② Super block :-

- It is consits of file system information.
- The way of data is stored in system it is depended upon file system, ntfs, ext2, ext3, FAT32, exFAT etc, like formate of the file system.
- It is responsible for the storing the data.
- Linux is support all the operating system.
- Linux is understand the all the OS data.
- Linux is having ext2 and ext3 file system but linux is support all the file system.

③ inode block :-

- It is consits of file information. It means every file is create one lookup table which is known as index table. which consits all information of the file.

④ data block :-

- Content (data) of the file is present in the data block.

* file system Organize the all data.

Linux support 7 different type of the file :-

1) Regular file (executable file, text file)
2) directory file (mkdir)
3) Link file
4) Pipe } → IPC related file
5) Char special file } → device driver related file
6) Block special file
7) Socket file } → TCP-IP related file

e.g vi abc

abcd
abcd
abcd
⋮

after (ls -l)

-	rw- r-- r--	1	Savan	Savan	57	Nov-13 03:38	abc
type of file	file access Permission	no. of hard links	user name	group name	file size	last modification time	file name

① ls -l ..

total 24

drwxrwxr-x 2 ----
drwxrwxr-x 2 ----

directory file (d means)

② ls -l /dev

total 0

crw-rw---- 1 root ---- (upto end)
crw-rw---- 1 root ---- (upto end)

c means Character Special file

l → link file

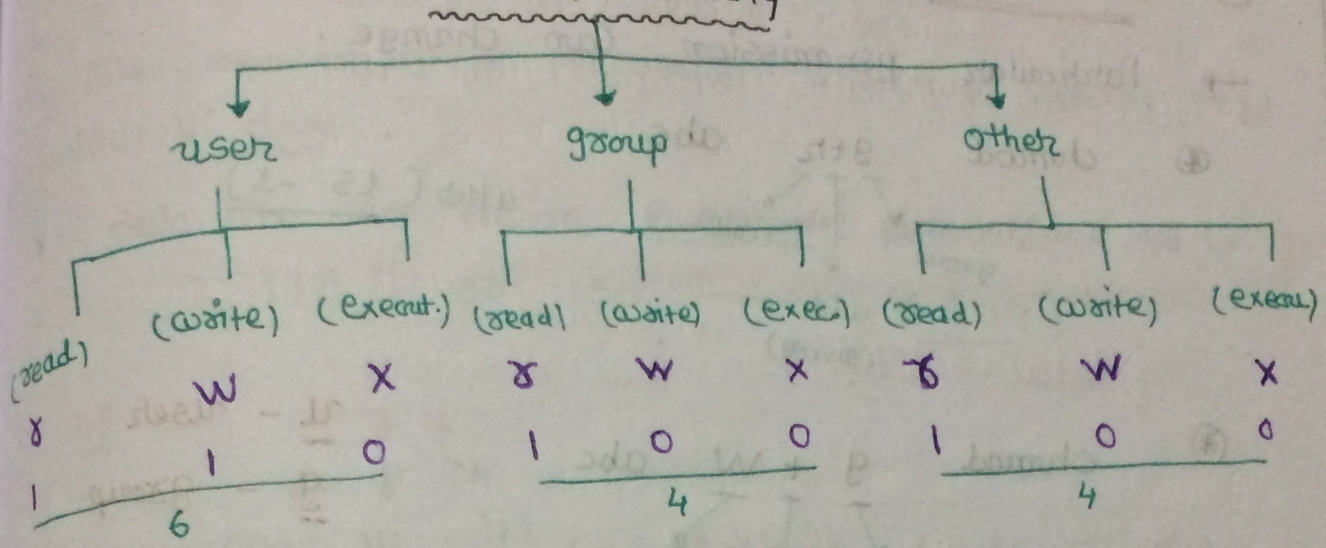
p → Pipe file

s → Socket file

* In Linux there are three (3) type of the security

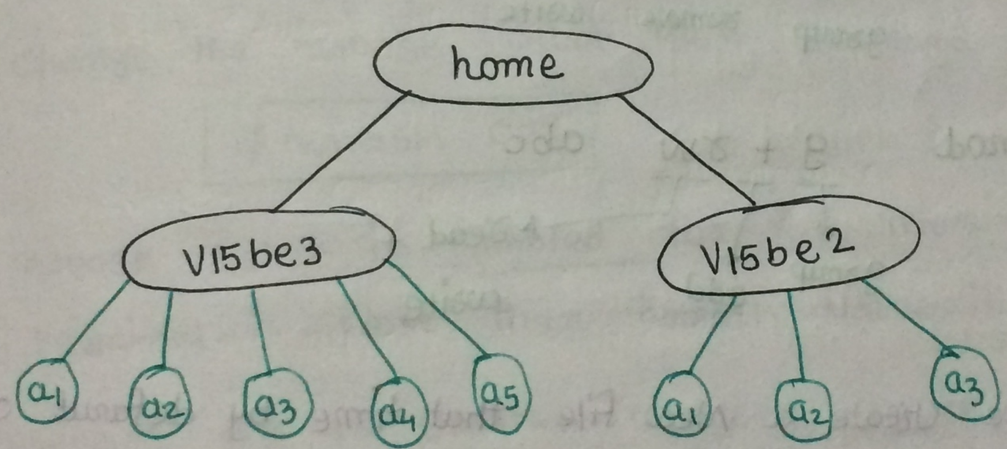
1) user
2) Group
3) Other

level of Security



0644 ⟶ file permission

→ The permission is represent in a Octal number.



→ whenever Create regular file manager giving this permission.

→ If we wants to change the permission of file access one command is use

Chmod

→ Chmod is used in two diffirent way

Chmod

1st method

2nd methode

① 1st method :-

→ You can give all new permission.

e.g chmod 0777 abc

② 2nd method :-

→ Particular permission can change.

* chmod g+r abc

g → group, + → add (giving), r → read

after (ls -l)

* chmod g + w abc

g → group, + → add, w → write

u - user
g - group
o - other

* chmod g - w abc

g → group, - → remove, w → write

* chmod g + rw abc

g → group, + → add, rw → read + write

⇒ When we create a New file that time by default 0644.

* $ umask

File permission is using a umask.

0022

0	0	2	2
for octal system	000	010	010

complement

111	101	101
110	100	100

(644)

For executable file (1) otherwise (0).

→ File manager give permission when we execute text file Vi data.

→ data file not having executable permission so

110 100 100 → (0644)

e.g

0 7 7

000 111 111

compl. → 110 000 000

file per. → 110 000 000 (600 file permission).

→ Change the umask Value than Command is

$ umask 077

→ umask Value is limited for the terminal.
(terminal is close than umask Value is default).

→ whenever file is created memory allocated in block Formate not in byte Formate.

Command ls -ls → New file create Vi def
{
aaaa----
1000 times
}

→ file size is more than 4096 byte, than other 4 byte create.

total = 8 (4097 char.)

block
8

user
group
other

ult 0644.

3 Command :- ls -li

1716110 -rw-r--r-- 1 .....

inode number

⇒ process managers recognize the process as process id.

⇒ like file managers recognize the file as an inode number.

There are two file present
Than

total = 12 → number of block recognize for that directory.

Program :- WAP find out the size of the file without opening the file.

```
#include <stdio.h>
# include <sys/types.h>
# include <sys/stat.h>
# include <unistd.h>
main (int argc, char ** argv)
{
        struct stat v;
        if (argc != 2)
        {
                printf (" Usage : ./a.out filename\n");
                return;
        }
        if (stat (argv[1], &v <0)
        {
                perror (" stat");
                return;
        }

Printf (" Size = %d\n ", v.st_size);
```

```
printf (" i
printf (" 
printf (" 
}
```

Output :-

Program (*)

```
# include
# include
# include
# include
main (i
{
        str
        if
        {

        }
        i

Printf
Printf
}
```

```
printf (" inode number = %d\n", V.st_ino);
printf (" User id = %s\n", V.st_uid);
printf (" group id = %s\n", V.st_gid);

}
```

output :- size = 4097
inode number = 176610
User id = savan
group id = savan

program

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
main (int argc, char ** argv)
{
    Struct stat V;
    if (argc != 2)
    {
        printf (" Usage : ./a.out filename\n");
        return;
    }
    if (Stat (argv[1], &V < 0)
    {
        Perror (" Stat");
        return;
    }
  Printf (" %o\n", V.st_mode);
  Printf (" %o\n", V.st_mode & 0777);
}
```

Output 1) 100644

mode

type of file — Permission

2) 644

$ man 2 stat

```
# include <sys/types.h>
# include <sys/stat.h>
# include <unistd.h>

int stat (const char *path, struct stat *buf);
int fstat (int fd, struct stat *buf);
int lstat (const char *path, struct stat *buf);
```

All these system calls return a stat structure. Contain of the structure is following :-

```
struct stat
{
    st_dev ; // device ID
    st_ino ; // inode number
    st_mode ; // ~~number of hard links~~ Protection
    st_nlink ; // ~~User ID~~ hard links
    st_uid ; // ~~Group~~ User ID
    st_gid ; // group ID
    st_rdev ; // device ID (if special file)
    st_size ; // total size in byte.
```