UNIT - V

File ownership and access permission

Basic file permission

Permission Groups (users)

Each file and directory has three user based permission groups.

Owner :- The owner permission apply only the owner of the file or directory, they will not impact the actions of other users.

group :- The group permissions apply only to the group that has been assigned to the file or directory, they will not affect the actions of other users.

All users :- The all users permission apply to all other users on the system, this is the permission group that you want to watch the most.

Permission Types

Each file and/or directory has three basic permission types -

Read - r

The read permission refers to a user's capability to read the contents of a file.

write - w

The write permission refers to a user's capability to create or modify the contents of the file or directory.

execute - (*)

The execute permission affects a user's capability to execute a file or view the contents of a directory.

Viewing the permission -

We can view the permission by checking the file or directory permission or by reviewing the output of the "ls -l" command while

in the terminal and while working in the directory which contains the file or folder.

The permission is the command line is displayed as-

-rwxrwxrwx 1 owner: group

1. user right / permission

1 The first character that marked with underscore is the special permission flag that can very.

2. the following set of three characters (rwx) is for the owner permission.

3 The second set of three characters (rwx) is for the group permission.

4. The third set of the three characters (rwx) is for the all user permissions.

5. following real grouping since

Page No. :
Date : / /

the integer/number displays the number of hardlinks to the file.

3. The last piece is the owner and group assignment formatted as owner:group.

Explicitly define permissions.

To explicitly define permissions you will need to reference the permission group and permission types.

The permission groups are :-

u - owner
g - Group
o or a - All users

The potential Assignment operators are + (plus) and - (minus). These are used to tell the system whether to add or remove the specific permissions.

Page No. :
Date : / /

for example :-

Lets we have a file named file1 that currently has the permission

-rw-rw-rw

which means that the owner, group and all users have read write permission only.

CHMOD # change to permission

we can change or modify permission by using chmod command.

The permission types that are used are -

r - Read
w - Write
x - Execute

Modifying permission.

for modification (to modify) we need chmod command.

syntax

chmod a-rw file1

To add permission we need chmod a+rw file1

As we can see, if we want to grant those permission, we need to change the minus character to a plus to add permissions.

How to change the owner and group assigned to a file / directory

CHOWN :

we can use chown command to change owner and group assignments.

Syntax

chown owner:group filename

for example :

To change the owner of file1 and the group to family we will enter

chown user1:family file1

Page No. :
Date : / /

Advance permission :-

The special permission flag can be marked with any of the following -

- - no special permission

d - directory

l - the file or directory is a symbolic link

s - this indicated the setuid / setgid permission.

t - This indicates the sticky bit permission

setuid / setgid special permission

The setuid / setgid permissions are used to let the system to run an executable ad the owner with the owner's permission.

Be careful using setuid / setgid in permission. If we incorrectly

assign permission to a file owned by root with the setuid/setgid bit set, then we can open our system to intrusion.

CHMOD :- (change mode)

This command is used to change the file permission for an existing file. we can use any one of the following notations to change file permissions-

a) Symbolic Notations
b) Octal Notations

Symbolic Mode

general format is -

chmod user_symbols set/deny_symbol access_symbols <filename(s)>

where user_symbols can be,

u — user

g — user group

o. others

whose set/deny symbols can be

+ Assign the permission
- Remove the permission
= Assign absolute permission

where access_symbols can be

r readable
w Writable
x Executable

Example :-

$ chmod u+x file1

This command adds execute permission to the user for executing the file - file1.

$ chmod g+x file1

This command assigns execute permission to usergroup to execute the file file1.

$ chmod ugo -rwx file2

This command removes read, write and execute permissions from the user, usergroup and others for the file file2.

Octal Notation

This mode uses a three digit no. to change the file permission.

In this number, the first digit represents user permission, the second digit represents usergroup permissions and the third digit represents other permissions.

General format is

chmod three_digit_no <fileName> [<filename2>...]

Digits and their meanings

0	No permissions
4	Read
2	Write
1	Execute

we can also sum the numbers for mixing of permissions.

3 Write and Execute
5 Read and Execute
6. Read and Write
7. Read, Write Execute

for example :-

```
$ chmod 740 file1
```

The user can read, write, and execute the file file1.

chown (change ownership)

This command is used to change the owner of the specified file.

Only the owner of the file and the super user can change the file ownership.

Syntax :-

```
chown <new-owner> <filename>
```

This command changes <new-owner> as owner of the file specified in <file name>.

chgrp :- (change group)

This command is used to change the group ownership of a specified file.

only the owner of the file and the superuser can change the group ownership of the file, irrespective of whether the user belongs to the same group or not.

general format is -

chgrp <new_groupname> <filename>

This command makes <new_groupname> as group-owner of the file specified in <file-name>.

Note that the group-owners of the file is also the group to which the file-owner belongs.

general format is:-

chgrp <new-groupname> <filename>

This command makes <new-group> as group-owner of the file specified in <filename>.

Note that group-owner of the file is also the group to which the file-owner

Example:-

chgrp mca file1.txt

Page No. :
Date : / /

User and its home directory

A home directory, also called a login directory is the directory on Linux operating system, the directory served as the repository for a user's personal files, directories and programs.

It is also the directory that a user is first in, after logging into the system.

A home directory is created automatically for every ordinary user in the called /home.

A standard sub-directory of the root directory, /home has the sole purpose of containing user's home directories.

The root directory, which is represented by forward slash (/) is the directory that contains all other directories and their sub-directories as well as all files on the system.

The name of user's home directory is by default identical to that of the user.

Thus for example — A user with a user name of "xyz" would typically have a home directory named "xyz".

It will have an absolute path name of /home/xyz. /home/bca5

(2) Absolute pathname :-

An absolute pathname is the location of a directory or file relative to the root directory, and it always starts with /.

The only user that will by default have its home directory in a different location is the root.

root is administrative user whose home directory is /root

There are several easy ways for a user to return to its home regardless of its current directory.

The simplest of these is to use the cd (change directory) command without any options, or agument.

The absolute pathname of a user's home directory is stored in that user's $HOME environmental variable.

environmental variable

Environmental variables are a class of variables that tell the shell, how to behave when a user works at the command line.

Thus a third way for a user to return to his home directory is to use

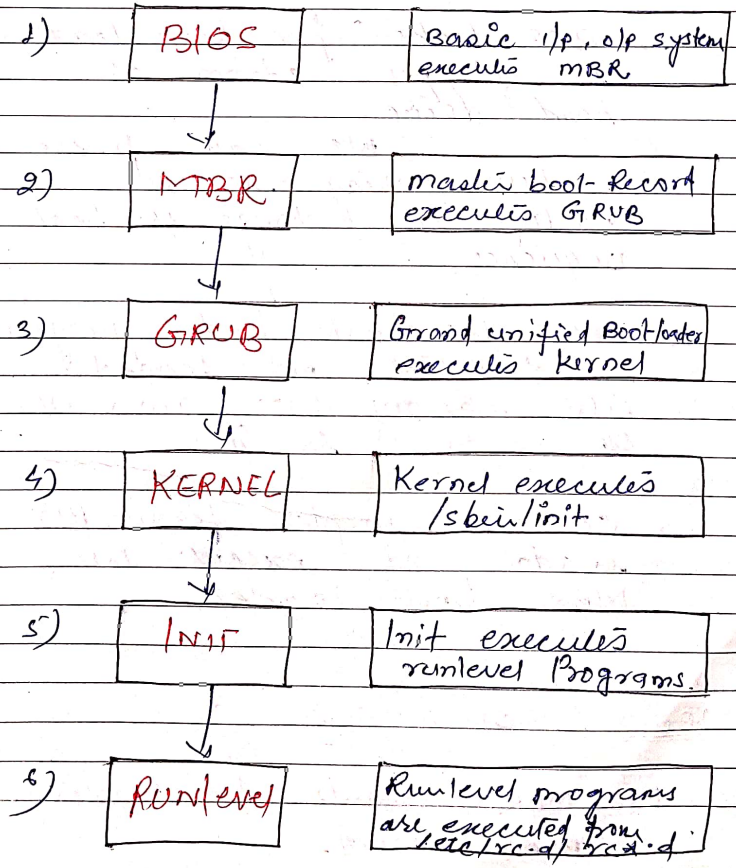
cd $ Home

i.e use $Home Environmental variable as an argument to cd.

Linux Boot Process

6 stages of Linux Boot Process

The following are the 6 high level stages of a typical Linux boot process.



1. BIOS (stage - 1)

→ BIOS stands for Basic I/O system.

→ Performs some integrity checks.

→ Searches, loads and executes the boot-loader program.

→ BIOS looks for boot-loader in floppy, cdrom, or hard drive.

We can press a key during the BIOS startup to change the boot-sequence.

→ Once the boot-loader program is detected and loaded into memory BIOS gives the control to it.

→ So in simple terms BIOS loads and executi the MBR boot-loader.

2. MBR (stage - 2)

→ MBR stands for Master Boot Record.

→ It is located in the 1st sector of the bootable disk. Typically /dev/hda or /dev/sda.

→ MBR is less than 512 bytes in size. MBR has three components -

a) Primary boot-loader info in the first 446 bytes.

b) Portion table info in next 64 bytes

c) MBR validation check in last 2 bytes.

→ It contains information about GRUB (OR LILO in old system).

→ So in simple terms MBR loads and executes the GRUB boot-loader.

3. GRUB (stage - 3)

→ GRUB stands for grand unified Bootloader.

→ If we have multiple kernel images installed on our system we can choose which one to be executed.

→ GRUB displays a splash screen, waits for few seconds, if we don't enter anything it loads the default kernel images as specified in the grub configuration file.

→ GRUB has the knowledge of the file system but the older Linux loader LILO did not understand file system.

→ GRUB configuration file is

/boot/grub/grub.conf

→ As we can see grup contain kernel and initrd images.

→ So in simple terms Grub just loads and executes kernel and initrd images.

4. KERNEL (4- stage)

→ Mounts the root file system as specified in the "root=" in grub.conf.

→ Kernel executes the /sbin/init program

→ Since init was the first prog. executed by linux kernel, it has PID 1.

We can check pid with

`ps -ef | grep init`

→ initrd stands for Initial RAM Disk.

→ initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted.

It also contains necessary drivers, compiled inside, which helps it to access the hard drive partitions and other hardware.

5) INIT :- (stage - 5)

→ It looks at the /etc/inittab file to decide the linux run level.

→ following are the available run levels.

0 - Halt
1 - Single user mode
2 - Multi-user without NFS
3 - Full multi user mode
4 - unused
5 - x11
6 - reboot

→ INIT identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.

→ Execute grep initdefault /etc/inittab on your system to identify the default run level.

→ If want to create trouble, set runlevel 0 or 6.

→ set default run level 3 or 5 for better performance.

6. Runlevel programs (stage - 6)

→ When the linux system is booting up, we can see various services getting started.

→ for example it might say,
"starting sendmail ... OK"

These are the runlevel programs executed from the run level directory as defined by our run level.

→ Depending on our default init level setting, the system will execute the programs from one of the following directories.

- Runlevel 0 — /etc/rc.d/rc0.d
- Runlevel 1 — /etc/rc.d/rc1.d
- Runlevel 2 — /etc/rc.d/rc2.d
- Runlevel 3 — /etc/rc.d/rc3.d
- Runlevel 4 — /etc/rc.d/rc4.d
- Runlevel 5 — /etc/rc.d/rc5.d
- Runlevel 6 — /etc/rc.d/rc6.d

→ under the /etc/rc.d/rc*.d/ directories we can see programs that start with S and K.

→ program start with S are used during start-up.

→ program start with K are used during shut-down. K - for Kill.

→ These are numbers right next to S and K in the program names.

These are the sequence numbers in which the program should be started or killed.

for example

S80 sendmail is to start sendmail daemon, which has the sequence no. of 80.