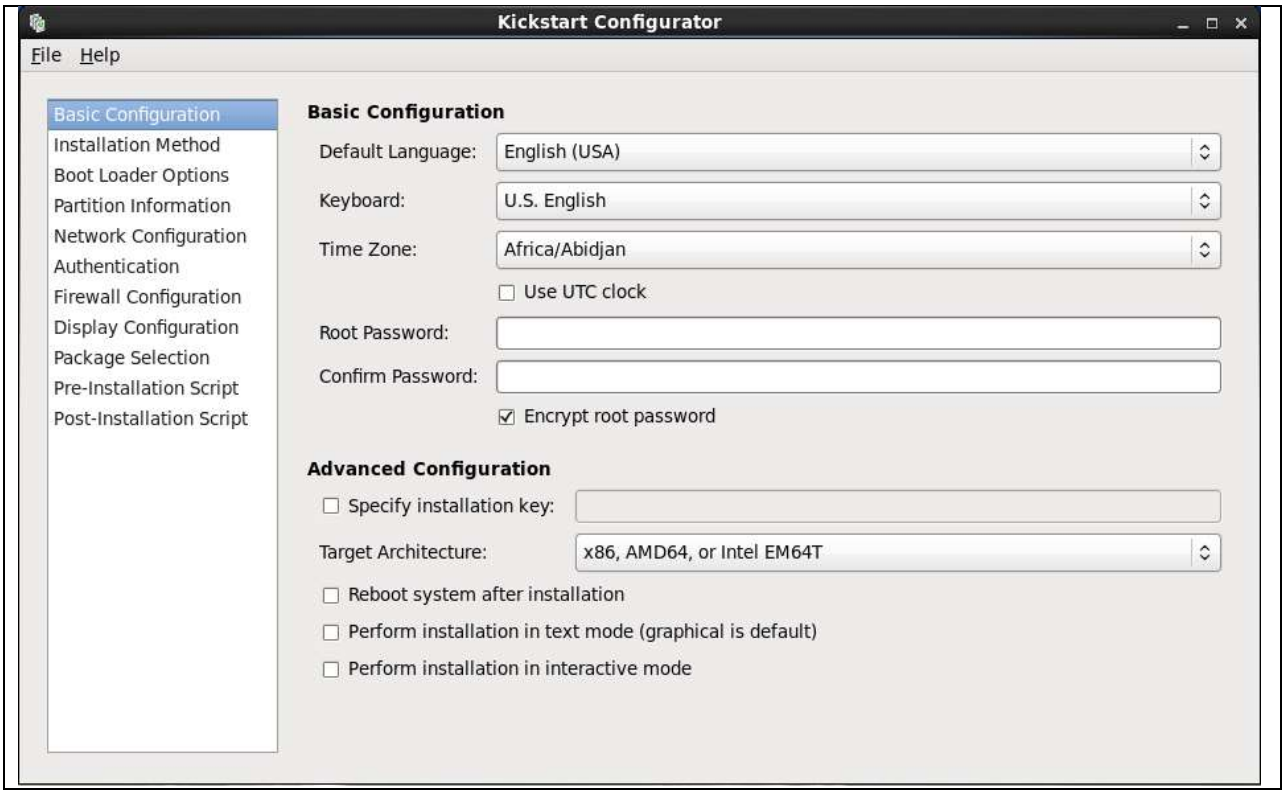
- **#rpm –q system-config-kickstart**

```
[root@localhost Desktop]# rpm -q system-config-kickstart
package system-config-kickstart is not installed
```

- Install the package using yum or rpm command
  #yum install system-config-kickstart

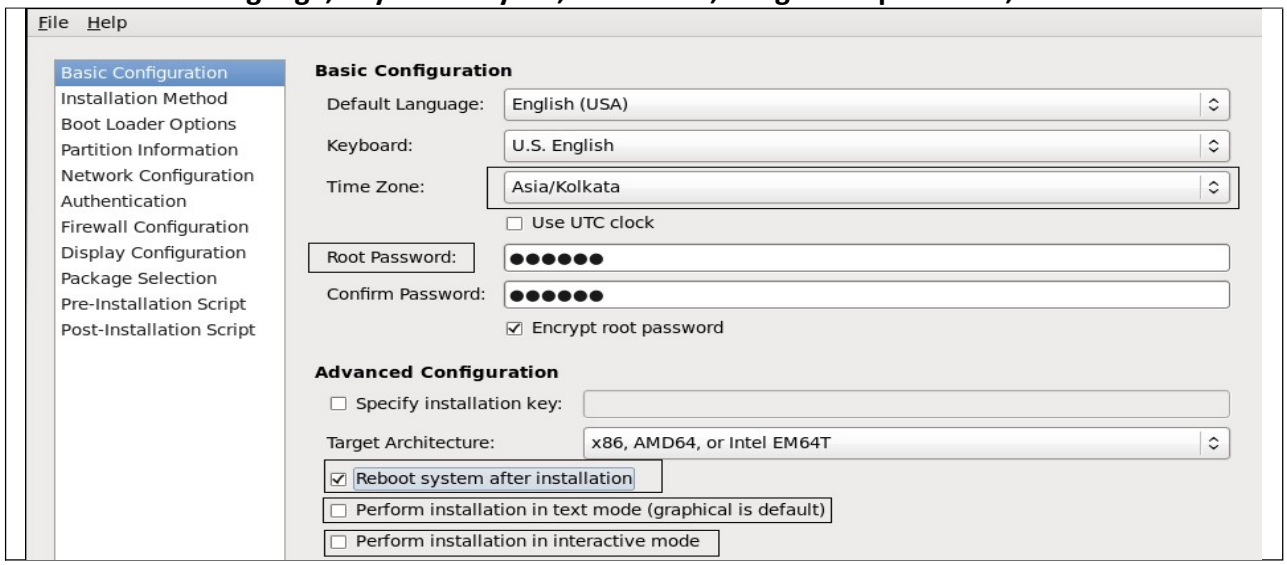
```
[root@ktcl2 ~]# yum install system-config-kickstart -y
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
KTREPO                                                   | 3.7 kB     00:00
Setting up Install Process
^@Resolving Dependencies
--> Running transaction check
---> Package system-config-kickstart.noarch 0:2.8.6.2-1.el6 set to be updated
--> Processing Dependency: pykickstart >= 0.96 for package: system-config-kickst
```

- **Once it is installed use "*system-config-kickstart"* command to create a kickstart file.**
  #system-config-kickstart

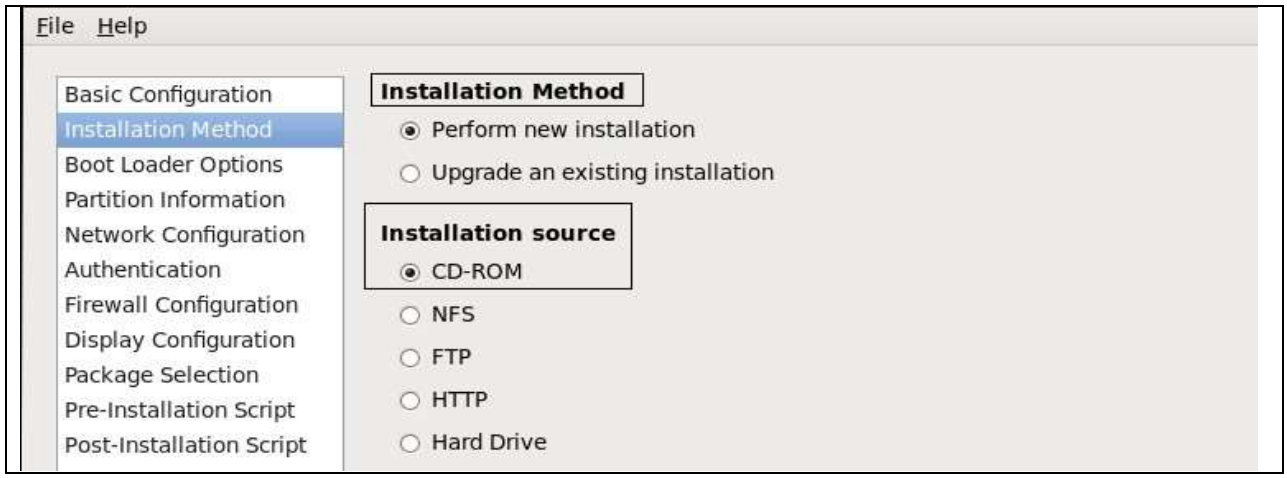


- **Let's see each option of kickstart file and create a new kickstart file.**

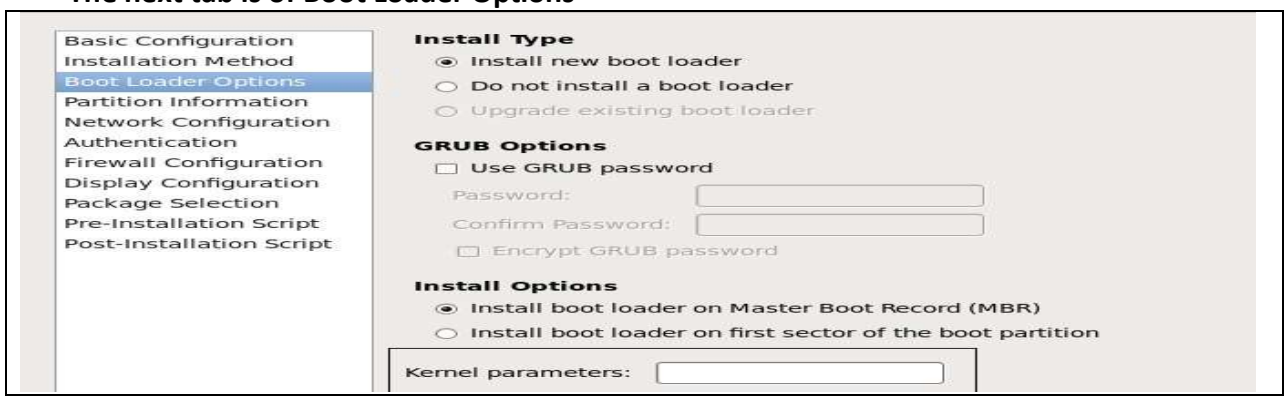
- **The first option in kickstart is basic configuration, Select the options required as below**
- **Select the language, keyboard layout, Time Zone, assign root password, etc**



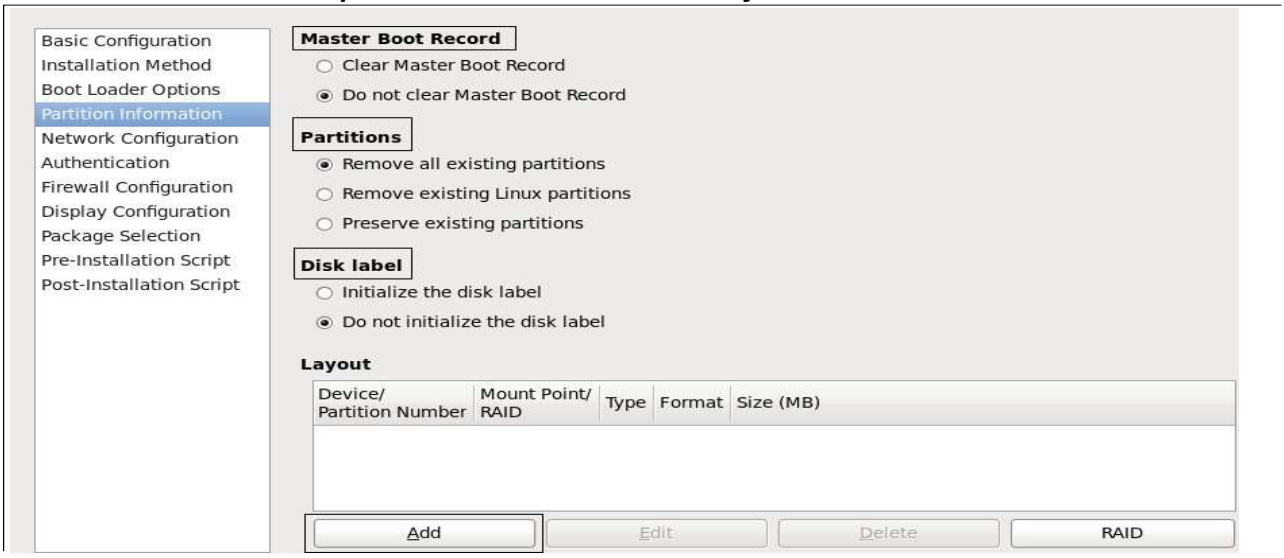
- **The second tab is installation Method. For normal installation select CD-ROM**



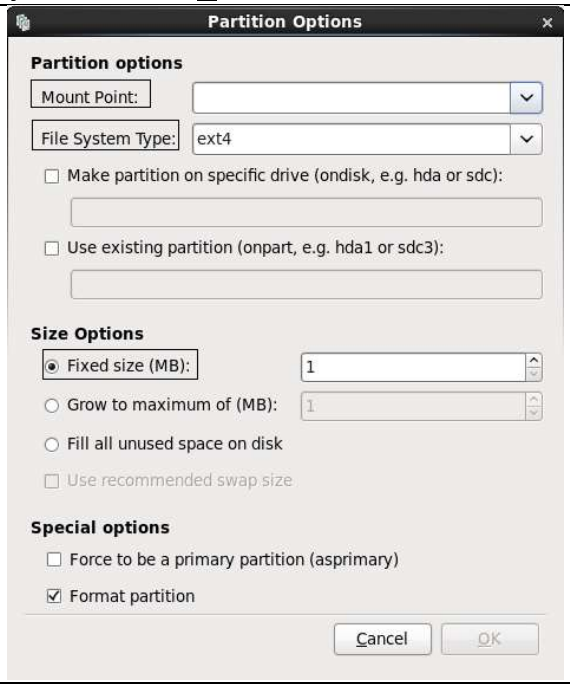
- **The next tab is of Boot Loader Options**



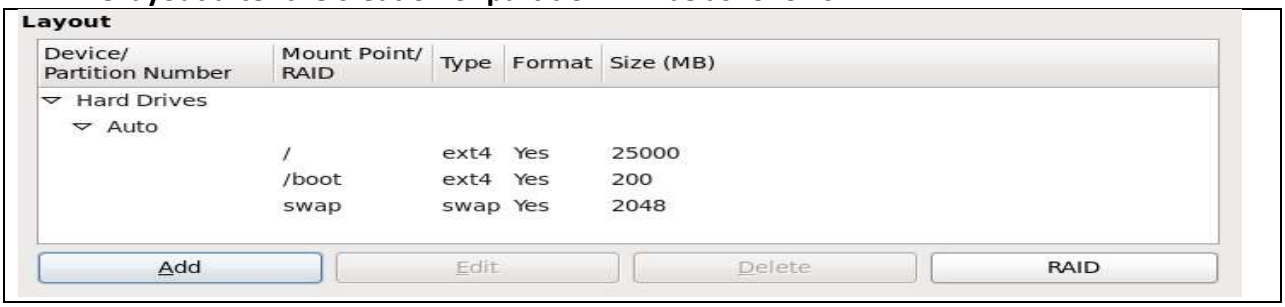
- **The next tab is an important tab called *"Partition Information"***



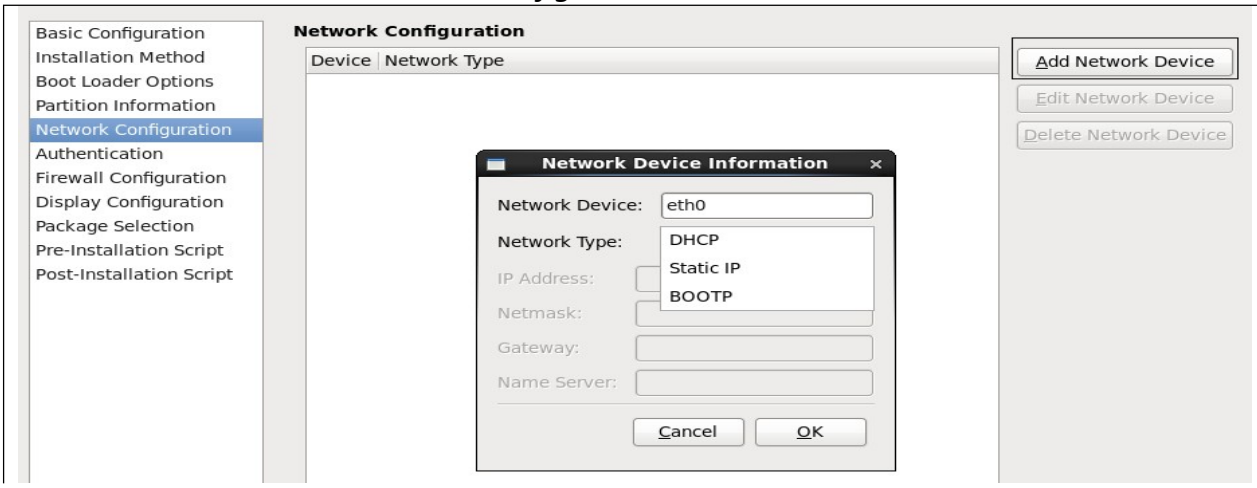
- **To create partitions layout, click on Add button and follow the screens below**



- **Select the mount points like /, /boot, /opt etc and create some partitions as usual**
- **The layout after the creation of partition will be as follows**

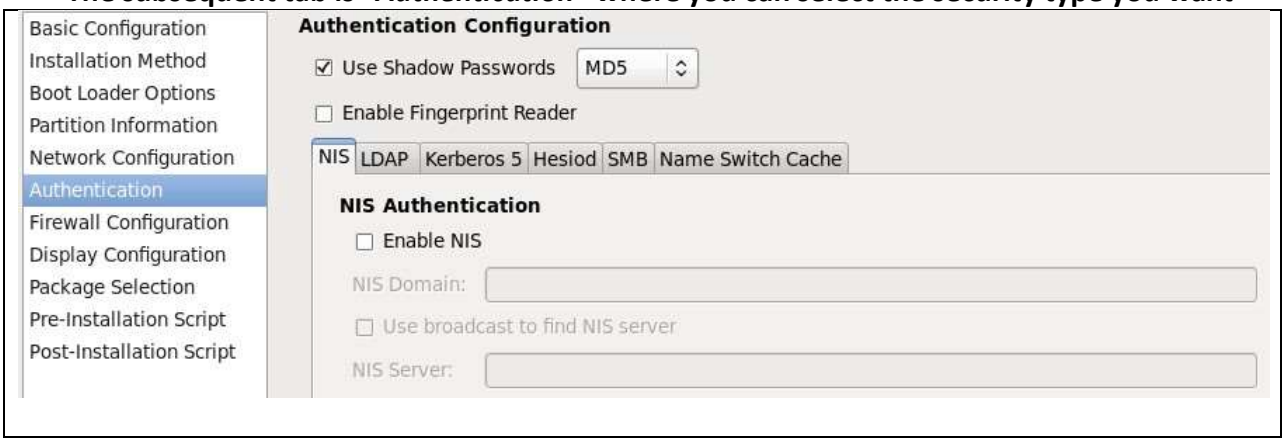


- **The next tab is about *"Network Configuration"***



- **Click on "*Add Network Device*" and add an NIC adapter as *"eth0"*.**
- **Select any of Network Type from list, but if selected Static IP, then you need to specify the IP address, net mask and other attributes to it.**

- **The subsequent tab is *"Authentication"* where you can select the security type you want**



- **After "*Authentication*" the next tab is "*Firewall Configuration",* where we can configure some firewall settings and SELinux settings.**

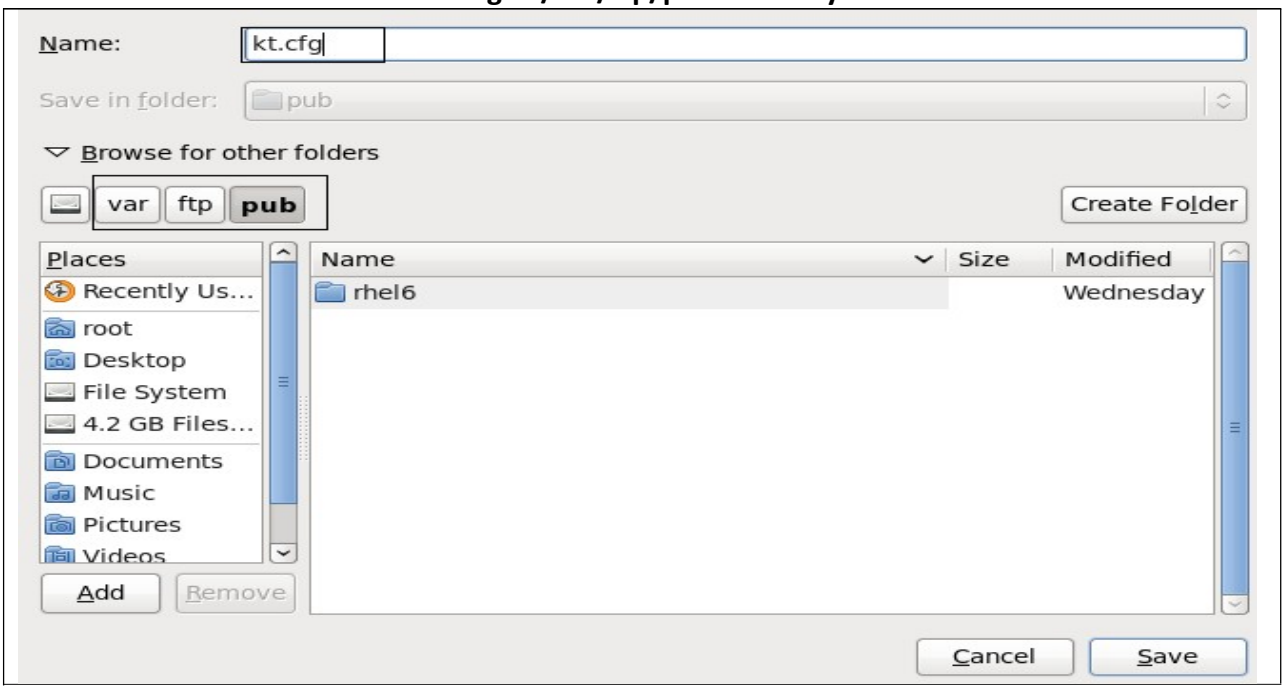
***Note:** If Firewall is enabled assign some services which are allowed in it.*

- ***"Display Configuration"*** **Tab for selecting display environment of the O/S**

- **The next tab after above is *"Package Selection"* where you can select various packages that will be installed with O/S**

- **After Package selection if you want to run any scripts pre and post- installation, you can go for remaining two tabs, otherwise leave it.**
- **Finally save the kickstart file in ftp's document root, so that it can be accessible from any machine in the network**

- **Save it with some name with .cfg in /var/ftp/pub directory**



- **Verify it in document root of ftp whether it is created or not**

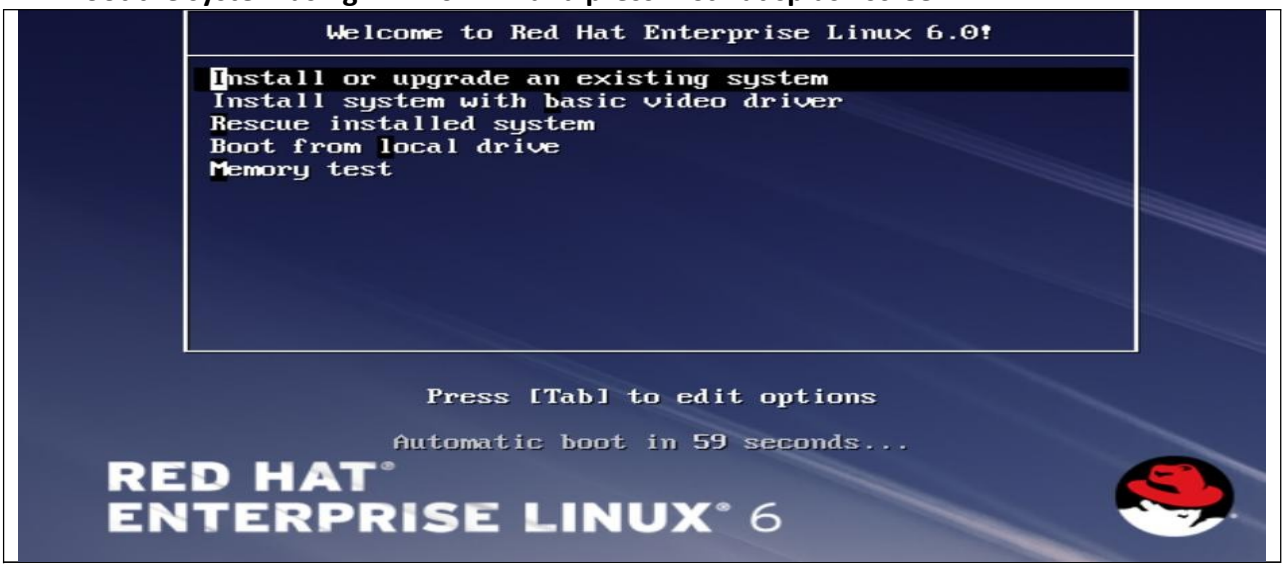
```
[root@ktadm pub]# ls
kt.cfg  rhel6
[root@ktadm pub]#
```

- **To modify the same, open system-config-kickstart application go to *File* -> *Open* and navigate to the document root of ftp and select the file to be opened.**

- **Once the kickstart file is opened modify it as per the requirement**

Client side operation for kickstart

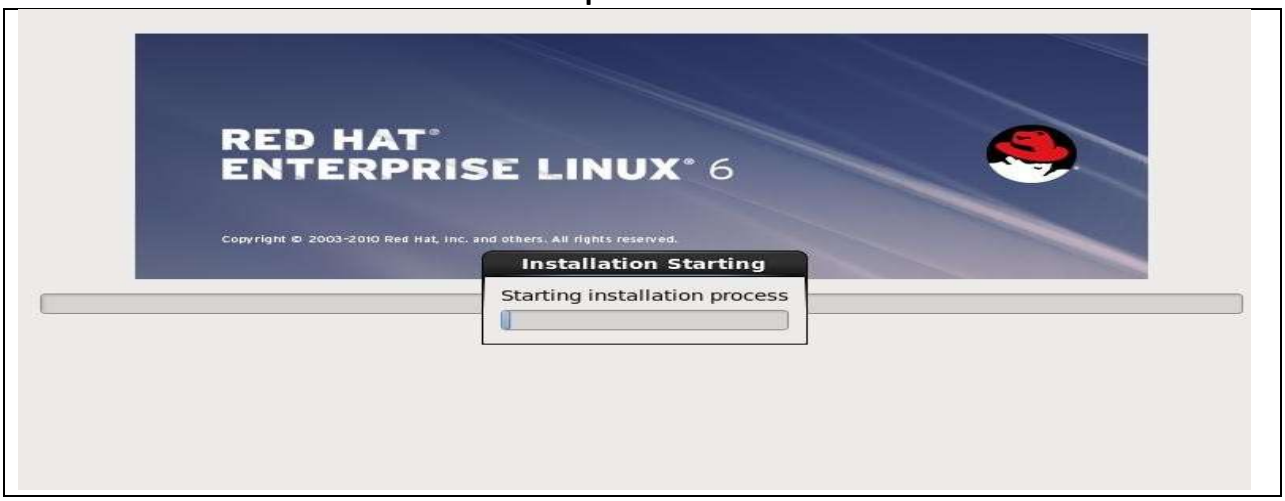
- **Boot the system using RHEL 6 DVD and press *"Esc"* at splash screen**



- **After pressing "*Esc*" , 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.**
  # linux ip=192.168.10.96 netmask=255.255.255.0 ks=ftp://192.168.10.95/pub/kt.cfg

- **After entering above information just 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

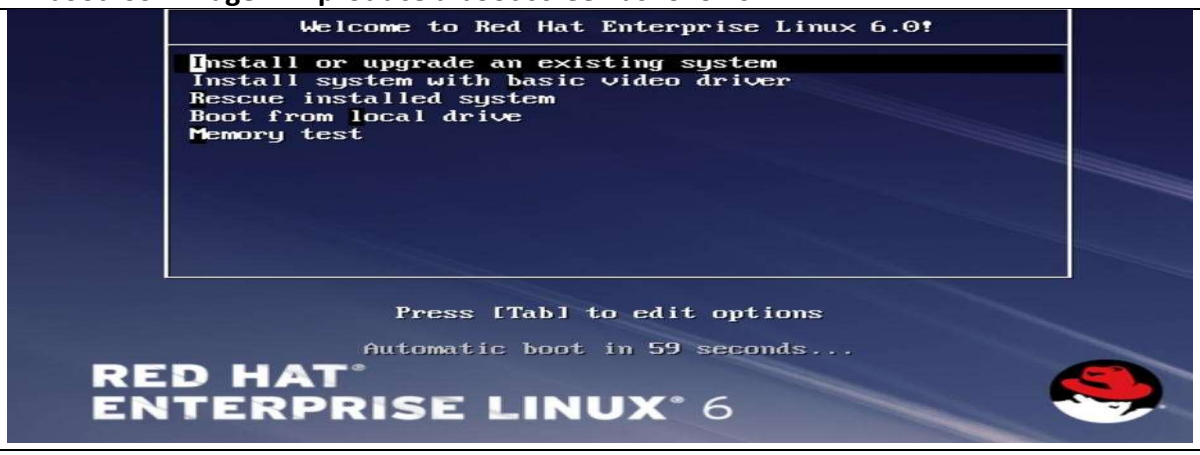
Steps to perform installation using FTP:

- Copy the entire RHEL6 DVD on document root of ftp i.e. /var/ftp/pub/

```
[root@ktadm ~]# cd /var/ftp/pub
[root@ktadm pub]# ls
kt.cfg  kts.cfg  rhel6
[root@ktadm pub]#
```

- Start installation using FTP directory

- **As we are trying to install RHEL6 from network still we require a boot media so that at least we can get the boot screen where we can type our 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)

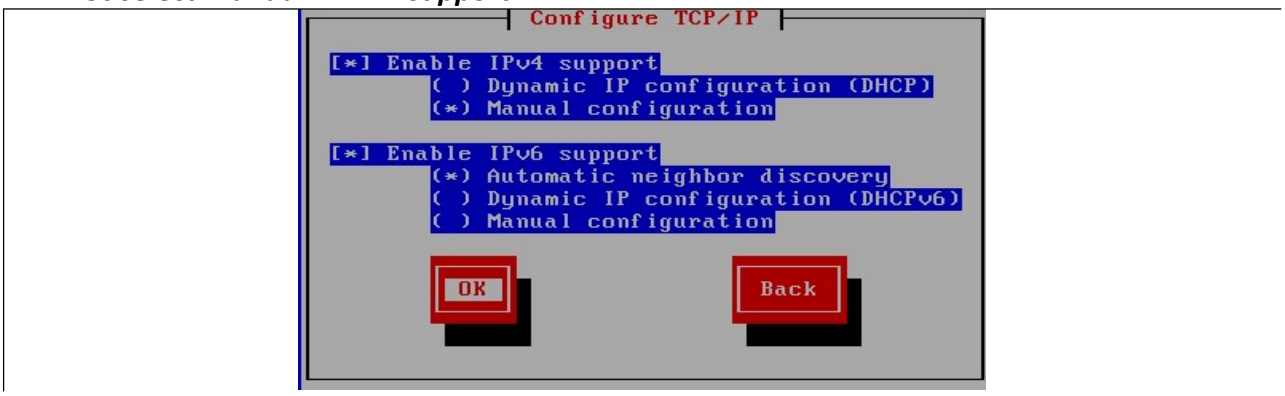
- **After making the boot media, make the system boot with it, press *"Esc"* to type the following command to take installation media from network**
  # boot: linux askmethod

- **After a while the following prompt will be display where you can select the preferred language**

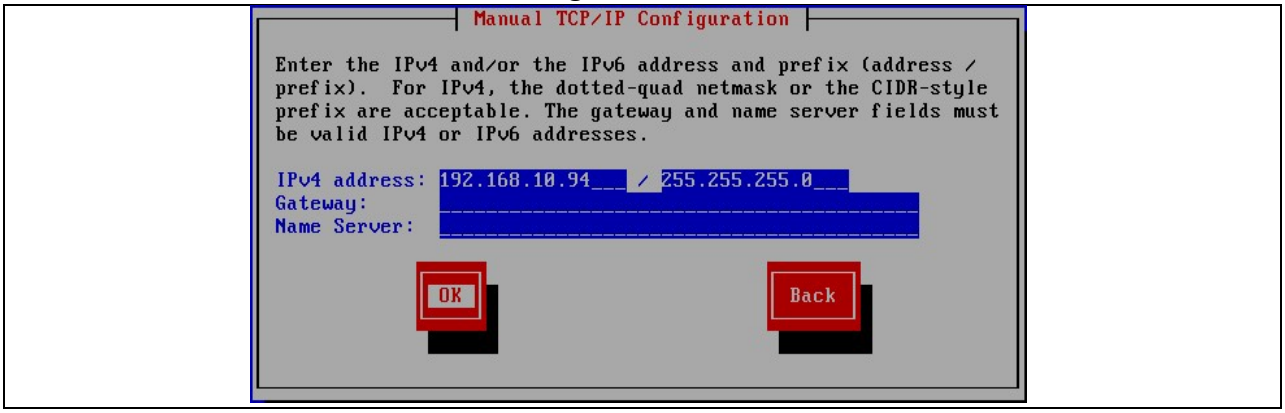
- **Select the required keyboard layout**

- **It is the time now to select the medium from which you want to install the O/S**

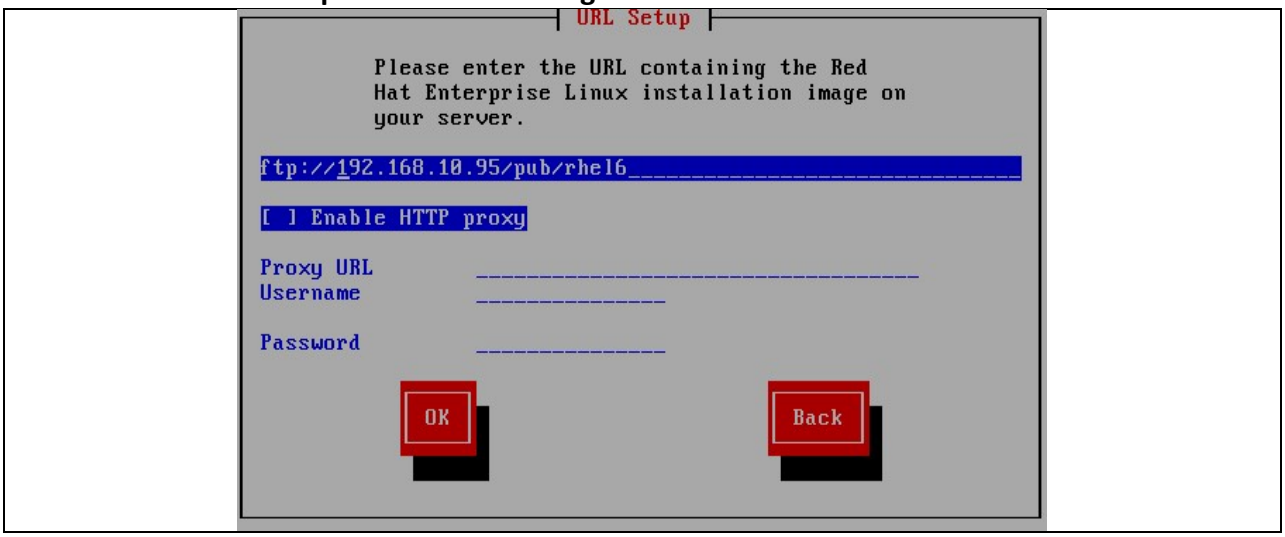
- **Select *URL* for *ftp* and *http,* whereas *NFS* for installing from NFS**
- **Define the network settings, if DHCP is configured in your environment select *Dynamic* if not select *Manual in iPv4 support***



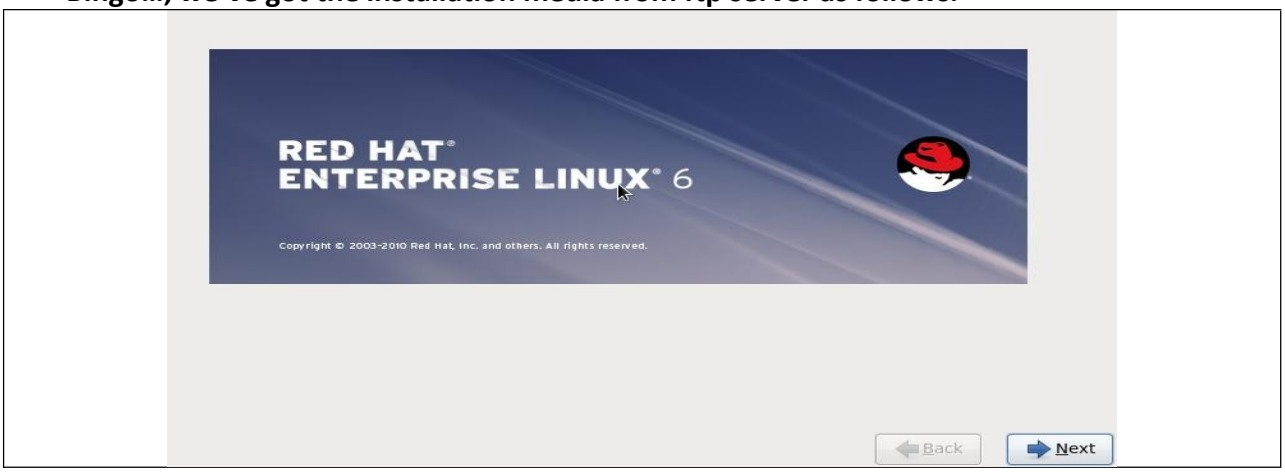
- **Assign some IP address to your machine so that it can communicate with the server. Make sure that IP should be in the same range that of the server.**



- **Once the network is configured automatically the following screen will appear, provide the details of the ftp server as following**



- **Bingo…, we've got the installation media from ftp server as follows.**



Now you can install as usual from here.

Configuring NFS and http servers for network installations

NFS Configuration

- Make an entry in ***/etc/exports*** to export the **RHEL6 media.**
- Let us say my RHEL6 DVD is dumped in ***/var/ftp/pub/rhel6*** directory

```
/var/ftp/pub/rhel6  192.168.10.0/24(rw,sync)
```

- Use the ***exportfs*** command to export the directory.

```
[root@ktadm ~]# exportfs -rv
exporting 192.168.10.0/24:/var/ftp/pub/rhel6
```

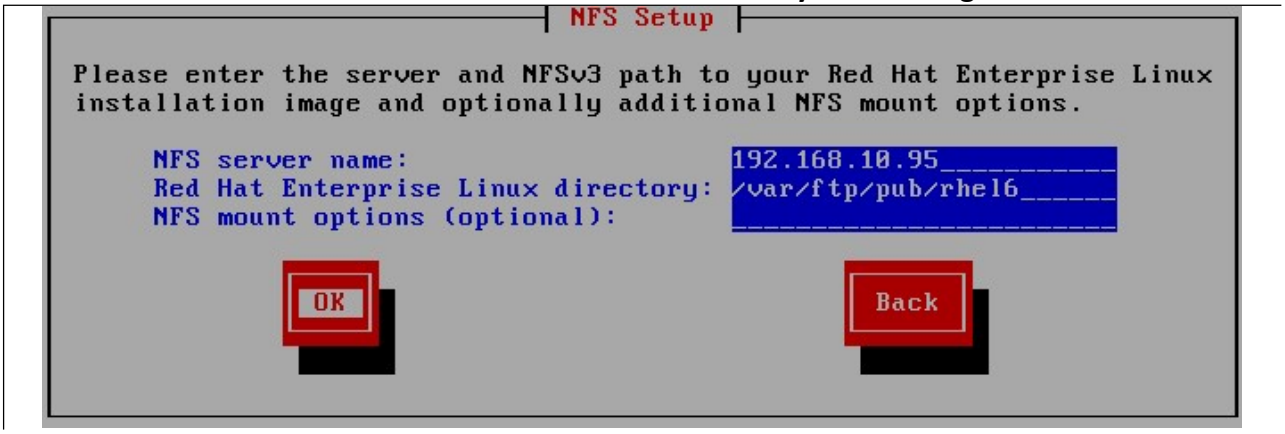
Don't forget to restart the services of *NFS*

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 directory instead of ***URL***

- **Give the information about the NFS server and directory as following**



That's it; your installation will be started from NFS server

<u>HTTP Configuration for network installations</u>

- Copy the **RHEL6 DVD** dump in the document root of http i.e., ***/var/www/html*** , else just create a soft link of the directory in the document root of http.

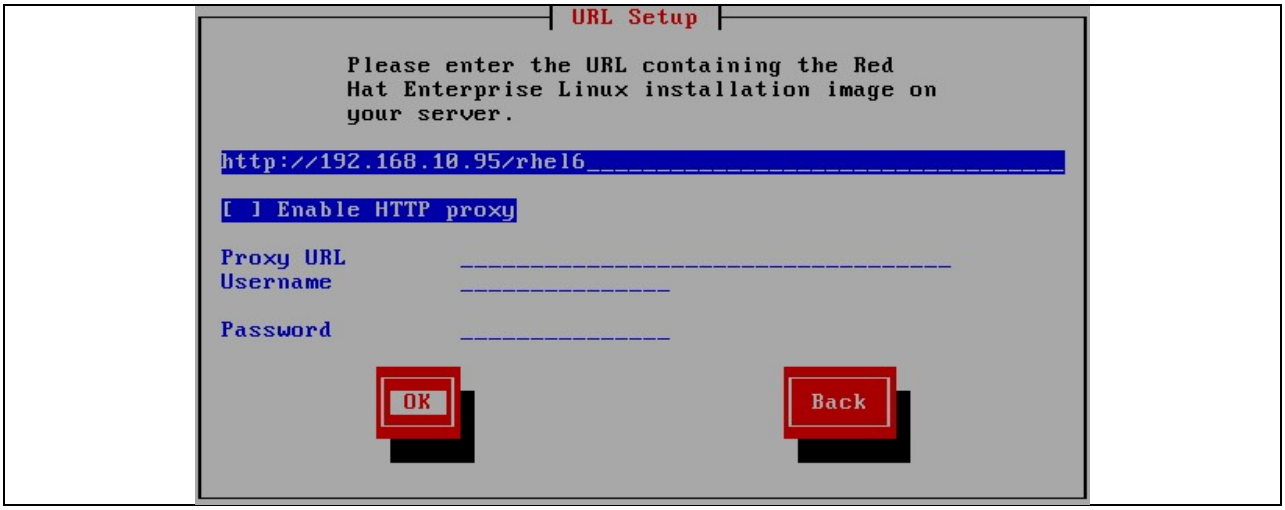
```
ln -s /var/ftp/pub/rhel6 /var/www/html/rhel6
```

```
[root@ktadm ~]# cd /var/www/html/
[root@ktadm html]# ls
index.html  rhel6
[root@ktadm html]#
```

- ***Restart the services of http and you are done with the server side configuration***

<u>Client side setup</u>

- **Repeat all the steps as done for *FTP* installation, the only change would be In the *URL* address which will as below**



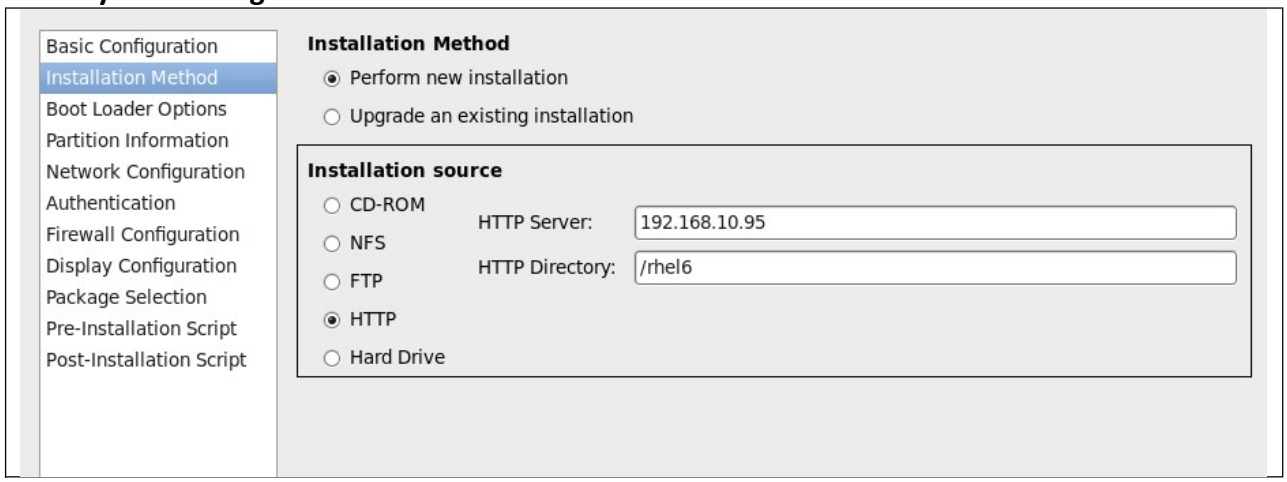
Once continued with *OK* installation will be started through http

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 installation predefined

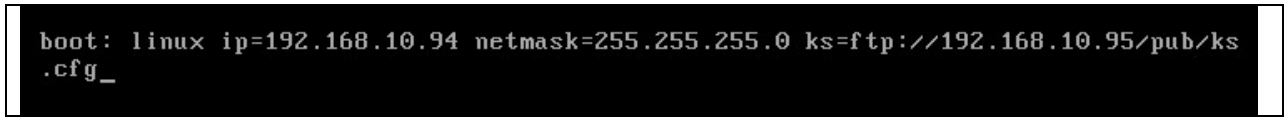
- Create a kickstart file as usual, the only change we need to do is in method of installation.
  #system-config-kickstart



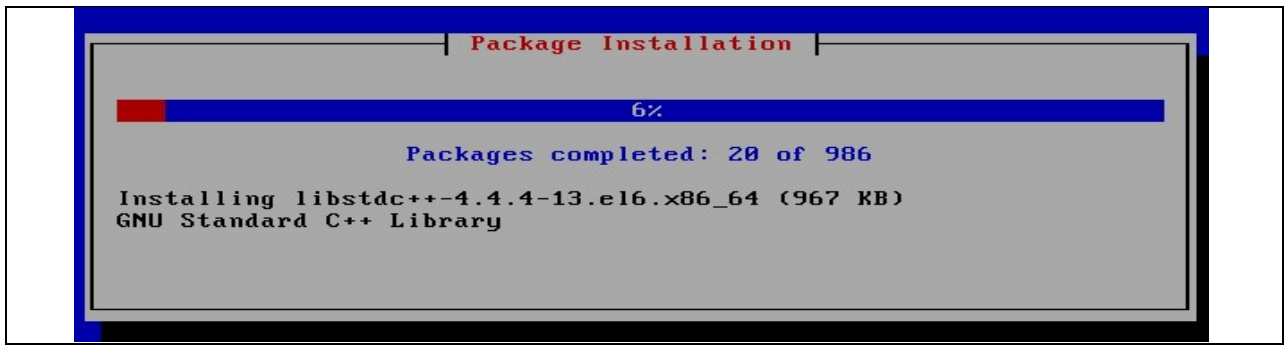
- We can select any method of installation from the list and specify the details regarding the server.
- If using ftp to access the kickstart file save it in document root of ftp

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.



Isn't it Amazing….!