* **Kernal** =>>

It is a piece of s/w that **handles communication** between user program/s/w and

peripheral devices, **keeps records** of running program and **shares timing** among

these process according to well-defined policy, **how to share time** among

different user of the system, **how to allocate memory** for running programs,

**how to utilize** processor and other resourses.

* **Upgrade kernel to the latest version** :
* **Download** kernel source package
* Switch to **/usr/src** and **copy** downloaded kernel package to this directory
* **Extract** content of package then we will get a directory **linux-3.3.1** then go there **#cd linux-3.3.1**
* copy the contents **cp /usr/src/kernels/2.6.32-71.el6.i686/.config .** to the current directory
* Now run **#make oldconfig** command and select the appropriate processor
* Then **#make dep, #make clean, #make bzImage**
* Now **cp arch/x86/boot/bzImage /boot/vmlinuz3,**

**cp System.map /boot/System.map3**

**make modules, make modules\_install**

**mkinitrd /boot/initrd-3.3.1.img 3.3.1**

* Now update **/boot/grub/grub.conf** file with **change the value of kernel(/vmlinuz3)** and **initrd(/initrd-3.3.1img)** and reboot the system. We can verify using **uname –r**
* **Modify and set kernel runtime/tuning parameter :**

When we want to improve the performance of server, need to set the kernel runtime parameters.

We can do this in below ways-

> **sysctl –a** we can get all the available kernel runtime parameters

**sysctl –w parameter / echo 1 > proc/sys** To set a kernel runtime parameter

**sysctl –p** To refresh with the new configuration

Note :- This change is instantaneously active, to persist at reboot time need to write into **/etc/rc.d/rc.local** file.

To permanently store kernel runtime parameter, write them into the **/etc/sysctl.conf**

* **SHELL =>>**

Shell can be considered as a command interpreter. Users don’t have a direct interaction with the kernel.

Shell takes the users command, interprets them, and takes the necessary action to execute them. Also, provide o/p of command to the user.

How Shell execute command->

1. First it scan the command for **metacharacter**
2. Then passes the command to the **kernel** for execution.
3. Shell wait to complete and can’t do any work.
4. After complete, prompts reappear.

Types ->

**/etc/shells** -> To see all available shell in the system.

1. **Bourne** shell(/bin/sh), 2) **bash shell**(/bin/bash), 3) **korn** shell(/bin/ksh), 4) **C** shell(/bin/csh), 5) **Tcsh**(/bin/tcsh).

* To change shell for user

# **chsh**

If not, then use **#usermod –s** or edit **/etc/passwd.**

* To end a user session , **exit** command is uset

---------------------------------------------------------------------------------------------------------------------------------

* **Superuser(Root) =>**

It is **created during installation process**. It has privilege to do all system administrator task. Any user with UID **Zero(0**) is a root user

**root:x:0:1:Super-User:/:/usr/bin/csh**

* **Disable root account :**

To protect server we can disable root account. When setup new server disable root user and create normal user then use **sudo** or **su** to login as root user.

**# vi /etc/ssh/sshd\_config**

**PermitRootLogin no**

Then Restart ssh service

**$/etc/init.d/ssh restart**

**---------------------------------------------------------------------------------------------------------------------------------**

* **Extrinsic and Intrinsic command::**

Those command which are built into shell being used called Intrinsic (Ex: cd, alias) and command which are stored as separate file called Extrinsic (Ex: ls)

**History** :: Files(in user’s home dir) which store repeating comman commands. Bydefault, bash writes it’s history at the end of each session.

**HISTFILESIZE** variable configure how many command stores in history file

**HISTSIZE** control the number(ex: 1 - 50) of line stored in memory for the current session

**() Recall and execute Previous command** ::

* **!n** where, n is history number
* **!-n** to execute command relative to our current position.
* **!!** To execute previous command

**Alias -:**  Assigning meaningful names to the command which stored(permanantly) in **.bashrc** file.

**Alias c=’clear’ =>** to set alias which remains effective till current session

**Unalias c =>** to unset alias name

* **Date =:**

To set system date login as root

**# date format**  –**s “String”**

format +%[**m**(month in No.), **h**(month name), **d**(day of month), **y**(year), **H,M,S**(hour, minute, second), **D**(date in mm/dd/yy), **T**(time in hh/mm/ss)]

Ex: to set date

**date +%d%m%y –s “30122014”**

**---------------------------------------------------------------------------------------------------------------------------------**

* **SHELL Startup Files importance ::**

After login some commands are executed automatically and environment is set for user (Env. Var. controls the behavior of user’s shell) Ex: history length, Shell prompt, in which directory shell should look for extrinsic command.

**System startup file(/etc/profile) :** It is comman for all user and environment is globally set for all login user Ex: PATH var, display message to login user, Umask.

**(/etc/.bashrc)** used for system wide function and aliases also it’s a shell script Ex: PS1

* **User startup File :**

These files are present in **/etc/skel** and copy in user’s home directory only exceptions when used **–M** option while creating user

After login **interactive login** shell is started using **/bin/login**, by reading **passwd** file. An **interactive non-login** shell(present when shell script is running and it is processing without waiting for user input) is started at command-line

**.bash\_profile :**A shell script also called initialization files where we can automatically define special environmental variable and executed whenever a user logs in.

User specific environment variable and startup programs also instruct user session to check .bashrc file for user aliases and function. User can add own directory in current path.

Note :- This file is not automatically re executed after make changes to it and only executed whenever we log in. we can execute it with (.) command also, using **source** command.

**.bashrc ::** It is a conf. file executed each time after enter BASH shell or generate any subshells.

This file is used to control **user variable, alias function** and other **profile** during login session.

user can use any command automatically after login.

If user want’s to make their **shell environment**, they can set some code in .bashrc

**.bash\_logout ::** To clear the terminal after exit of current user

**--------------------------------------------------------------------------------------------------------------------------------**

* **Mail Addressing :**

**Recipient : Unix ID@Domain name(st.com)**

1. **Sending mail : $ mail mukhers**

**Subject**

* **- - - - - -**

**Ctrl-d**

Mail with subject and take msg body from a file

**$ mail –s “subject” mukhers < file**

Copy Mail to other people

**$ mail –s “subject” –c “ankitsh,ranjeet” mukhers < file**

1. After mail has been sent, it will be resides in **/var/spool/mail/mukhers**

Or user can use **$** **mail** command to view incoming mail

The pointer (**>**) indicate current message, to view msg user need to enter the number associated with msg or press (Enter) and then **q** to quite mail **and then message stored in user’s home to the mbox (home/users/mbox).**

User can save mail using **w** command (ex: w note3)

**m user :** forward mail to user and **!** run unix command.

-----------------------------------------------------------------------------------------------------------------------

* **Difference betn Windows and Linux**

1. On windows, S.A. **logs directly** to perform system admin duties. However, On Linux, S.A. **not directly log in as root**. Instead, S.A. log in as non-root user, and use other mechanism(**Su, Sudo, or policykit**) to temporary gain root priviledges.

**Ex :** sudo su – ankitsh

**Su –** To log into root with root passwd

**Note :- $ su username** : Start a non-login shell(just sets shell of that user with current environment setting )

**su – username :** Start login shell(start shell environment of that user)

1. On Linux, **root privileged not given** to other accounts. On windows, Admin priviledge often **given to other accounts**.
2. Windows is a **single user** multitasking O.S. while as unix is **multiuser** multitasking.

* **File system Hierarchy**

There are two groups of directories and contain **15** directory almost.

1. **Static**: where system files are srored and doesn’t change frequently or after system installation. Ex: **/sbin, /usr, /opt.**
2. **Dynamic :** changed on a routine basis. Ex: **log** file, **/var, /tmp.**

* **/ :** Top level directory called ROOT directory.
* **/root :** It is home directory for root user
* **/boot :** It contain bootable files for linux Like **vmlinuz, initrd, GRUB.** Also, kernel images files.
* **/lib/modules :** Holds kernel modules
* **/etc :** It contain all conf. files
* **/usr/share :** Data to be shared by different applications
* **/opt** : It is optional directory for /usr contain third party software
* **/bin :** contain all user command
* **/sbin:** contain all super user command. Contain executable files(script /**sbin/init.d**) needed at boot time.
* **/dev :** Contains device files
* **/proc :** It contain process files and useful info. Used by OS

**/proc/meminfo**(RAM/SWAP) **, /proc/cpuinfo(**CPU**)**

* **/var :** contain variable data like log files,mails

**/var/log/messages** log of all **system** task such as loading drivers, mounting file system, logins to system. Logs are managed by the **syslogd** daemon and configuration info. held in /**etc/syslog.conf** file. Only root can view these logs.

**/var/log/boot :** system boot log

**/var/log/utmp, wtmp :** Login records log and /**btmp** for failed login

**/var/log/dmesg :** kernel related log

**/var/log/secure :** info. related to authentication and authorization privileges.

**/var/tmp :** Holds temporary files that should be preserved while system booting

* **/mnt :** It is default mount point for any partition and empty**.**
* **/tmp :** This directory has permission for everybody to create or delete files. **Empty** at boot time and **sticky** bit is enable.

**-----------------------------------------------------------------------------------------------**

* **File creation :**
* **Gedit :** It is a graphical tool for editing **Text**  file similar to windows **Notepad.**
* **emacs**
* **Vim** and **nano** are command line editors.
* **touch :** create new **empty** file also, update last **modified** timestamp to the current time of existing file

**Vi optinn :**

**l :** Right I **i :** To begin insert mode at cursor position.

**h** : Left I  **I** : To insert at **beginning** of line

**k** : Up I  **a** : To append to the next **word** latter

**j** : Down I **A** : To append at end of the **line.**

**o :** To append new line **below** cursor position

**O :** To append new line **above** cursor position.

**gg :** To go **beginning** of page

**G :**  To go **end** of page

**w :** To move cursor **forward** word by word

**b :** To move cursor **backword** word by word

**u** : To **undo** last change I **Esc:w** To save changes

**yy :** To copy a line I **Esc:q** To quit

**p :** To paste line **below** I **Esc:wq** To save and quit

**P :** To paste line **above** I **Esc:wq!** To save and quit forcefly

**dw :** To **delete** word letter by letter(Backspace) I **Esc:20(n)** To go line 20

**dd :** To **delete** entire line

**/ :** To **search** pattern in file

:%s/word1/word2

* **Directory creation** :-

**#mkdir -p d1/d2/d3/d4** p for create subdirectory recursively

* We can change file parameters/values using

**#echo xxxxx > /proc/sys/kernel/hostname**

--------------------------------------------------------------------------------------------------------------

* **who :** Tells about how many users are logged in(ID, console, Login Time, remote host).

**who –Hu:** Shows when the activity has occurred(IDLE) where, (**. Or 0.00s in w** command) shows activity has occurred in the last one minute before command was invoked.

And shows process ID( As in ps command).

**-b** -- boot time of last system boot

**-d --** print dead processes

**-p** -- print active processes spawned by init

-q - - all login names and number of users logged on

Who use **/var/run/utmp.**

**/proc** process information

---------------------------------------------------------------------------------------------------------------------------------

* **W** : Tells who is logged in and what they are doing.

**Uptime :** Tellshow long the system has been up or running, no. of users, load avg of 1, 5, 15 min.

**JCPU : Time** used by all processes attached to the terminal includingcurrent running background **jobs** but not including past background **jobs**.

**PCPU : Time** used by current process which, names in **what** field.

----------------------------------------------------------------------------------------------------------------------

* **uname : print system information(feature of O.S. )**

**-r :** kernel release, or version of O.S. (ex: 5.8, 2.6.32-431.el6.x86\_64)

First kernel version was 0.01

**2 :** denotes **kernel version(released on 2003)** and it change only when truly major changes in the concept and the code of the kernel occur. It has been changed twice in the history

**6 :** denotes **major revision** of kernel version where **even** no. indicates stable version which is for production use i.e non-experimental. Likewise, odd no. represented development release for testing new feature and device drivers.

**32 :** denotes **minor revision** which only changed when new features or new drivers are added.

**431(it may be followed by letters, rcl, ac, ck, and mm) :** denotes **corrections** , such as security patches and bug fixes.

**-n :** machine name (same as hostname)

To check **version of OS /etc/redhat-release**

-**a,** --all print all information, in the following order,

except omit -p and -i if unknown:

-**s**, --kernel-name print the kernel name

-**n**, --nodename print the network node hostname

-**r,** --kernel-release print the kernel release

-**v**, --kernel-version print the kernel version

-**m**, --machine print the machine hardware name

-**p**, --processor print the processor type or "unknown"

-**i,** --hardware-platform print the hardware platform or "unknown"

-**o**, --operating-system print the operating system

------------------------------------------------------------------------------------------------------------------------------

* **Stty : Displaying and setting terminal characteristic**

**^c** : Interrup(intr) a program. setting > **stty intr \^c**

**^d** end of file(eof) for command that expect input from keyboard. setting > **stty intr \^d**

(terminate the input)

**^h** erase last character type

**^W** erase the last word typed

**^u** kill character erase the current line

**^r** will redraw the current line

^z suspend will send a terminal stop signal

**^S** will stop the output

**^Q** will restart the output after stopping it

**^\** quit

**Swtch** will switch to a different shell layer

**echoe :** This character set means backspace work, if unset (-echoe) means it is disable.

**echo :** Indicate whether shell script should accept password or not(-echo)

**sane :** set terminal char. to value that will work on most terminals.

* **Secure Terminal In Linux :**

To deny all local login to all normal user as well as only allow root login on first terminal i.e. tty1.

#touch /etc/nologin

Comment all available terminal in file **/etc/securitty** except **tty1** terminal.

As see in third line of **/etc/pam.d/login** file, pamodules **pan\_nologin** deny to all non-root users to login locally. Also, **pam\_securetty** modules checks the **/etc/securetty** file, which terminal are available to root.

**PAM** is about security, which check service should be used or not also validate passwords. Most of applications uses PAM for authentication. PAM consults conf. file to see what security action to take for an application and Admin can add or remove new rules at any time.

PAM module **pam\_listfile.so** authenticateusers

To block a user then make entry in **/etc/sshd/sshd.deny** file

Open **/etc/pam.d/sshd** module to append some file

* **item=user** : Check the username
* **sense=deny** : Deny user if existing in specified file
* **file=/etc/sshd/sshd.deny** : Name of file which contains the list of user (one user per line)

and restart **sshd** service

When user try to login using ssh then log entry recorded in **/var/log/secure**

---------------------------------------------------------------------------------------------------------------------------------

4=> File system

**ls :**  Listing file in ASCII sequesnce

**Number, uppercase and lowercase and in linux dir. and files in different colors.**

**-x :** o/p in multiple column(modern linux have this option)

**-F :** To identify directories and executable files(**indicated as \*** ) or **lsf.**

**- a** : hidden files

- **r :** list reverse order

**- t :** Last modification time(using vi editor)

**-u** : Last access time(using cat )

**-c :** last inode modification.

**-A** : To see if the directory is empty or not

**Note1** : when we create any sub-directory the first two files are special files( **. & ..)** represent current and parent directory and created by kernel which we can’t remove it nor write. They **help in holding file system together.**

**Note2 :**  If we change Perm/Ownership that doesn’t effect on modification time. The only content should be change

**# tree /directory :** To list files in the directory

# **echo ./\* or dir or find or locate or for()**

**5 =>** **Copy Files**

Syntax : **cp F1 F2 (**If file 2 doesn’t exist, it will create. If exist it will overwritten without warning**)**

**Option**

**-p :** To keep permission as it is as source file.

**Note :-** when we copy files in another directory, permission will set according to that directory. Hence we use **–p** option for copy with its original permission.

-**R** : To copy multiple files but, the destination must be directory and it must be exist**.**

**-r :** To copy a directory

**-i :** It first check destination file exist or not, if it exist then **it will ask(y)** for overwrite

=> **Stop overwriting :-**

**# set -o noclobber**

**6 => Remove Files :**

**rm –rf \* :** Remove all files and directory in current directory

**rm \* :** Remove all files

**rmdir :** To remove empty directory

---------------------------------------------------------------------------------------------------------------------------------

7 **Move file (Rename)**

When we try to move directory from one file system to another i.e across file system, it will occurs error

Note 1):- when we move multiple files/dir, we can’t rename them. We have to use separate mv for each item.

Note 2) : In cp/mv, when we want to move/copy multiple file/dir the destinamtion must be a directory.

**7 => grep(Global regular expression print) :** It scan input for a pattern and display line containing pattern, the line no, filename where pattern occur.

Syntax : **grep options pattern filename(multiple)**

**i ->** make search insensitive, ignore upper & lower case diff.

**v >** make search inverse

**n ->** Display line no.

**c >** count of the lines with the word

**l**  >Display filename of pattern

**e** > Matching multiple pattern

**f** > Take pattern from file

**Note :** If pattern consist of multiple word, then quoting is necessary.

If single pattern, we used with & without quote

To list only directory

**$ ls –l | grep “^d “**

**Matching multiple pattern(ERE)**

**$ grep -E ‘word1 | word2’ file**

**$ grep -E ‘(sen | das)gupta’ file**

- To serach several patterns in the file listwith more fast **use $ fgrep patterns file-list**

Can’t interpret special char and search for regular expn.

- To search patterns in the file list $ egrep '^(From|Date):' $HOME/mbox

Like grep, use regular expn. Unlike grep can also interpret extended special char. **? | +**

**---------------------------------------------------------------------------------------------------------------------------------**

**Extended regular expr**

It can be defined as a strings that represent several sequence of characters. It allow us to filter the output of command or file, edit a text ..

**- Metacharacter**

It is a string that allow to filter the output of a command or file

**+** Match **one or more** occurances

**?** Match **zero or one** occurance

**\*** Any no. of character[They don’t match file beginning with **dot(.),** also, **/** in a pathname]

**.** Match any character

[all] match all strings containing character a,l,l , [0-9] contain at least a single no.

**^** To search at beginning

**$** To search at the end

**^sagar$** Line containing only one word

^$ To match a blank line

**\**  Turn of meaning of metacharacter

Redirect std output(1)

**cat newfile > file1**

Append

**cat file1 >>file2**

Redirect std input(0)

**mail jane <myfile**

Redirect std error(2)

**ll xyz 2>abc**

stdout as well as to another file

**who | tee whofile**

* **Single quote:** Turn of meaning, meaning of special character turn off.

**‘ ’ protect** all special character they **don’t execute**

**NAME = ‘$NAME Junior’**

**echo $NAME**

$ **$NAME Junior**

* **double quote :**

**“ ” Not protect** all special character, **command execute.** Use variable evaluation or command substitution.

**NAME = “$NAME Junior”**

**echo $NAME**

**$** Mike Ron **Junior**

**$ echo “My login name is `whoami ` “**

* Assigning value of command to a variable

1. **NAME = `whoami`**
2. **Name=$(whoami)**

* Add home directory in **PATH** variable

It contain pathnames of directory where commands are located.

**$ PATH= $PATH : $HOME**

**Export PATH**  To make them accessible to any subshells

Note :- To add new directory to PATH permanently, need to edit **.bash\_profile** file.

* **8)** Effects of **permission :-**

File Directory

1. Read(**r**) content can be **read** | content(files) can be **listed [ls** work, ls –l not work**]**
2. Write(**w**) content can be **modified** | file in directory may be **created** or **deleted**
3. Execute(**x**) file can be **execute** | content can be **accessed**

**Note :-** 1) If directory have both **read and execute**, they can list and access it’s content.

1. Also, having **write** permission **without a read** will allow user to **delete** the file but, user will **not able to edit** file. Also, having execute without read will not able to execute file.
2. If only **execute perm. set** without read and execute then we can do **cd** into directory

and execute program inside the directory if program name is already known.

1. If **execute perm. not set**, files inside directory can be listed(**ls work but, ls –l not work**) with name only. Also, no files inside directory can be **read & executed**
2. If directory contain a file that user can’t delete(no **write** perm.), then **can’t delete directoey** even write perm. for directory.

* File permission :

**chmod =>** change file access permissions

**Syntax :: chmod [option] mode File/Directory**

Option : mode:

**u >** owner **r (4)** read setuid

**g >** group **w(2)**  write setgid

**o >** other **x(1)** execute sticky bit

**a >** all

**+** To **grant** perm.

- To **revoke** perm.

= To exactly assign a perm., regardless of the previous perm.

Means if perm –rwx rw- r- - and after use **a=r** will be –r— r— r—

* Default Permission :-

When we create new file or directory, bydefault file have **666(rw-rw-rw-)** and directory have **777(rwxrwxrwx).** We can control default perm with the **umask** command which show which perm. will be revoked when new file/directory create.

**Note :- 1.** We set the umask in **$HOME/.profile** file so that the value set as soon as user logs in.

2. If chmod command shows perm. Denied.

A. First check perm. file(**/usr/bin/chmod**) should have **x** perm

B. can set perm using **# setfacl -m u:root:x /bin/chmod**

**#setfacl -m u: :x /bin/chmod**

Or we can use **#/lib/ld\_liloxd.z /bin/chmod**

* **Chattr command :**

Used to stop accidently delete of files and folder. Can’t delete/move/append the files secured via chattr attribute even though files have full permission. Useful for files like **passwd** and **shadow.**

**Syntax**

**#chattr [oprator] [switch] file**

[operator] > + attributes to be added to the existing attributes,

* to remove and ‘=’

[switch] > **a** can only be open in **append** mode for writing and read the file.

**i** cannot be modified: it cannot be deleted or renamed, no link can be created to this file. **We can only read the file.**

Ex : # **chattr +I file** To set the attribute

# **lsattr file** to list the attribute

-----------------------------------------------------------------------------------------------------------------------------

* **10. Changing Owner and Group** :-

Any user other than owner can’t change **ownership** , except superuser. A user can change group membership of file only if he/she is member of that group.

**$ chown newuser file**  To change owner of file

**$ chgrp newgroup file** To change group of file

**$ chown newuser : newgroup** To change owner as well as group

Note :- 1) Use **–R**  when we change ownership/group of **directory**

2) When we change ownership of **Soft** **link** file(But, link file will have same ownership), ownership of file to which the link points is also changed. To avoid this, use **–h** option.

In **Hard link**, ownership will change for both files. No effect of **–h** option.

---------------------------------------------------------------------------------------------------------------------------------

* **11 Special Permission**

1. **Sticky bit ::** Used to protect files in that directory and only **set for directory**.

File may be remove by anyone who has **write** permission to the directory regardless of the ownership or perm. on file. This can be overridden with sticky bit.

If write and sticky bit both set on directory then only **owner can delete.**

**Ex. /tmp** directory has **777(**rwxrwxrw**t)** perm. ,so any user can delete data. To avoid this we set sticky bit.

**$ chmod o+t(1) Directory**

1. **Setuid ::** when any user executes command which has SUID set, that user **gets temporary owner privilege.**

Means effective UID for user changed to owner UID for that file.

**Ex. /etc/passwd** rw**s**r-xr-xthis file owned by root but, when any normal user execute this command gets root priviledge and able to modify the content.

**$ chmod u+s(4) file**

1. **Setgid ::** File created in the directory will inherit **same group affiliation as that of directory,** rather than inheriting from creating user.

**Ex :** rwxrw**s**r-x sagar **bhuyar** so, newly create file has group **bhuyar**. Although, user has different primar group.

**$ chmod g+s(2)**

1. **Access control list :** To **grant or deny** perm. to users or group

To enable :-

**/home /home ext3 default,acl 1 2**

ACL need to enable while mounting the partition

**# mount –o acl /partition /home**

**Note –** If partition already exist, then add acl after default

# **getfacl /Dir** To check acl permission

**# setfacl <option> <arg>: perm. <Dir/File>**

**Option: m** To modify, arg : **u** owner

**x** To remove, **g** group

**R** Recursive **o** all

**b** Remove acl

**# setfacl –m u:sagar:rw /dir**

**-**rwxrwxrwx **+**3

**---------------------------------------------------------------------------------------------------------------------------------**

* **12. Find**

It recursively examine directory tree for file, and then take some action on selected file.

Syntax : **find Path-list selection criteria action**

**Note :-** 1)If path-list is not given, then search in current directory

2) When find is used to match metacharacter(wild character) pattern then it should be in **“ ”**(double quote)

* selection criteria :
* locating file by **inode no.**

$ find / **inum 1234** - print (display file default)

* File type and permission

$find **.**  **–type d** -print [Locate all directory]Also, **f** for ordinary and **l** for link file

$ find $HOME **–perm** 777 **–type d** –print

* Finding unused files

$find / **-mtime -2** print[List of files that have been **modified** in **less than 2 Day**]

* **-mmin -x** modified **in less than x Min**

**Note : +x : x days before**

**x : Exact on x day**

**-x : within x days**

$find / **-atime +365** –print[All files that have been **accessed** for **more than** **Year**]

* **-amin +x**  accessed in **more than x Min**
* **user usname** owned by username
* **- group gname** owned by groupname
* **- size +x[c]** size greater than x block
* **Prune**  Don’t descend directory if matched
* Action :
* **print**
* **ls**  execute **ls –lids** on selected files
* **exec cmd**  execute unix command followed by **{ } \ ;**

ex :

# find / -name “.log” –exec rm –rf { } \; [To remove all log files]

# find /usr/bin -perm -u+s –exec ll { } \; [To list all files for which SUID bit is set]

# find / -perm –u+s, g+w, o+w [To list all files which SUID, & write perm for group and other]

# find -type f -name "\*.log.\*" -size +100M -exec du -sh {} \; To find size of log files greater thatn 100MB.

---------------------------------------------------------------------------------------------------------------------------------

**# wall :** All users currently log in will receive this msg on their terminal

---------------------------------------------------------------------------------------------------------------------------------

* **13 Link File :-**

Many times we need to refer same file that has different name. Link file that is not the actual file but **it points to some other file** to which it is link.

Ex : **telinit** is really a link to the init command.

**Inode :** Every file is associated with inode that contain everything **except name and content & even inode no.**

* **Inode contain :**

Type of file, permission, link count, UID, GID, size, last access time, last modification time , time of last inode change

* We can check inode using **#stat filename**

**Note** :- If /root partition has 100% inode used then we cann’t create files/Dir inside it.

We can check usage of inode **# df –hi**

Directory store inode no. along with filename

* **Hard link :-** Have **same inode** no. , changes made in one file are automatically available in other file

**Syntax : $**  **ln file1 file2(must not exist)**

**Note** :- ln command return error when destination file exist. Use **–f** option to force removal of existing file

**Link count will be 2**

Limitation :- We **can’t link a directory** within same F.S.

We can’t have two linked filename in two F.S.

* **Soft link :** Have **different inode** no., soft link **doesn’t have files content** but, simply provide pathname of file that actually has content

**Syntax : ln –s file1(dir1) file2**

**Note** : Link count for file1 & file2 will be **1** but, **dir1** will be **2**

**Size**  of soft link is equal to no. of character in the name of original file

If we **remove file1,**  we would lose file containing data

Unlike hard link, soft link establish **across F.S.**

We can link multiple files but, destination must be directory.

---------------------------------------------------------------------------------------------------------------------------------

**14. Backup and Restore**

Copies of data which may be used to restore original after a data loss

* **Archieve :** Group of set of file into single file

To create :-

**$ tar –cvf <Destination> <source file>**

To extract :-

**$ tar -xvf <file.tar>**

To list content :-

**$ tar –tvf <file.tar>**

* **Compression :**

$ **gzip**(.gz)file.tar | **zip(**.zip**)**

$ **gunzip** file.tar.gz | **unzip**

To create archive and compress :

$ **tar –cvzf**

To view compressed file use **gzcat**

15. **Managing SElinux(Security enhancement) :-**

It allows

It adds protection for files, applications, process, and so on.

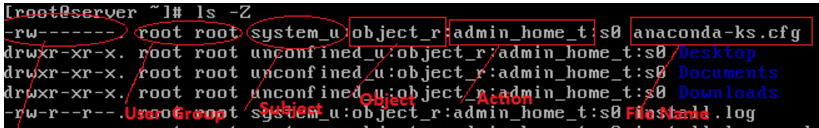
- To check SELinux mode **#getenforce**

- To change the modes **#setenforce <0 / 1>** where **0** for **permissive** and **1** for **enforcing**

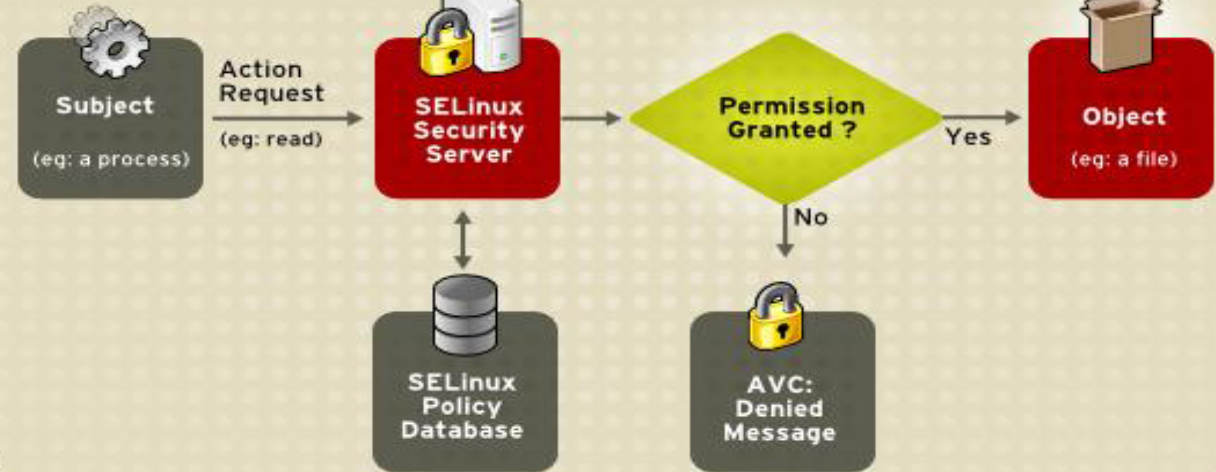
**Note** :- To change it to **disable mode** edit file **/etc/selinux/config** and make entry **SELINUX=disabled** and then **restart** the system

- To show current status **#sestatus**

* Display SELinux context of file **# ls –z file** and for dir **#ls –ldZ Dir**



* **SElinux context** :-



* **subject** :- subject is a **command**, **process** or **application** which want to access any linux file.
* **object** :- object is a linux **file** or **services**.
* **action** :- an action is what may be done by the subject to the object.

**Processes** and **files** are labeled with a SELinux context that contain **SELinux user, role, type, level**

* File context

|  |  |  |
| --- | --- | --- |
| **Contexts** | **Values** | **Description** |
| User: | unconfined\_u | Unprotected user |
| system\_u | System user |
| user\_u | Normal user |
| Role: | object\_r | File |
| system\_r | Users and processes |
| Domain: | unconfined\_r | Unprotected file or process |

- Changing the context **#chcon -t <argument> file/Dir**

**-u sets user context**

**-r sets role context**

**-t sets type context**

- Restore the context **#restorecon –v file**

- Checking the Booleans **# getsebool –a** and modify it **#setsebool <option> <on/off>**

for all services .

**Note: -** 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.