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

  E.g.: MS-DOS

- Single User : Multitasking Operating System

  E.g.: Windows -98,Xp,vista,Seven etc.

- Multi User : Multitasking Operating System

  E.g.: UNIX, LINUX etc.

HISTORY OF UNIX :

- 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.

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

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

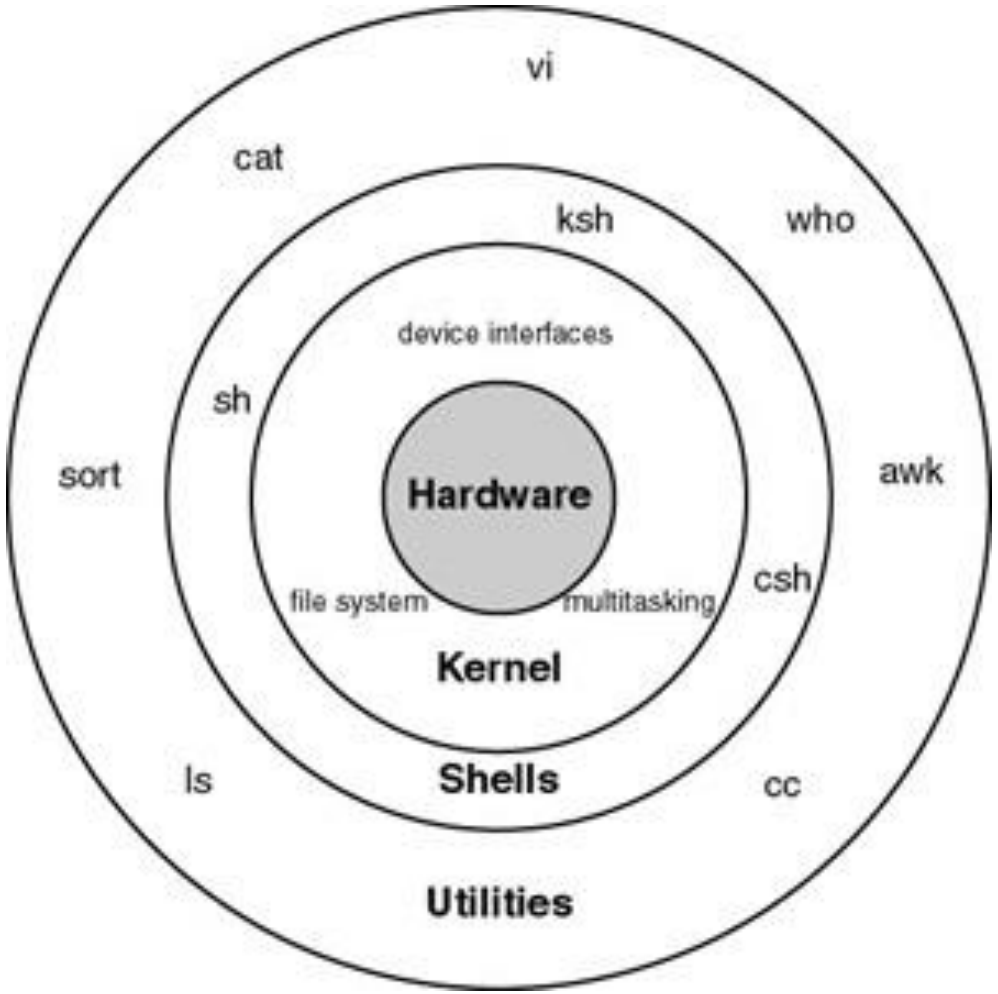
GNU Project/ FSF :

- 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

Why Linux?

- Open source development model
- Multi-User and Multi-tasking
- Supports wide variety of hardware
- Fast , Secure & Privacy
- Supports many networking protocols and Configurations
- Fully supported
- Better Community Support

ARCHITECTURE OF UNIX

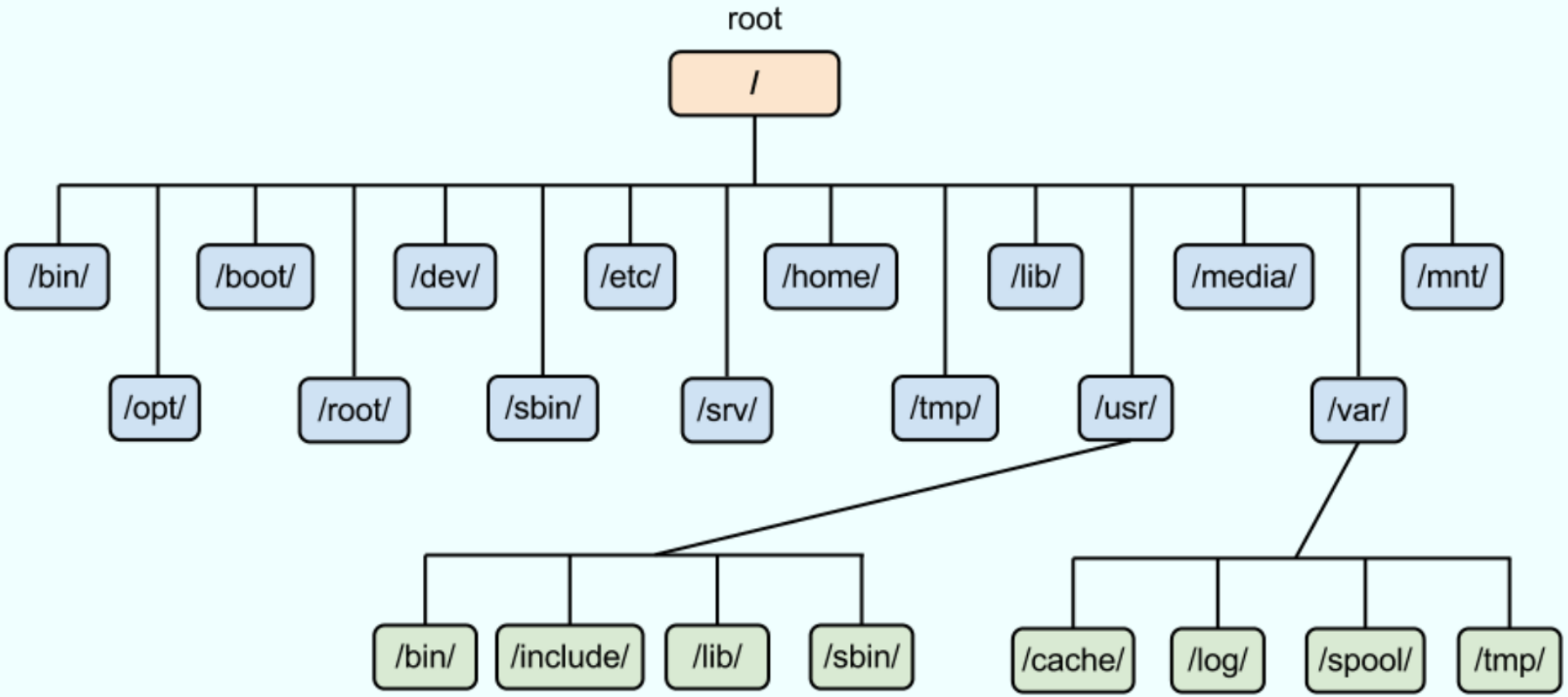


Kernel : which schedules tasks , manages resources, and controls security.

Shell : acts as the user interface, interpreting user commands and starting applications.

Utilities : which provides utility functions. In other words it is the USER level

Linux Directory Structure (File System Structure) ?



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

Virtual Box Installation

- Open the below url and download Windows package https://www.virtualbox.org/wiki/Downloads
- Try to run it by double clicking.

Download the Ubuntu Inamage

https://ubuntu.com/download/desktop

Follow the below link to complete the Ubuntu Installation on Virtual Box

https://www.codeooze.com/windows-10/windows-10-ubuntu-vbox/

- Basic Linux Commands

    - Creating, Removing, Copying, Moving files & Directories

    - VIM Editor

    - Links (Soft link and Hard Link)

    - Regular Expressions, Pipelines & I/O Redirections

    - Filter Commands

    - Umask

    - Processes, Memory and Disk Management

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:

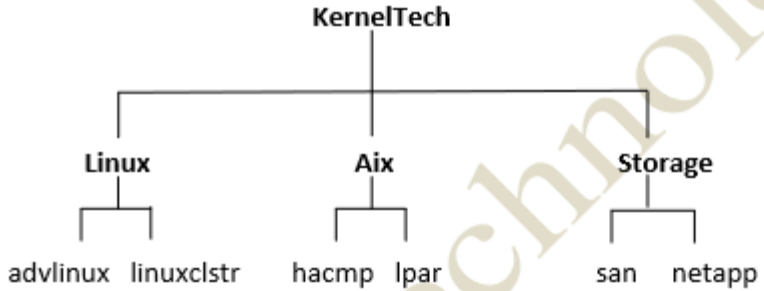
#mkdir <dir name>

#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
```

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
```

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**

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

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 ~]#
```

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)

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 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

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
```

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
```

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
```

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
```

cut command in Linux with examples

The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by **byte position, character and field**. Basically the cut command slices a line and extracts the text. It is necessary to specify option with command otherwise it gives error. If more than one file name is provided then data from each file is **not precedes** by its file name.

Syntax:

```
cut OPTION... [FILE]...
```

Let us consider two files having name **state.txt** and **capital.txt** contains 5 names of the Indian states and capitals respectively.

```
$ cat state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
```

Options and their Description with examples:

1. -b(byte): To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma. Range of bytes can also be specified using the hyphen(-). It is necessary to specify list of byte numbers otherwise it gives error. **Tabs and backspaces** are treated like as a character of 1 byte.

```
List without ranges
$ cut -b 1,2,3 state.txt
And
Aru
Ass
Bih
Chh

List with ranges
$ cut -b 1-3,5-7 state.txt
Andra
Aruach
Assm
Bihr
Chhtti
```

It uses a special form for selecting bytes from beginning upto the end of the line:

```
In this, 1- indicate from 1st byte to end byte of a line
$ cut -b 1- state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh

In this, -3 indicate from 1st byte to 3rd byte of a line
$ cut -b -3 state.txt
And
Aru
Ass
Bih
Chh
```

```
$cut -d "delimiter" -f (field number) file.txt
```

Like in the file **state.txt** fields are separated by space if -d option is not used then it prints whole line:

```
$ cut -f 1 state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
```

If -d option is used then it considered space as a field separator or delimiter:

```
$ cut -d " " -f 1 state.txt
Andhra
Arunachal
Assam
Bihar
Chhattisgarh
```

```
Command prints field from first to fourth of each line from the file.
Command:
$ cut -d " " -f 1-4 state.txt
Output:
Andhra Pradesh
```

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
```

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
```

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.

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.

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.

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 ~]#
```

```
ps aux | grep chrome
```

The *aux* options are as follows:

- a = show processes for all users
- u = display the process's user/owner
- x = also show processes not attached to a terminal

The *x* option is important when you're hunting for information regarding a graphical application.

Signal Name	Signal Value	Behaviour
SIGHUP	1	Hangup
SIGKILL	9	Kill Signal
SIGTERM	15	Terminate

The common syntax for **kill command** is:

```
# kill [signal or option] PID(s)
```

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.**

Memory Usage

On linux, there are commands for almost everything, because the gui might not be always available. When working on servers only shell access is available and everything has to be done from these commands. So today we shall be checking the commands that can be used to check memory usage on a linux system. Memory include RAM and swap.

1. free command

The free command is the most simple and easy to use command to check memory usage on linux. Here is a quick example

```
$ free -m
             total       used       free     shared    buffers     cached
Mem:          7976       6459       1517          0        865       2248
-/+ buffers/cache:       3344       4631
Swap:         1951          0       1951
```

2. /proc/meminfo

The next way to check memory usage is to read the /proc/meminfo file. Know that the /proc file system does not contain real files. They are rather virtual files that contain dynamic information about the kernel and the system.

```
$ cat /proc/meminfo
MemTotal:         8167848 kB
MemFree:          1409696 kB
Buffers:           961452 kB
Cached:           2347236 kB
SwapCached:             0 kB
Active:           3124752 kB
Inactive:         2781308 kB
Active(anon):     2603376 kB
Inactive(anon):    309056 kB
Active(file):      521376 kB
Inactive(file):   2472252 kB
```

3. vmstat

The vmstat command with the s option, lays out the memory usage statistics much like the proc command. Here is an example

```
$ vmstat -s
       8167848 K total memory
       7449376 K used memory
       3423872 K active memory
       3140312 K inactive memory
        718472 K free memory
       1154464 K buffer memory
       2422876 K swap cache
       1998844 K total swap
```

4. top command

The top command is generally used to check memory and cpu usage per process. However it also reports total memory usage and can be used to monitor the total RAM usage. The header on output has the required information. Here is a sample output

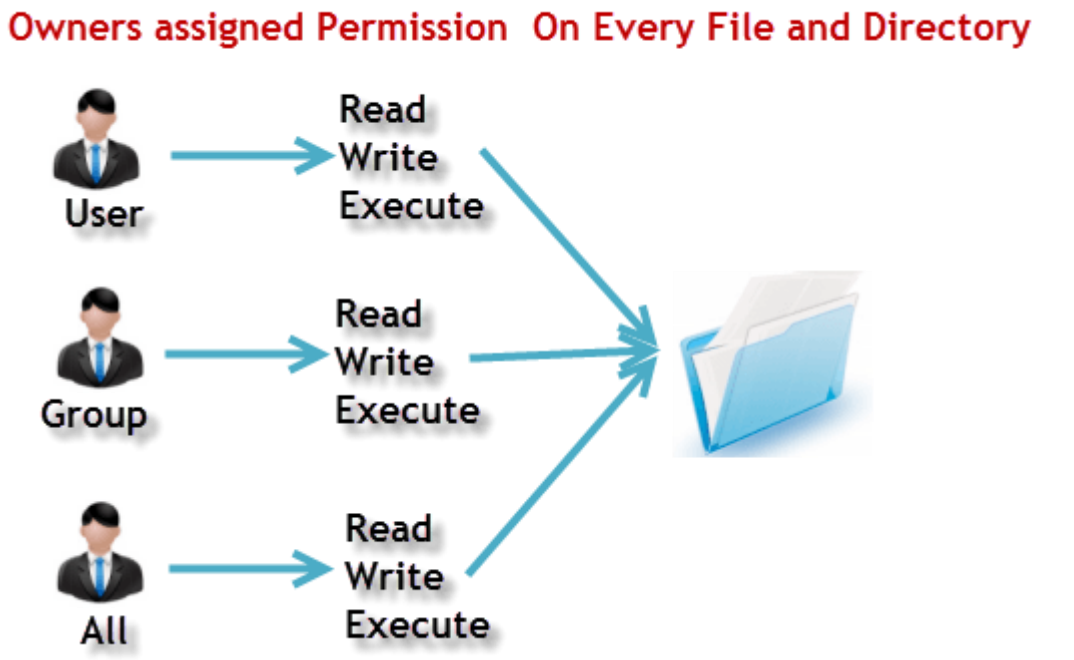
```
top - 15:20:30 up  6:57,  5 users,  load average: 0.64, 0.44, 0.33
Tasks: 265 total,   1 running, 263 sleeping,   0 stopped,   1 zombie
%Cpu(s):  7.8 us,  2.4 sy,  0.0 ni, 88.9 id,  0.9 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   8167848 total,  6642360 used,  1525488 free,  1026876 buffers
KiB Swap:  1998844 total,        0 used,  1998844 free,  2138148 cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 2986 enlighte  20   0  584m  42m  26m S  14.3  0.5   0:44.27 yakuake
 1305 root      20   0  448m  68m  39m S   5.0  0.9   3:33.98 Xorg
 7701 enlighte  20   0  424m  17m  10m S   4.0  0.2   0:00.12 kio_thumbnail
```

File Permissions in Linux/Unix :

Linux is a clone of UNIX, the **multi-user operating system** which can be accessed by many users simultaneously. Linux can also be used in mainframes and servers without any modifications. But this raises security concerns as an unsolicited or **malign user** can **corrupt, change or remove crucial data**. For effective security, Linux divides authorization into 2 levels.

1. Ownership
2. Permission



Ownership of Linux files

Every file and directory on your Unix/Linux system is assigned 3 types of owner, given below.

User

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.

Group

A user- group can contain multiple users. All users belonging to a group will have the same access permissions to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

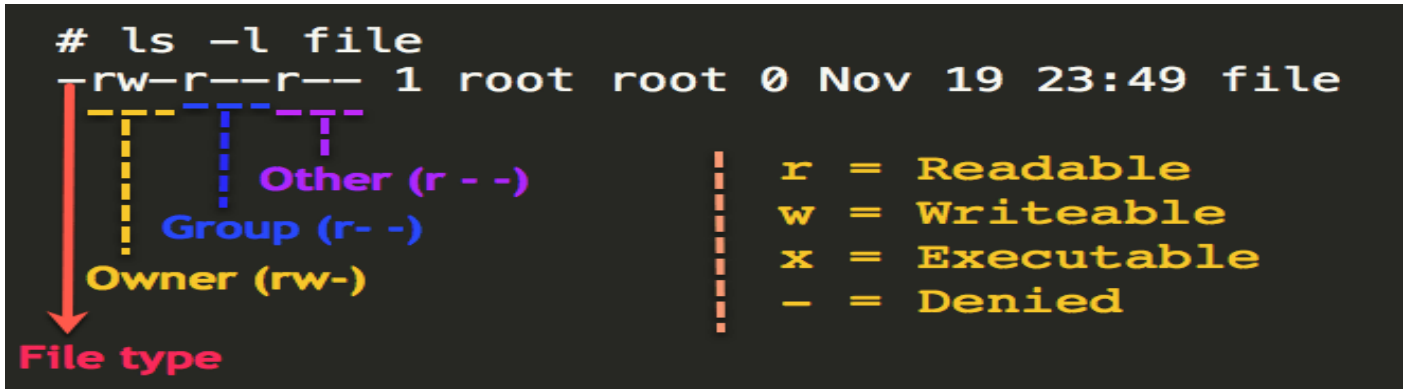
Other

Any other user who has access to a file. This person has neither created the file, nor he belongs to a usergroup who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for the world.

Permissions

Every file and directory in your UNIX/Linux system has following 3 permissions defined for all the 3 owners discussed above.

- **Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
- **Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.
- **Execute:** In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.



Changing file/directory permissions with 'chmod' command

Say you do not want your colleague to see your personal images. This can be achieved by changing file permissions.

We can use the '**chmod**' command which stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world. **Syntax:**

```
chmod permissions filename
```

There are 2 ways to use the command -

1. **Absolute mode**
2. **Symbolic mode**

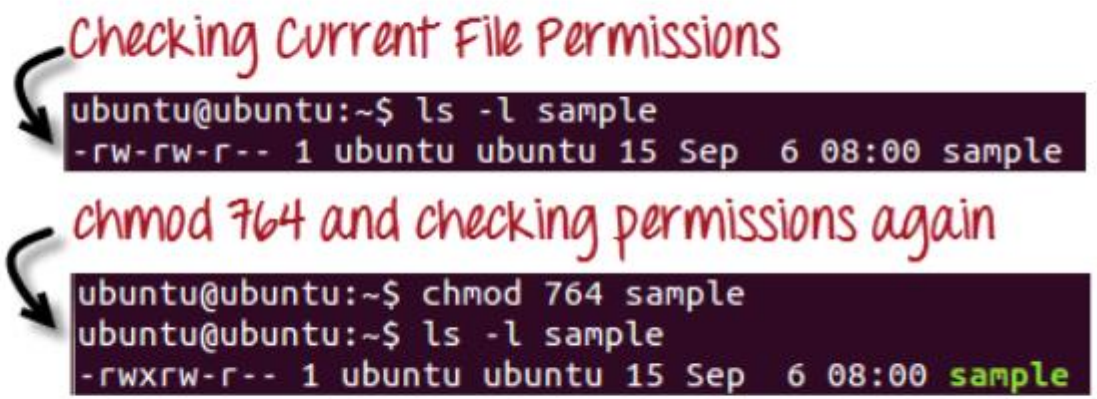
Absolute(Numeric) Mode

In this mode, file **permissions are not represented as characters but a three-digit octal number.**

The table below gives numbers for all for permissions types.

Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

Let's see the chmod command in action.



In the above-given terminal window, we have changed the permissions of the file 'sample to '764'.

Symbolic Mode

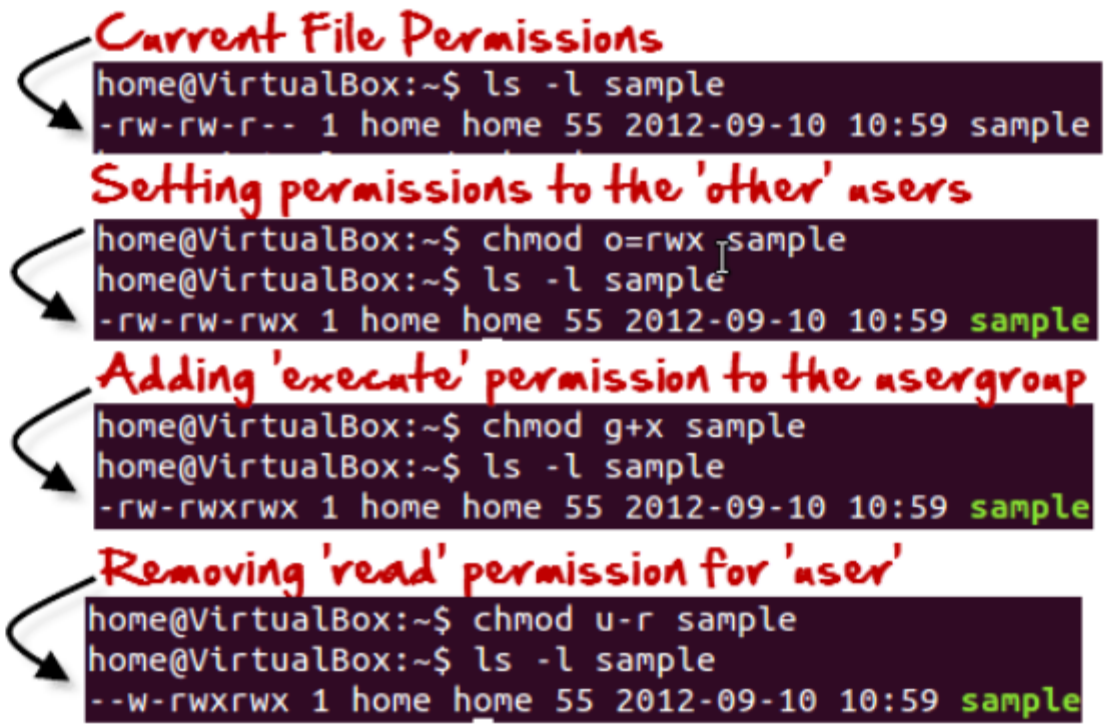
In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permissions set earlier.

The various owners are represented as -

User Denotations	
u	user/owner
g	group
o	other
a	all

We will not be using permissions in numbers like 755 but characters like rwx. Let's look into an example



1. How to Add a New User in Linux

To add/create a new user, all you've to follow the command '**useradd**' or '**adduser**' with 'username'. The 'username' is a user login name, that is used by user to login into the system.

Only one user can be added and that username must be unique (different from other username already exists on the system).

For example, to add a new user called '**tecmint**', use the following command.

```
[root@tecmint ~]# useradd tecmint
```

When we add a new user in Linux with '**useradd**' command it gets created in locked state and to unlock that user account, we need to set a password for that account with '**passwd**' command.

```
[root@tecmint ~]# passwd tecmint
Changing password for user tecmint.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

```
tecmint:x:504:504:tecmint:/home/tecmint:/bin/bash
```

The above entry contains a set of seven colon-separated fields, each field has it's own meaning. Let's see what are these fields:

- **Username**: User login name used to login into system. It should be between 1 to 32 charcters long.
- **Password**: User password (or x character) stored in /etc/shadow file in encrypted format.
- **User ID (UID)**: Every user must have a User ID (UID) User Identification Number. By default UID 0 is reserved for root user and UID's ranging from 1-99 are reserved for other predefined accounts. Further UID's ranging from 100-999 are reserved for system accounts and groups.
- **Group ID (GID)**: The primary Group ID (GID) Group Identification Number stored in /etc/group file.
- **User Info**: This field is optional and allow you to define extra information about the user. For example, user full name. This field is filled by 'finger' command.
- **Home Directory**: The absolute location of user's home directory.
- **Shell**: The absolute location of a user's shell i.e. /bin/bash.

Creating groups and adding users

Now it's time to create a group. Let's create the group editorial. To do this, you would issue the command:

```
sudo groupadd editorial
```

Now we want to add our new user, olivia, to the group editorial. For this we will take advantage of the *usermod* command. This command is quite simple to use.

```
sudo usermod -a -G editorial olivia
```

The *-a* option tells *usermod* we are appending and the *-G* option tells *usermod* we are appending to the group name that follows the option.

How do you know which users are already a member of a group? You can do this the old-fashioned way like so:

```
grep editorial /etc/group
```

```
[root@ip-172-31-39-69 ~]# groups anand
anand : anand hello
[root@ip-172-31-39-69 ~]# id anand
uid=1001(anand) gid=1001(anand) groups=1001(anand),1002(hello)
[root@ip-172-31-39-69 ~]#
```

userdel command examples

Let us remove the user named vivek or account named vivek from the local Linux system / server / workstation, enter:

```
# userdel vivek
```

Next, delete the user's home directory and mail spool pass the `-r` option to userdel for a user named ashish, enter:

```
# userdel -r ashish
```

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.

Windows, you have the NTFS, FAT, FAT16, or FAT32 filesystems.

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.

Mount an EBS volume to EC2 Linux

In this tutorial, we will teach you how to attach and mount an EBS volume to ec2 linux instances. Follow the steps given below carefully for the setup.

Step 1: Head over to EC2 –> Volumes and create a new volume of your preferred size and type.

Step 2: Select the created volume, right click and select the “attach volume” option.

Step 3: Select the instance from the instance text box as shown below.

Step 4: Now, login to your ec2 instance and list the available disks using the following command.

```
lsblk
```

The above command will list the disk you attached to your instance.

> **You Might Like:** Best AWS Certification and Training Course

Step 5: Check if the volume has any data using the following command.

```
sudo file -s /dev/xvdf
```

If the above command output shows "/dev/xvdf: data", it means your volume is empty.

Step 6: Format the volume to ext4 filesystem using the following command.

```
sudo mkfs -t ext4 /dev/xvdf
```

Step 7: Create a directory of your choice to mount our new ext4 volume. I am using the name “newvolume”

```
sudo mkdir /newvolume
```

Step 8: Mount the volume to “newvolume” directory using the following command.

```
sudo mount /dev/xvdf /newvolume/
```

Step 9: cd into newvolume directory and check the disk space for confirming the volume mount.

```
cd /newvolume
df -h .
```

The above command would show the free space in the newvolume directory.

To unmount the volume, you have to use the following command.

```
umount /dev/xvdf
```

EBS Automount on Reboot

By default on every reboot the EBS volumes other than root volume will get unmounted. To enable automount, you need to make an entry in the /etc/fstab file.

1. Back up the /etc/fstab file.

```
sudo cp /etc/fstab /etc/fstab.bak
```

2. Open /etc/fstab file and make an entry in the following format.

```
device_name mount_point file_system_type fs_mntops fs_freq fs_passno
```

For example,

```
/dev/xvdf       /newvolume    ext4    defaults,nofail        0       0
```

3. Execute the following command to check id the fstab file has any error.

```
sudo mount -a
```

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.

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

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

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

Crontab Different Services :

- Start/stop/reload/status a crontab service

  service crond start/stop/status/reload

- Setup a crontab to execute a job for every minute

  - Edit the crontab
    crontab –e

  - Schedule the job to execute for every 2 mins
    */2 * * * * sh /root/shell.sh

What are Linux log files

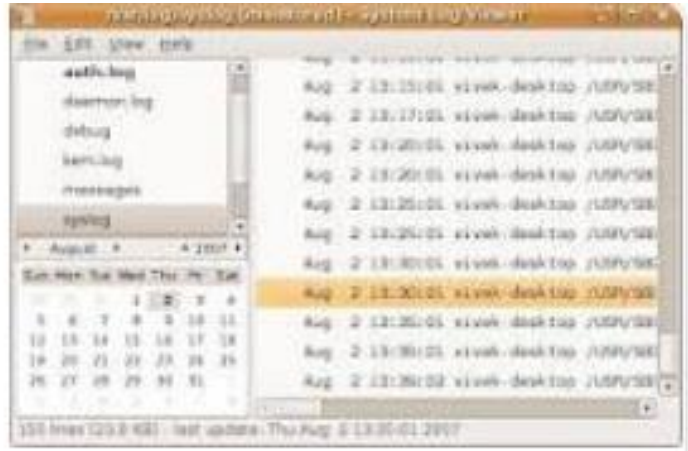
Log files are a set of records that Linux maintains for the administrators to keep track of important events. They contain messages about the server, including the kernel, services and applications running on it.

Linux provides a centralized repository of log files that can be located under the /var/log directory.

The log files generated in a Linux environment can typically be classified into four different categories:

- Application Logs
- Event Logs
- Service Logs
- System Logs

Common Linux log files names and usage



- /var/**log**/messages : General message and system related stuff.
- /var/**log**/auth.**log** : Authenication **logs**.
- /var/**log**/kern.**log** : Kernel **logs**.
- /var/**log**/cron.**log** : Crond **logs** (cron job)
- /var/**log**/maillog : Mail server **logs**.
- /var/**log**/qmail/ : Qmail **log** directory (more files inside this directory)

scp command in Linux with Examples

scp (secure copy) command in Linux system is used to copy file(s) between servers in a secure way. The SCP command or secure copy allows secure transferring of files in between the local host and the remote host or between two remote hosts. It uses the same authentication and security as it is used in the Secure Shell (SSH) protocol. SCP is known for its simplicity, security and pre-installed availability.

SCP examples

Copy file from local host to a remote host SCP example:

```
$ scp file.txt username@to_host:/remote/directory/
```

Copy directory from local host to a remote hos SCP example:

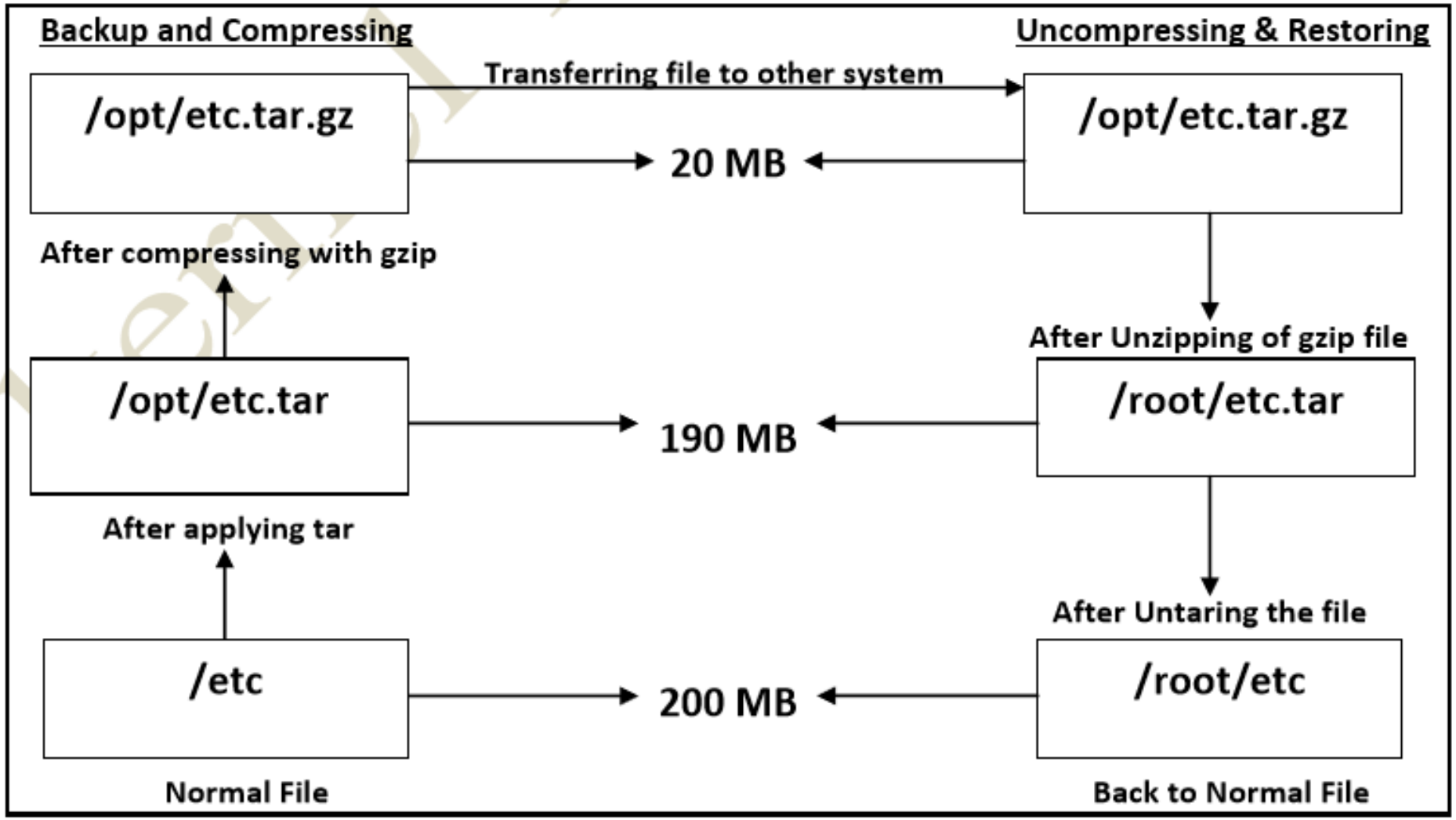
```
$ scp -r /local/directory/ username@to_host:/remote/directory/
```

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.

- 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



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]#
```

- 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.

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.

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
```

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 ~]#
```

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 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]#
```

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**

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
```

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

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
```