# Session 1: Introduction to Linux (3 Hrs.)

# Content (1.5 Hrs.)

## Agenda:

1. What is operating system?
2. What is kernel?
3. Is Linux an operating system or a kernel?
4. Is Linux Operating System Immune to Malware?
5. Disk and Partition Type
   5.1 What is basic disk?
   5.2 What is dynamic disk?
6. What is the difference between UEFI and Legacy Mode?

## 1. What is operating system?

An operating system is a set of software that makes a computer work. It is the back bone of all software in a computer, allowing other software to be installed and executed. Without an operating system, you cannot use a computer.

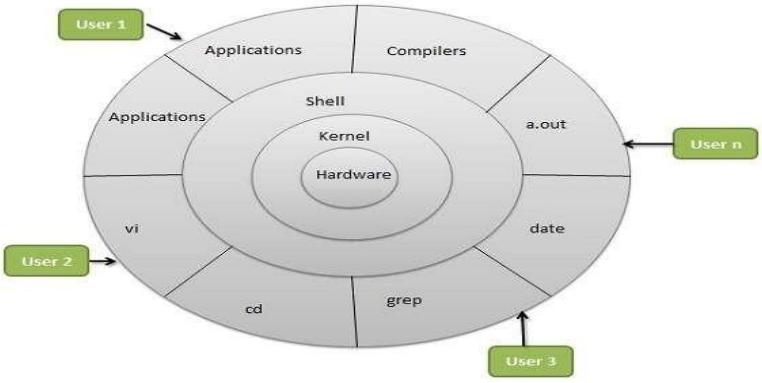
## 2. What is kernel?

A kernel is the most important element of an operating system. For an OS to work, it needs a kernel. The task of the kernel is to handle hardware resources (memory allocation, disk access, device usage...). It sits in between the hardware and the software being executed.

## 3. Is Linux an operating system or a kernel?

Linux is only a kernel. On top of that, you can run a lot of things. The kernel plus the things you run over the kernel is the operating system.

Let me show a diagram:



This is how any modern operating system works: you have the kernel dealing with the hardware, and the shell dealing with you, the user, and the applications you launch.

So, an operating system is Debian, CentOS, Ubuntu, Redhat. All of them use the Linux kernel, so we call them Debian Linux, CentOS Linux, Ubuntu Linux, or RedHat Linux.

## 4. Is Linux Operating System Immune to Malware?

To be true, No! No OS on this earth can be ever be 100% immune to Viruses and Malware. But still Linux never had a widespread malware-infection as compared to Windows. Why? Let us find the reason behind this.

Some people believes that Linux still has a minimal usages share, and a Malware is aimed for mass destruction. No programmer will give his valuable time, to code day and night for such group and hence Linux is known to have little or no viruses. Had it been true, Linux should be the primary target of Malware infection because more than 90% of high end server runs on Linux today.

Destroying or Infecting one server means collapse of thousands of computer and then Linux would have been the soft target of hackers. So, certainly usages share ratio is not in consideration for the above said fact.

Linux is architecturally strong and hence very much immune (not totally) to security threats. Linux is Kernel and GNU/Linux is the OS. There are hundreds of distributions of Linux. At Kernel Level they all are more or less the same but not at the OS Level.

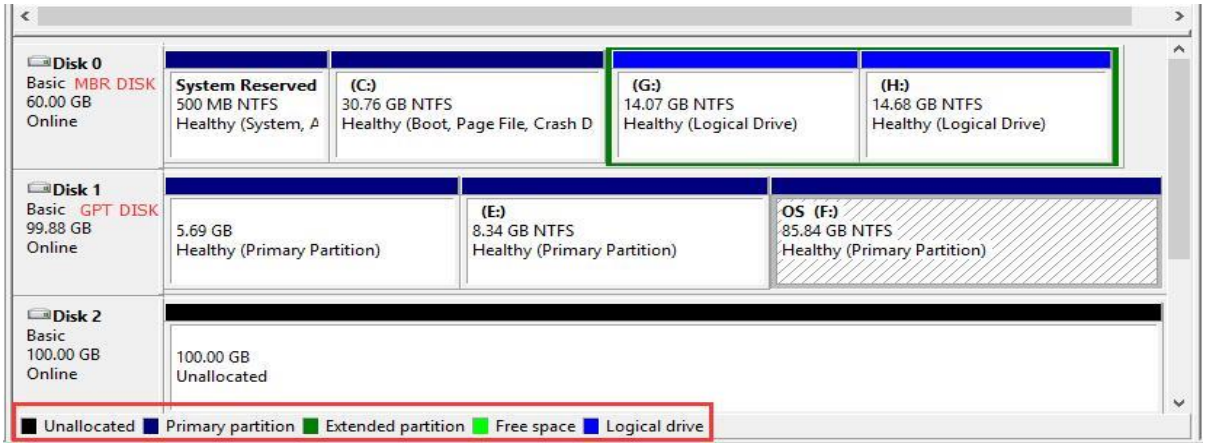
## 5. Disk and Partition Type

**Basic disks and dynamic disks are 2 types of hard disk configurations** most often used in Windows. The former appears along with hard disk while the latter was introduced since Windows 2000. Different disk configurations have different features, but they are related.

### 5.1 What is Basic Disk?

Basic disk uses partitions to manage data, and one partition cannot share and split data with other partitions (the late versions of Windows OS like Windows 7, Windows 8 and Windows 10 also call partitions volumes).

**On a basic disk, we can create 2 styles of partitions, namely MBR style partition and GPT style partition.** To create MBR style partitions, we need to initialize the hard disk to MBR (master boot record). To create GPT style partitions, we should initialize the disk to GPT (GUID partition table).



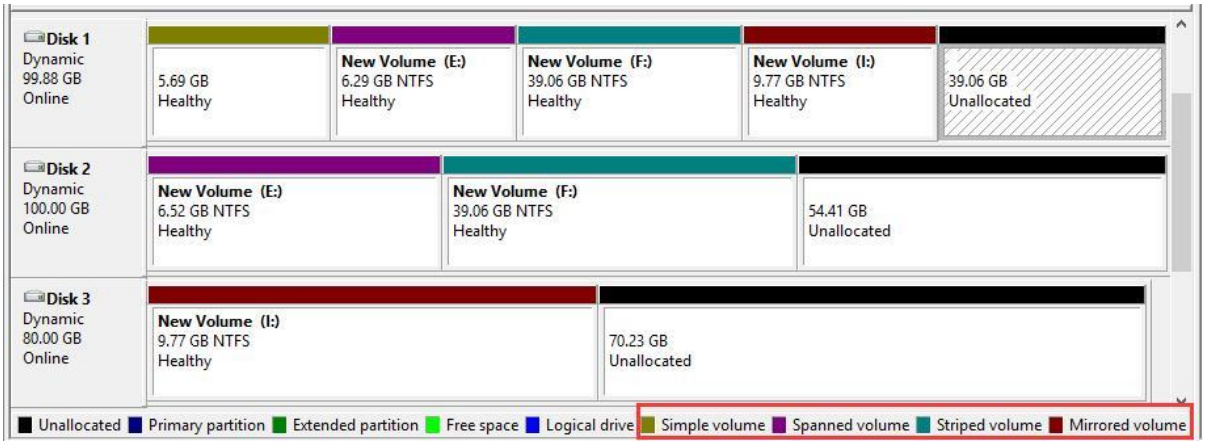

On MBR disks, partitions are called primary partition, extended partition, or logical partition while all partitions on GPT disk are called GPT partitions which function like primary partitions. **An MBR based basic disk can have either four primary partitions OR three primary plus one extended partition, but the extended partition can contain an unlimited number of logical drives. And a GPT based hard disk can hold up to 128 partitions.**

We can extend a primary partition by adding contiguous unallocated space on the same disk or extend a logical drive by adding free space contained in extended partition, but these partitions should be formatted with NTFS file system. Otherwise, Extend Volume feature is grayed out.

## 5.2 What is dynamic disk?

Dynamic disk uses dynamic volumes to manage data. It is a separate form of volume management that allows one volume to have noncontiguous extents on one or more physical disks. Dynamic disks use a database to track information about all volumes on the disk as well as information about other dynamic disks, and the location of the database is determined by partition style of the disk.

On MBR partitions, the database is contained in the last 1 megabyte (MB) of the disk while database on GPT partitions is located in a 1MB reserved partition. In addition, each dynamic disk in a computer saves a copy of the database, so that Windows can repair a damaged database by using the database on other dynamic disks.




On a dynamic disk, we can create 5 types of dynamic volumes to enhance computer performance, including simple volume, mirrored volume, striped volume, spanned volume, and RAID-5 volume.

- Simple volume functions like primary partitions on basic disk;
- Mirrored volume provides fault tolerance by creating a copy of data contained in this volume;
- Striped volume improves disk input/output performance by distributing I/O requests across disks;
- Spanned volume combines spaces on 2 hard disks at least to a dynamic volume;
- RAID-5 volume stripes data and parity across three or more disks.

## 6. What is the difference between UEFI and Legacy Mode?

UEFI (Unified Extensible Firmware Interface) is the successor to BIOS. UEFI uses the GUID Partition Table (GPT) whereas BIOS uses the Master Boot Record(MBR) partitioning scheme. GPT and MBR are both formats specifying physical partitioning information on the hard disk. Below I have listed the difference:

- Max partition size in MBR is ~2TB whereas in UEFI it is ~9 ZetaBytes.
- MBR can have at max 4 primary partition whereas GPT can have 128.
- MBR can store only one bootloader whereas GPT has a separate dedicated EFI System Partition(ESP) for storing multiple bootloaders. This is very helpful if you have two or more operating systems which require different bootloaders.
- UEFI offers secure boot which can prevent boot-time viruses from loading.

BIOS is pretty much outdated and UEFI offers many useful features. Thus it is recommended to install any operating system in UEFI mode. Note: One can't install in UEFI mode by booting in legacy mode.

# Task (1.5 hrs.)

1. What is operating system?
2. What is kernel?
3. Is Linux an operating system or a kernel?
4. Is Linux Operating System Immune to Malware?
5. How many disk types have?
6. What is basic disk type?
7. What is dynamic disk type?
8. How many partition types have?
9. What is MBR partition?
10. What is GPT partition?
11. How many BIOS mode have?
12. What is UEFI Mode?
13. What is BIOS/legacy mode?
14. What is the difference between UEFI and Legacy Mode?
15. BIOS stands for_____?
16. MBR stands for _____?
17. GPT stands for_____?
18. UEFI stands for_____?

# Session 2: Linux Installation (3 Hrs.)

## Content (2 Hrs.)

## Agenda:

1. What is dual boot?
2. What is Swap area?
3. What is Mount point?
4. What is ext4?
5. Ubuntu installation process?

### 1. What is dual boot?

A dual boot system is a computer system in which two operating systems are installed on the same hard drive, allowing either operating system to be loaded and given control. When you turn the computer on, a boot manager program displays a menu, allowing you to choose the operating system you wish to use.

## 2. What is Swap area?

Swap space is the area on a hard disk which is part of the Virtual Memory of your machine, which is a combination of accessible physical memory (RAM) and the swap space. Swap space temporarily holds memory pages that are inactive.

Swap space is used when your system decides that it needs physical memory for active processes and there is insufficient unused physical memory available.

**In a word,** The Swap Area is just fake RAM that lives on your hard drive. It is much slower than actual RAM, but is necessary in many cases to keep a computer running normally.

## 3. What is mount point?

A mount point is a directory (typically an empty one) in the currently accessible filesystem on which an additional filesystem is mounted.

## 4. What is ext4?

The ext4 or fourth extended filesystem is a journaling file system for Linux. Ext4 is the evolution of the most used Linux filesystem, Ext3. In many ways, Ext4 is a deeper improvement over Ext3 than Ext3 was over Ext2. Ext3 was mostly about adding journaling to Ext2, but Ext4 modifies important data structures of the filesystem such as the ones destined to store the file data.

## 5. Ubuntu installation process

1. **Download latest LTS Ubuntu version from official ubuntu web site.**

2. **Check BISO mode** – if it is legacy mode that's good but if it is UEFI mode then you have to change BIOS mode or you have to boot the ubuntu ISO image file in UEFI compatible (You can use Rufus USB installer).

3. **Check disk type** – if disk type basic that's good for dual boot ubuntu installation but if it is dynamic than you have to convert disk type dynamic to basic or you have to format the Hard Disk Drive (HDD).

4. **Check partition type** - if disk type MBR that's good for dual boot ubuntu installation but if partition type GPT then you have to boot the ubuntu ISO image file in GPT compatible (You can use Rufus USB installer).

5. **Partition an unallocated space (min 25 GB) for ubuntu OS installation.**

6. **Boot ubuntu in USB** –

   **Note:** Here we can use universal USB installer or Rufus.

   i. Select USB drive
   ii. Select ISO image file
   iii. Select file system/file format FAT32
   iv. Select partition scheme and target system type in Rufus (**if the BIOS mode UEFI or partition type GPT this selection is necessary**)
   v. Select start button for boot

7. **Final Steps –**

   i. Put recently bootable USB drive on our laptop's then restart your laptop.

   ii. On startup time select boot option key on your laptop (this can vary with different laptop manufacturer) then select USB/Jet flash drive.

   iii. After selecting USB drive you will see the ubuntu is loading and after finishing loading ubuntu you have to select install ubuntu option from given options.

   iv. Then select your language, location etc.

   v. Then when you will see "Installation type" window, here you have to select "something else" option from given options because we want to install ubuntu in dual boot.

vi. Then select your free space partition and create swap area, here size of swap will be double of RAM size or equal to RAM size or minimum 2 GB.

vii. Then again select your free space partition and select file system ext4 (maybe by default it is ext4) then select mount point "/" ("/" means root).

viii. Then select "install now" button to start installation process.

**Note:** During installation please give an overview about disk type, partition type, BIOS mode, swap area, mount point and ext4.

## Task (1 Hrs.)

1. What is dual boot?
2. What is Swap area?
3. What is Mount point?
4. What is file system?
5. What is ext4?
6. How many disk types have?
7. What is basic disk?
8. What is dynamic disk?
9. Which disk type is being used in your system?
10. What is the problem of dynamic disk during ubuntu installation?
11. How many partition types have?
12. What is MBR partition?
13. What is GPT partition?
14. Which partition type is being used in your system?
15. How many BIOS mode have?
16. What is UEFI Mode?
17. What is BIOS/legacy mode?
18. Which BIOS mode is being used in your system?
19. What is the difference between UEFI and Legacy Mode?
20. What is benefit of LTS ubuntu version?
21. BIOS stands for_____?
22. MBR stands for _____?
23. GPT stands for_____?
24. UEFI stands for_____?
25. LTS stands for _____?

# Session 3: Introduction to Shell, User and User Account, Course Project discussion (3 Hrs.)

## Content (1.5 Hrs.)

## Agenda:

1. What is the Shell?
2. What is a Terminal?
3. What defines a user account?
4. User account management?
5. Project discussion

## 1. What is the Shell?

Simply put, the shell is a program that takes commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix-like system such as Linux. Nowadays, we have graphical user interfaces (GUIs) in addition to command line interfaces (CLIs) such as the shell.

On most Linux systems a program called **bash** (which stands for Bourne Again SHell, an enhanced version of the original Unix shell program, **sh**, written by Steve Bourne) acts as the shell program. Besides **bash**, there are other shell programs that can be installed in a Linux system. These include: **ksh**, **tcsh** and **zsh**.

## 2. What is a Terminal?

It's a program called a terminal emulator. This is a program that opens a window and lets you interact with the shell. There are a bunch of different terminal emulators you can use. Most Linux distributions supply several, such as: **gnome-terminal**, **konsole**, **xterm**, **rxvt**,**kvt**, **nxterm**, and **eterm**.

## 3. What defines a user account?

Linux is multi-user system. This means more than one person can use the Linux. With the help of various software servers, configurations, and commands, multiple users can use Linux.

In order to gain access to the system and its resources, users are required to log in. By controlling access to system, you can prevent unauthorized users from using system as well as control access to data.

Most modern Linux distribution creates more than two user accounts when you install first time.

- The first is root user. The root is superuser (you can compare this account with Administrator account under MS-Windows 2000/2003/XP server). The root has all rights to all files, system services and software's.
- The second user is a normal (general-purpose) user account that you name. It has limited access to Linux.

***More on root**

Root account has full access to system so it is recommended that you do not use root it unless you have to. When you need to perform system level administrative task (such as adding new users, installing software etc) they you have to use the root account using su or sudo command. If you are new Linux, admin and use the root for all activity then senior (or coworkers) will make fun out of you.

## 4. User account management?

- `sudo` - execute a command as another user (default as root)
- `su` - switch user (default to root)
- `sudo adduser {username}` – to create new user account
- `sudo login` – to login/switch into a account
- `logout` – to logout from current account
- `deluser {username}` – to delete a user , you have to enter into root (use `sudo su` to enter into root) for delete a user
- `exit` – to exit from root

## 5. Project discussion

- **Recommended project list:**

1. Setting up a Mail Server
2. Setting up a DHCP(Dynamic Host Configuration Protocol)
3. Setting up a Web Server
4. Setting up a DNS(Domain Name System) Server
5. Setting up a Print Server
6. Setting up a NFS(Network File System) Server
7. Setting up a Samba Server
8. Setting up a Proxy server
9. Android application

10. Windows application
11. Shell command based work

**Note:** Use project presentation slide for describe these project list.

## Task (1.5 Hrs.)

1. What is shell?
2. What is terminal?
3. What is user friendly between terminal and GUI & why?
4. What defines a user account?
5. What root user?
6. What is general purpose user?
7. What are the difference between root user and general purpose user?
8. What is the meaning of sudo?
9. What is the meaning of su?
10. Why we use sudo?
11. Why we use su?
12. How to create a user account?
13. How to delete a user?
14. How to login into a user account?
15. How to logout from a user account?
16. How many times required password for creating a user account?
17. How to exit from root?
18. What is the meaning of `sudo adduser {username}`?
19. What is the meaning of `deluser {username}`?
20. What is the meaning of `sudo su`?

# Session 4: Files and Directory Commands (3 Hrs.)

## Content (1.5 Hrs.)

### Agenda:

1. What are Commands
2. Details of some useful Files and Directory Commands
3. Examples of some useful Files and Directory Commands

### 1. What are Commands

A command is an instruction given to our computer by us to do whatever we want. In Mac OS, and Linux it is called terminal, whereas, in windows it is called command prompt. Commands are always case sensitive.

Commands are executed by typing in at the command line followed by pressing enter key.

This command further passes to the shell which reads the command and execute it. Shell is a method for the user to interact with the system. Default shell in Linux is called bash (Bourne-Again Shell).

There are two types of shell commands:

- **Built-in shell commands:** They are part of a shell. Each shell has some built in commands.
- **External/Linux commands:** Each external command is a separate executable program written in C or other programming languages.

### 2. Details of some useful Files and Directory Commands

- `Who` – User details
- `Whoami` – current user name (type)
- `ls` - This command '**ls**ts' the contents of your present working directory.
- `pwd` - Shows you what your **p**resent **w**orking **d**irectory is.
- `cd` - Lets you **c**hange **d**irectories.
- `rm` - **R**e**m**oves one or more files.
- `rmdir` - **R**e**m**ove an empty **dir**ectory.
- `mkdir` - **M**a**k**e a **dir**ectory.
- `ps` - Provides a list of currently running **p**rocesse**s**.
- `cp` - **C**o**p**y a file.

- `mv` - **Mov**e a file (this is also used to rename a file, "moving" it from one file name to another.)
- `find` - Find a file on the filesystem (100% accurate, but not fast).
- `locate` - Find a file on the filesystem from a cached list of files (Fast, but not 100% accurate).
- `man` - Displays the **man**ual for most commands (including 'man').
- `clear` - clear the screen
- `less` - view the contents of a file
- `sudo` - execute a command as another user (default as root)
- `su` - switch user (default to root)

## 3. Examples of some useful Files and Directory Commands

| Command | Description |
|---|---|
| `# cd /home` | enter to directory '/ home' |
| `# cd ..` | go back one level |
| `# cd ../..` | go back two levels |
| `# cd` | go to home directory |
| `# cd ~user1` | go to home directory |
| `# cd -` | go to previous directory |
| `# cp file1 file2` | copying a file |
| `# cp dir/* .` | copy all files of a directory within the current work directory |
| `# cp -a /tmp/dir1 .` | copy a directory within the current work directory |
| `# cp -a dir1 dir2` | copy a directory |
| `# cp file file1` | outputs the mime type of the file as text |
| `# ls` | view files of directory |
| `# ls -F` | view files of directory |
| `# ls -l` | show details of files and directory |
| `# ls -a` | show hidden files |
| `# ls *[0-9]*` | show files and directory containing numbers |
| `# mkdir dir1` | create a directory called 'dir1' |
| `# mkdir dir1 dir2` | create two directories simultaneously |
| `# mv dir1 new_dir` | rename / move a file or directory |

| | |
|---|---|
| `# pwd` | show the path of work directory |
| `# rm -f file1` | delete file called 'file1' |
| `# rm -rf dir1` | remove a directory called 'dir1' and contents recursively |
| `# rm -rf dir1 dir2` | remove two directories and their contents recursively |
| # rmdir dir1 | delete directory called 'dir1' |

# Task (1.5 Hrs.)

1. What is Command?
2. Why we use command?
3. What is the meaning of `who & whoami?`
4. What are the meaning of `cd, cd .. , cd ../.. & cd - ?`
5. What is the meaning of `pwd?`
6. What is the meaning of `mkdir?`
7. What is the meaning of `touch?`
8. What are the meaning of `rm , rm -f & rm -rf ?`
9. What are the meaning of `ls , ls -l & ls -a ?`
10. What is the meaning of `cp?`
11. What is the meaning of `mv?`
12. What is the meaning of `clear?`
13. What is the meaning of `less?`
14. What is the meaning of `sudo?`
15. What is the meaning of `su?`
16. How to create a directory folder?
17. How to create a document file?
18. How to delete a empty directory folder?
19. How to delete a non-empty directory folder?
20. How to delete a document file?

# Session 5: File Permissions (3 Hrs.)

## Content (1.5 Hrs.)

## Agenda:

1. Understanding and Using File Permissions
2. Changing Permissions
   2.1 chmod with Letters
   2.2 chmod with Numbers
   2.3 chmod with sudo

## 1. Understanding and Using File Permissions

In Linux and Unix, everything is a file. Directories are files, files are files and devices are files. Devices are usually referred to as a node; however, they are still files. All of the files on a system have permissions that allow or prevent others from viewing, modifying or executing. If the file is of type Directory then it restricts different actions than files and device nodes. The super user "root" has the ability to access any file on the system. Each file has access restrictions with permissions, user restrictions with owner/group association. Permissions are referred to as bits.

To change or edit files that are owned by root, **sudo** must be used.

If the owner read & execute bit are on, then the permissions are:

```
-r-x------
```

There are three types of access restrictions:

| **Permission** | **Action** | **chmod option** |
|---|---|---|
| read | (view) | r or 4 |
| write | (edit) | w or 2 |
| execute | (execute) | x or 1 |

There are also three types of user restrictions:

| **User** | *ls* **output** |
|---|---|
| owner | -rwx------ |
| group | ----rwx--- |
| other | -------rwx |

## 2. Changing Permissions

The command to use when modifying permissions is chmod. There are two ways to modify permissions, with numbers or with letters. Using letters is easier to understand for most people. When modifying permissions be careful not to create security problems. Some files are configured to have very restrictive permissions to prevent unauthorized access. For example, the /etc/shadow file (file that stores all local user passwords) does not have permissions for regular users to read or otherwise access.

```
user@host:/home/user# ls -l /etc/shadow
-rw-r-----  1 root shadow 869 2005-11-08 13:16 /etc/shadow
user@host:/home/user#

Permissions:
owner = Read & Write (rw-)
group = Read (r--)
other = None (---)

Ownership:
owner = root
group = shadow
```

## 2.1 chmod with Letters

```
Usage: chmod {options} filename
```

| **Options** | **Definition** |
|---|---|
| u | owner |
| g | group |
| o | other |

| | |
|---|---|
| a | all (same as ugo) |
| x | execute |
| w | write |
| r | read |
| + | add permission |
| - | remove permission |
| = | set permission |

**Here are a few examples of chmod usage with letters (try these out on your system).**

First create some empty files:

```
user@host:/home/user$ touch file1 file2 file3 file4
user@host:/home/user$ ls -l
total 0
-rw-r--r--  1 user user 0 Nov 19 20:13 file1
-rw-r--r--  1 user user 0 Nov 19 20:13 file2
-rw-r--r--  1 user user 0 Nov 19 20:13 file3
-rw-r--r--  1 user user 0 Nov 19 20:13 file4
```

Add owner execute bit:

```
user@host:/home/user$ chmod u+x file1
user@host:/home/user$ ls -l file1
-rwxr--r--  1 user user 0 Nov 19 20:13 file1
```

Add other write & execute bit:

```
user@host:/home/user$ chmod o+wx file2
user@host:/home/user$ ls -l file2
-rw-r--rwx  1 user user 0 Nov 19 20:13 file2
```

Remove group read bit:

```
user@host:/home/user$ chmod g-r file3
user@host:/home/user$ ls -l file3
-rw----r--  1 user user 0 Nov 19 20:13 file3
```

Add read, write and execute to everyone:

```
user@host:/home/user$ chmod ugo+rwx file4
user@host:/home/user$ ls -l file4
```

```
-rwxrwxrwx  1 user user 0 Nov 19 20:13 file4
user@host:/home/user$
```

### 2.2 chmod with Numbers

```
Usage: chmod {options} filename
```

| Options | Definition |
|---|---|
| #-- | owner |
| -#- | group |
| --# | other |
| 1 | execute |
| 2 | write |
| 4 | read |

Owner, Group and Other is represented by three numbers. To get the value for the options determine the type of access needed for the file then add.

**For example if you want a file that has -rw-rw-rwx permissions you will use the following:**

| Owner | Group | Other |
|---|---|---|
| read & write | read & write | read, write & execute |
| 4+2=6 | 4+2=6 | 4+2+1=7 |

```
user@host:/home/user$ chmod 667 filename
```

**Another example if you want a file that has --w-r-x--x permissions you will use the following:**

| Owner | Group | Other |
|---|---|---|
| write | read & execute | execute |
| 2 | 4+1=5 | 1 |

```
user@host:/home/user$ chmod 251 filename
```

**Here are a few examples of chmod usage with numbers (try these out on your system).**

First create some empty files:

```
user@host:/home/user$ touch file1 file2 file3 file4
user@host:/home/user$ ls -l
total 0
-rw-r--r--  1 user user 0 Nov 19 20:13 file1
-rw-r--r--  1 user user 0 Nov 19 20:13 file2
-rw-r--r--  1 user user 0 Nov 19 20:13 file3
-rw-r--r--  1 user user 0 Nov 19 20:13 file4
```

Add owner execute bit:

```
user@host:/home/user$ chmod 744 file1
user@host:/home/user$ ls -l file1
-rwxr--r--  1 user user 0 Nov 19 20:13 file1
```

Add other write & execute bit:

```
user@host:/home/user$ chmod 647 file2
user@host:/home/user$ ls -l file2
-rw-r--rwx  1 user user 0 Nov 19 20:13 file2
```

Remove group read bit:

```
user@host:/home/user$ chmod 604 file3
user@host:/home/user$ ls -l file3
-rw----r--  1 user user 0 Nov 19 20:13 file3
```

Add read, write and execute to everyone:

```
user@host:/home/user$ chmod 777 file4
user@host:/home/user$ ls -l file4
-rwxrwxrwx  1 user user 0 Nov 19 20:13 file4
user@host:/home/user$
```

## 2.3 chmod with sudo

Changing permissions on files that you do not have ownership of: (**Note** that changing permissions the wrong way on the wrong files can quickly mess up your system a great deal! Please be careful when using **sudo**!)

```
user@host:/home/user$ ls -l /usr/local/bin/somefile
-rw-r--r--  1 root root 550 2005-11-13 19:45 /usr/local/bin/somefile
user@host:/home/user$

user@host:/home/user$ sudo chmod o+x /usr/local/bin/somefile

user@host:/home/user$ ls -l /usr/local/bin/somefile
```

```
-rw-r--r-x  1 root root 550 2005-11-13 19:45 /usr/local/bin/somefile
user@host:/home/user$
```

# Lab Task (1.5 Hrs.)

Practical Task:

1. Create 2 files in Desktop, name them lab5 & lab6, then change their permission for owner have “read & execute”, group will have "write & read" & others can only do "execute". Do it in both numeric and character for lab5 & lab6 respectively.
2. Set the file permission null for all.

1. What is file permission?
2. Why we use file permissions?
3. What key command we use for file permissions?
4. How can you see the existing file permission of a file?
5. What is the basic format for changing file permission of a file?
6. What is the meaning of chmod u/g/o+r/w/x?
7. What is the meaning of chmod u/g/o-r/w/x?
8. What is the meaning of chmod ugo+r/w/x?
9. What is the meaning of chmod ugo-r/w/x?
10. What is the meaning of chmod u/g/o-r+x?
11. What is the meaning of chmod u/g/o+rw-x?
12. What is the meaning of chmod u/g/o-rwx?
13. What is the meaning of chmod a+r/w/x?
14. What is the meaning of chmod a-r/w/x?
15. What is the meaning of chmod +r/w/x?
16. What is the meaning of chmod -r/w/x?
17. What is the meaning of chmod 2?
18. What is the meaning of chmod 52?
19. What is the meaning of chmod 764?
20. What is the meaning of sudo chmod o+x?
21. Create a file with execute permission for all(owner, group & others)
22. Create a file with execute permission for owner
23. Create a file with write permission for group
24. Create a file with read & execute permission for other
25. Create a file with read, write & execute permission for owner
26. Create a file with read, write & execute permission for group
27. Create a file with read, write & execute permission for others

28. Create a file without read permission for group
29. Create a file without read, write & execute permission for owner
30. Create a file without read, write & execute permission for all (owner, group & others)