# Disk Devices Systems &Permissions

## Disk Device Terminology (Optional)

**ide**

Actually, the title should be ata or scsi, since ide is an ata compatible device. Most desktops use ata devices, most servers use scsi.

**ata**

An ata controller allows two devices per bus, one master and one slave. Unless your controller and devices support cable select, you have to set this manually with jumpers. With the introduction of sata (serial ata), the original ata was renamed to parallel ata. Optical drives often use atapi, which is an ATA interface using the SCSI communication protocol.

**scsi**

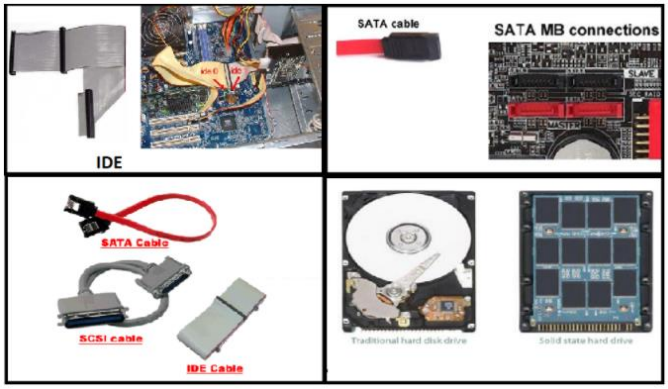
A scsi controller allows more than two devices. When using SCSI (small computer system interface), each device gets a unique scsi id. The scsi controller also needs a scsi id, do not use this id for a scsi-attached device.

**block device**

Random access hard disk devices have an abstraction layer called block device to enable formatting in fixed-size (usually 512 bytes) blocks. Blocks can be accessed independent of access to other blocks.

**solid state drive**

A solid state drive or ssd is a block device without moving parts. It is comparable to flash memory. An ssd is more expensive than a hard disk, but it typically has a much faster access time.




## Device Naming

Linux used to deal with two kinds of drives, depending on the electronic interface (controller), IDE and SCSI.

### ATA (IDE) Device Naming (Old IDE Names)

All ata drives on your system will start with /dev/hd followed by a unit letter. The master hdd on the first ata controller is /dev/hda, the slave is /dev/hdb. For the second controller, the names of the devices are /dev/hdc and /dev/hdd.

- ide0 Controller device naming

| connection | device name |
|---|---|
| master | /dev/hda |
| slave | /dev/hdb |

- ide1 Controller device naming

| connection | device name |
|---|---|
| master | /dev/hdc |
| slave | /dev/hdd |

A typical PC has two IDE controllers, each of which can have two drives connected to it. For example, /dev/hda is the first drive (master) on the first IDE controller and /dev/hdd is the second (slave) drive on the second controller (the fourth IDE drive in the computer).

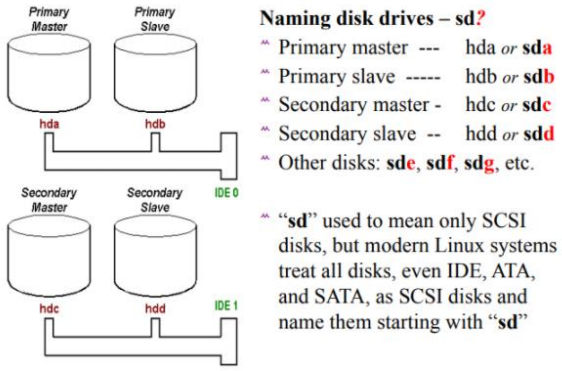
**Tips:**

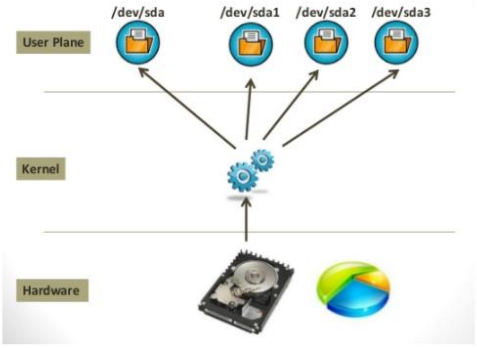
- IDEO --> Your internal Hard Disk Drive, HDD as Master on IDE0
- IDE1 --> Second Hard Disk Drive, CD-ROM as master on IDE1
- USB HDD --> External Hard Disk Drive connected to USB port (or USB Flash Drive with letter assigned C)
- USB FDD --> External Floppy Disk Driver connected to USB port (or USB Flash Drive with the letter assigned A)
- USB CDROM --> External CD/DVD drive connected to USB port

### SCSI Device Naming (New Hard Drives Names)

- SCSI drives follow a similar scheme, but all start with /dev/sd. When you run out of letters (after /dev/sdz), you can continue with /dev/sdaa and /dev/sdab and so on.

| device | scsi id | device name |
|---|---|---|
| disk 0 | 0 | /dev/sda |
| disk 1 | 1 | /dev/sdb |
| raid controller 0 | 5 | /dev/sdc |
| raid controller 1 | 6 | /dev/sdd |




**Tips:**

In general, the letters (fd, sd, hd) refer to the device type ('SATA, SCSI/SATA, IDE'), the third letter is for the device order (a the first, b the second, etc) and the numbers refer to the partitions the device has, starting by zero.

- hd refers to an IDE-type drive
- sd refers to a SCSI drive in general, but is mostly popular for SATA drives and CD/DVD
- fd is floppy disk
- Examle;
  - the first (1) partition on your first (a) SATA drive is /dev/sda1
  - The third (3) partition on your second (b) SATA drive is /dev/sdb3
  - the second partition (2) of the second (b) IDE hard disk is '/dev/hdb2'

> Linux disks and partition names may be different from other operating systems. You need to know the names that Linux uses when you create and mount partitions. Here's the basic naming scheme:

- The first floppy drive is named /dev/fd0.
- The second floppy drive is named /dev/fd1.
- The first hard disk detected is named /dev/sda.
- The second hard disk detected is named /dev/sdb, and so on.
- The first SCSI CD-ROM is named /dev/scd0, also known as /dev/sr0.

> The partitions on each SCSI disk are represented by appending a decimal number to the disk name: sda1 and sda2 represent the first and second partitions of the first SCSI disk drive in your system.

## lsblk Command

The lsblk (list blok devices) Linux command is a useful command which lists information about all or the specified block devices, however, it does not list information about RAM disks.

> Examples of block devices are a hard disk, flash drives, CD-ROM e.t.c

## Install lsblk

- **Ubuntu and Linux Mint installation** : `sudo apt-get install util-linux -y`
- **Fedora and CentOS installation** : `sudo yum install util-linux-ng`

## Use lsblk command

```
$ lsblk
NAME                      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0                       7:0    0 88.5M  1 loop /snap/core/7270
loop1                       7:1    0 88.7M  1 loop /snap/core/7396
sda                         8:0    0  7.3T  0 disk
├─sda1                      8:1    0    1M  0 part
├─sda2                      8:2    0    1G  0 part /boot
└─sda3                      8:3    0  7.3T  0 part
  └─ubuntu--vg-ubuntu--lv 253:0    0  7.3T  0 lvm  /
sdb                         8:16   0  7.3T  0 disk
└─sdb1                      8:17   0  7.3T  0 part
sr0                        11:0    1 1024M  0 rom
```

There are seven columns namely:

- NAME : This is the device name.
- MAJ:MIN : This column shows the major and minor device number.
- RM : This column shows whether the device is removable or not. Note in this example the device sdb and sr0 have their RM values equals to 1 indicating they are removable.
- SIZE : This is column give information on the size of the device.
- RO : This indicates whether a device is read-only. In this case all devices have a RO=0, indicating they are not read only.
- TYPE :This column shows information whether the block device is a disk or a partition(part) within a disk. In this example, sda and sdb are disks while sr0 is a read only memory (rom).
- MOUNTPOINT : This column indicates mount point on which the device is mounted.

## Linux File System

> The Linux filesystem is responsible for storing your system data and managing them. A filesystem can be defined as the mechanism behind data storing and retrieval. Filesystems are usually comprised of several layers, including a logical layer that provides user interaction, APIs for different file operations, and such.

## Common File Systems

> Linux supports numerous file systems, but common choices for the system disk on a block device include the ext* family;

- **ext2 and ext3**

The disadvantage is that file system checks on ext2 can take a long time. ext2 was being replaced by ext3 on most Linux machines. They are essentially the same, except for the journaling which is only present in ext3.

- **ext4**

The newest incarnation of the ext file system is named ext4 and is available in the Linux kernel since 2008. ext4 supports larger files (up to 16 terabyte) and larger file systems than ext3 (and many more features). Development started by making ext3 fully capable for 64-bit. When it turned out the changes were significant, the developers decided to name it ext4.

- **xfs**

Redhat Enterprise Linux 7 will have XFS as the default file system. This is a highly scalable high-performance file system. xfs was created for Irix and for a couple of years it was also used in FreeBSD. It is supported by the Linux kernel, but rarely used in distributions outside of the Redhat/CentOS realm.

- **iso 9660, udf, swap, gfs, and more**

## /proc/filesystems

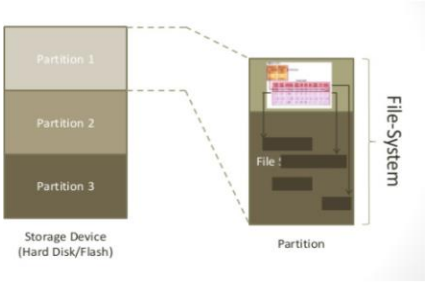
The /proc/filesystems file displays a list of supported file systems. When you mount a file system without explicitly defining one, then mount will first try to probe /etc/filesystems and then probe /proc/filesystems for all the filesystems without the nodev label. If /etc/filesystems ends with a line containing only an asterisk (*) then both files are probed.

> **Tips:** The three steps to create a file system don't change much from one operating system to another, but the specific details and utilities used vary greatly. The steps are:
> - Prepare the device (e.g. a disk, USB key, etc.) to receive a file system(fdsik)
> - Create the file system inside the partition (mkfs)
> - Mount the new file system to make it visible to the users (mount)
>
> The trio is always partition, make file system, and mount.

## Disk Partition

> A partition is a section of a physical disk. It can be all or part of the disk. It is usually formatted to contain a file system.

- A hard disk can be divided into several partitions. Each partition functions as if it were a separate hard disk. The idea is that if you have one hard disk, and want to have, say, three operating systems on it, you can divide the disk into three partitions. Each operating system uses its partition as it wishes and doesn't touch the other ones. This way the three operating systems can co-exist peacefully on the same hard disk. Without partitions, one would have to buy a hard disk for each operating system.




- Floppies are not usually partitioned. There is no technical reason against this, but since they're so small, partitions would be useful only very rarely. CD-ROMs are usually also not partitioned, since it's easier to use them as one big disk, and there is seldom a need to have several operating systems on one.

> Creating and deleting partitions in Linux is a regular practice because storage devices (such as hard drives and USB drives) must be structured in some way before they can be used. In most cases, large storage devices are divided into separate sections called partitions. Partitioning also allows you to divide your hard drive into isolated sections, where each section behaves as its own hard drive. Partitioning is particularly useful if you run multiple operating systems.

## `fdisk` Command

> `fdisk` is the universal command-line partition table manipulator for Linux:

```
fdisk [options] device
```

- This command is responsible for displaying and management of disk partitions.
- Using this command you can, Display disk partitions, Create new partitions, Delete existing partitions, Change the size of existing partitions.

> **Tips:** fdisk command does not support GPT partition table format, for that use the parted command

```
ubuntu@clarusway:~$ sudo fdisk -l
Disk /dev/loop0: 89.1 MiB, 93417472 bytes, 182456 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop1: 18 MiB, 18853888 bytes, 36824 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop2: 91.4 MiB, 95805440 bytes, 187120 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop3: 18 MiB, 18857984 bytes, 36832 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/xvda: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xb32e823c

Device     Boot Start      End  Sectors Size Id Type
/dev/xvda1 *     2048 16777182 16775135   8G 83 Linux
```

mkfs Command

## Creating Filesystems

The mkfs (build a Linux file system) command makes file systems. On other operating systems, creating a file system is called formatting. Regardless of its name, it is the process that prepares a partition so that it can store data.

> The mkfs command makes file systems. On other operating systems, creating a file system is called formatting.

```
mkfs [options] device
```

> ⚠ **Avoid ! :**
> While executing this command, as this will destroy all data on the device.

- Linux supports a number of file system types, some of which are described as follows.

| Filesystem | Description |
|---|---|
| ext2 | High performance for fixed disk and removable media |
| ext3 | Journaling version of ext2 |
| ext4 | Supports larger files and file system sizes |
| vfat | MS-DOS file system useful when sharing files between Windows and Linux |
| XFS | High-performance journaling file system |
| Btrfs | Addresses scalability requirements of large storage systems |

- The mkfs command is actually a front end for the different file system builder utilities such as mkfs.ext2 and mkfs.ext4.
- When using the mkfs wrapper, include the -t fstype option to specify the type of file system to be built. If not specified, the default file system type, ext2, is created.

> Example: To create an ext3 file system, use any of the following commands:

```
# mkfs -t ext3 /dev/xvdd1
# mke2fs -t ext3 /dev/xvdd1
# mkfs.ext3 /dev/xvdd1
```

> Example: We will build a Linux ext2 filesystem on /dev/sdb1 partition.

```
# mkfs /dev/sdb1
mke2fs 1.41.12 (17-Mar-2020)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
6168576 inodes, 24657920 blocks
1232896 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
753 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 21 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

> Example: When the above checking for bad blocks is happening, if you try to do the following from another terminal, it will says that the device is used by the system.

```
# mkfs /dev/sdb1
mke2fs 1.41.12 (17-Mar-2020)
/dev/sdb1 is apparently in use by the system; will not make a filesystem here!
```

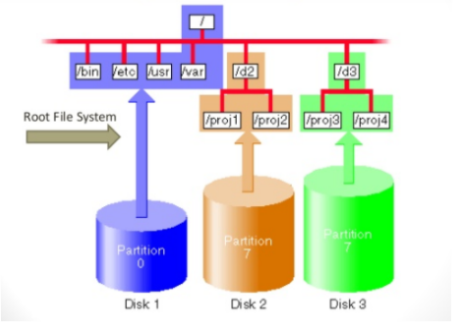
## Mounting

- Mounting a file system makes it available for use, usually as a directory.

> Mounting is the attaching of an additional filesystem to the currently accessible filesystem of a computer.

- To access a file, you need to know the full path starting from the root directory. When adding a file system to your computer, you need to make it available somewhere in the file tree. The directory where you make a file system available is called a mount point.

> 💡 **Tips:** A filesystem is a hierarchy of directories (also referred to as a directory tree) that is used to organize files on a computer or storage media (e.g., a CDROM)




- All partitions are attached to the system via a mount point. The mount point defines the place of a particular data set in the file system. Usually, all partitions are connected through the root partition. On this partition, which is indicated with the slash (/), directories are created. These empty directories will be the starting point of the partitions that are attached to them.

> - The mount command to attach (mount) file systems and removable devices such as USB flash drives at a particular mount point in the directory tree.
> - The umount command detaches (unmounts) the mounted file system from the directory tree.

> The standard form of the mount command is:

```
mount -t type device dir
```

This tells the kernel to attach the filesystem found on device (which is of type type) at the directory dir. The previous contents (if any), owner, and mode of dir become invisible, and as long as this filesystem remains mounted, the pathname dir refers to the root of the filesystem on device.

> For example, you have added a disk /dev/sdb on on your system. Now you want to mount this on /data directory. Use following command to mount it.

| Description | Windows | Linux |
|---|---|---|
| Partition | Disk1 | /dev/sda1 |
| Filesystem type | NTFS/FAT32 | EX3/EXT/XFS |
| Mounting PArameters | Drive Letter | Mount Point |
| Base Folder (OS is stored) | C drive | / |

> Mount command automatically detects the file system on disk. But in some cases, you need specify the file system type with command.

```
$ mount -t ext4 /dev/sdb /data
```

**unmount**

Use umount command to unmount any mounted filesystem on your system. Run umount command with disk name or mount point name to unmount currently mounted disk.

```
umount DIRECTORY
umount DEVICE_NAME
```

## Mounting File Systems

**mkdir**

- This example shows how to create a new mount point with mkdir

```
mkdir /home/clarusway
```

**mount**

- When the mount point is created, and a file system is present on the partition, then mount can mount the file system on the mount point directory.

```
mount -t ext2 /dev/sdb1 /home/clarusway/
```

**/etc/filesystems**

Actually the explicit -t ext2 option to set the file system is not always necessary. The mount command is able to automatically detect a lot of file systems. When mounting a file system without specifying explicitly the file system, then mount will first probe /etc/filesystems. Mount will skip lines with the nodev directive.

```
cat /etc/filesystems
ext3
ext2
nodev proc
nodev devpts
iso9660
vfat
hfs
```

**/proc/filesystems**

- When /etc/filesystems does not exist, or ends with a single * on the last line, then mount will read /proc/filesystems.

```
cat /proc/filesystems | grep -v ^nodev
ext2
iso9660
ext3
```

**umount**

- You can unmount a mounted file system using the umount command.

```
umount /home/reet
```

## Displaying Mounted File Systems

- To display all mounted file systems, issue the mount command. Or look at the files /proc/mounts and /etc/mtab.

**mount**

- The simplest and most common way to view all mounts is by issuing the mount command without any arguments.

```
mount | grep /dev/sdb
/dev/sdb1 on /home/project42 type ext2 (rw)
```

**/proc/mounts**

- The kernel provides the info in /proc/mounts in file form, but /proc/mounts does not exist as a file on any hard disk. Looking at /proc/mounts is looking at information that comes directly from the kernel

```
cat /proc/mounts | grep /dev/sdb
/dev/sdb1 /home/project42 ext2 rw 0 0
```

**/etc/mtab**

- The /etc/mtab file is not updated by the kernel, but is maintained by the mount command. Do not edit /etc/mtab manually

```
cat /etc/mtab | grep /dev/sdb
/dev/sdb1 /home/project42 ext2 rw 0 0
```

**df**

- A more user friendly way to look at mounted file systems is df. The df (diskfree) command has the added benefit of showing you the free space on each mounted disk. Like a lot of Linux commands, df supports the -h switch to make the output more human readable.

```
df
```

**df -h**

- In the df -h example below you can see the size, free space, used gigabytes and percentage and mount point of a partition.

```
df -h | egrep -e "(sdb2|File)"
```

**du**

- The du command can summarize disk usage for files and directories. By using du on a mount point you effectively get the disk space used on a file system.

```
du -sh /boot /home
```

## File Ownership

**User Owner and Group Owner**

- The users and groups of a system can be locally managed in /etc/passwd and /etc/group, or they can be in a NIS, LDAP, or Samba domain. These users and groups can own files. Actually, every file has a user owner and a group owner, as can be seen in the following

```
user@clarusway:~$  ls -lh
```

```
clarus@DESKTOP-4QQ1SSL: /etc/network
clarus@DESKTOP-4QQ1SSL:/etc/network$ ls -lh
total 0
drwxr-xr-x 1 root root 4.0K Apr 27  2018 if-down.d
drwxr-xr-x 1 root root 4.0K Apr 27  2018 if-post-down.d
drwxr-xr-x 1 root root 4.0K May 21  2019 if-pre-up.d
drwxr-xr-x 1 root root 4.0K May 21  2019 if-up.d
-rw-r--r-- 1 root root  190 May 21  2019 interfaces
drwxr-xr-x 1 root root 4.0K Apr 27  2018 interfaces.d
clarus@DESKTOP-4QQ1SSL:/etc/network$
```

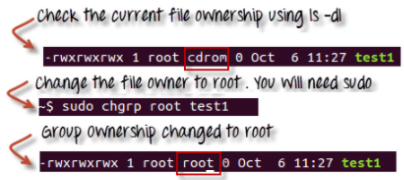
**Listing User Accounts**

- You can use the following command to list all local user accounts.

```
user@clarusway:~$ cut -d: -f1 /etc/passwd | column
```

**chgrp**

- You can change the group owner of a file using the chgrp command.

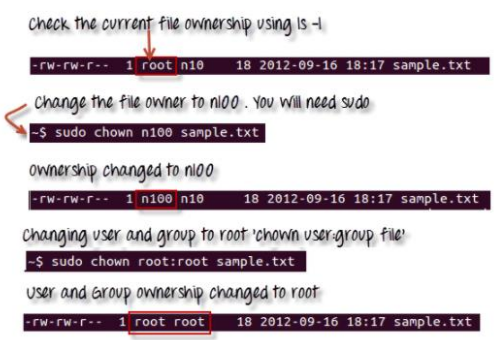



```
root@clarusway:/home/oliver# ls -l file2
-rw-r--r--. 1 root tennis 185 Apr 8 18:46 file2
root@clarusway:/home/oliver# chgrp snooker file2
root@clarusway:/home/oliver# ls -l file2
-rw-r--r--. 1 root snooker 185 Apr 8 18:46 file2
root@clarusway:/home/oliver#
```

Tips:
chgrp stands for change group.

**chown**

- The user owner of a file can be changed with chown command.




```
root@clarusway:/home/oliver#  ls -l FileForOliver
-rw-r--r-- 1   root   oliver  0   2020-03-14 14:11 FileForOliver
root@clarusway#:/home/oliver# chown oliver FileForOliver
root@clarusway:/home/oliver# ls -l FileForOliver
-rw-r--r-- 1   oliver  oliver 0 2020-03-14 14:11 FileForOliver
```

Tips:
- The file /etc/group contains all the groups defined in the system
- You can use the command "groups" to find all the groups
- You cannot have 2 groups owning the same file.

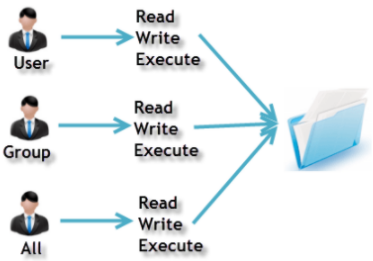
> Q: Explain Linux file ownership.
> A: Every Linux system have three types of owner:

- User: A user is the one who created the file. By default, whosoever, creates the file becomes the owner of the file. A user can create, delete, or modify the file.
- Group: A group can contain multiple users. All the users belonging to a group have same access permission for a file.
- Other: Any one who has access to the file other than user and group comes in the category of other. Other has neither created the file nor is a group member.

— - Interview Q&A

## Permissions

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




**rwx**

- The nine characters following the file type denote the permissions in three triplets. A permission can be r for read access, w for write access, and x for execute. You need the r permission to list (ls) the contents of a directory. You need the x permission to enter (cd) a directory. You need the w permission to create files in or remove files from a directory
- Standard File Permissions

| Permission | On a File | On a Directory |
|---|---|---|
| (read) read | file contents (cat) | read directory contents (ls) |
| w (write) | change file contents (vi) | create files in (touch) |
| x (execute) | execute the file | enter the directory (cd) |

**three sets of rwx**

- When you are the user owner of a file, then the user owner permissions apply to you. The rest of the permissions have no influence on your access to the file. When you belong to the group that is the group owner of a file, then the group owner permissions apply to you. The rest of the permissions have no influence on your access to the file. When you are not the user owner of a file and you do not belong to the group owner, then the others permissions apply to you. The rest of the permissions have no influence on your access to the file.
- File Permissions Position

| Position | Characters | function |
|---|---|---|
| 1 | - | this is a regular file |
| 2-4 | rwx | permissions for the user owner |
| 5-7 | r-x | permissions for the group owner |
| 8-10 | r-- | permissions for others |

## Setting Permissions (chmod)

- Permissions can be changed with chmod.

There are 2 ways to use the chmod command:
- Symbolic mode
- Absolute mode

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

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

> The various owners:

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

## Absolute(Numeric) Mode Permission

File permissions are not represented as characters but a three-digit octal number.

| binary | octal | permission |
|---|---|---|
| 000 | 0 | --- |
| 001 | 1 | --x |
| 010 | 2 | -w- |
| 011 | 3 | -wx |
| 100 | 4 | r-- |
| 101 | 5 | r-x |
| 110 | 6 | rw- |
| 111 | 7 | rwx |

> Example: This makes 777 equal to rwxrwxrwx and by the same logic, 654 mean rw-r-xr-- . The chmod command will accept these numbers.

> Q: How to change file permissions in linux?
> A: I can change permissions from command line with **chmod** command. **Read (r), write (w) and execute (x)** are permission types that can be given to a file. You can also indicate these permissions with numeric mode. For example **777** gives full access permission (rwx).
>
> — - Interview Q&A