# 2. LINUX

## 2.1 **Introduction**

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. The defining component of Linux is the Linux kernel, an **operating system kernel first released 5 October 1991 by Linus Torvalds.**

Linux was originally developed as a free operating system for Intel x86-based personal computers. It has since been ported to more computer hardware platforms than any other operating system.

It is a leading operating system on servers and other big iron systems such as mainframe computers and supercomputers more than 90% of today's 500 fastest supercomputers run some variant of Linux, including the 10 fastest.

Linux also runs on embedded systems (devices where the operating system is typically built into the firmware and highly tailored to the system) such as mobile phones, tablet computers, network routers, televisions and video game consoles; the Android system in wide use on mobile devices is built on the Linux kernel.

A distribution oriented toward desktop use will typically include the X Window System and an accompanying desktop environment such as GNOME or KDE Plasma.

Applications commonly used with desktop Linux systems include the Mozilla Firefox web browser, the LibreOffice office application suite, and the GIMP image editor.

### 2.1.1 **Linux Distributions**

The term **distribution** refers to the assembly of a set of software around a Linux kernel to provide a ready-to-use system. The kernel of a distribution can be updated to make it possible to include recent hardware.

It involves in recompiling the kernel requires a certain level of knowledge of the system and hardware. Recompiling the kernel must be reserved for specialized users that are ready to make their system unusable for learning purposes.

Most distributions propose their own graphical installation as well as a packet management system which makes it possible to automatically install software.

The best known distributions are:

- The **RedHat** distribution;
- The **Debian** distribution;
- The **SuSe** distribution;
- The **Ubuntu** distribution;
- The **Fedora** distribution;
- The Linux **Mint** distribution.

### 2.1.2 Linux Advantages

- **Low cost**: We don't need to spend time and money to obtain licenses since Linux and much of its software come with the GNU General Public License.
- **Stability**: Linux doesn't need to be rebooted periodically to maintain performance levels.
- **Performance**: Linux provides persistent high performance on workstations and on networks.
- **Network friendliness**: Linux was developed by a group of programmers over the Internet and has therefore strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks such as network backups faster and more reliably than alternative systems.
- **Flexibility**: Linux can be used for high performance server applications, desktop applications, and embedded systems.
- **Compatibility**: It runs all common Unix software packages and can process all common file formats.
- **Choice**: The large number of Linux distributions gives you a choice. Each distribution is developed and supported by a different organization.
- **Fast and easy installation**: Most Linux distributions come with user-friendly installation and setup programs.
- **Multitasking**: Linux is designed to do many things at the same time; e.g., a large printing job in the background won't slow down your other work.
- **Security**: Linux is one of the most secure operating systems. "Walls" and flexible file access permission systems prevent access by unwanted visitors or viruses.

- **Open Source**: If you develop software that requires knowledge or modification of the operating system code, Linux's source code is at your fingertips. Most Linux applications are Open Source as well.

## 2.2 LINUX Essential Commands

Linux based Operating Systems are very powerful but their true power lies in the command line. There is a lot that you can do with the help of commands but can't otherwise (using GUI). In this article, we will find out about basic Linux commands that are used most frequently. Now, let's start from the beginning.

**Syntax**

The commands in Linux have the following syntax:

**$command options arguments**

**Basic Commands:**

**1) pwd command -** 'pwd' command prints the absolute path to current working directory.

```
$ pwd
/home/raghu
```

**2) cal command -** Displays the calendar of the current month.

```
$ cal
July 2012
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

'cal ' will display calendar for the specified month and year.

```
$ cal 08 1991
August 1991
Su Mo Tu We Th Fr Sa
1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

**3) echo command -** This command will echo whatever you provide it.

```
$ echo "linoxide.com"
linoxide.com
```

The 'echo' command is used to display the values of a variable. One such variable is 'HOME'. To check the value of a variable precede the variable with a $ sign.

```
$ echo $HOME
/home/raghu
```

**4) date command -** Displays current time and date.

```
$ date
Fri Jul 6 01:07:09 IST 2012
```

If you are interested only in time, you can use 'date +%T' (in hh:mm:ss):

```
$ date +%T
01:13:14
```

**5) tty command -** Displays current terminal.

```
$ tty
/dev/pts/0
```

**6) whoami command -** This command reveals the user who is currently logged in.

```
$ whoami
raghu
```

**7) id command -** This command prints user and groups (UID and GID) of the current user.

```
$ id
uid=1000(raghu) gid=1000(raghu)
groups=1000(raghu),4(adm),20(dialout),24(cdrom),46(plugdev),112(lp
admin),120(admin),122(sambashare)
```

**8) clear command** - This command clears the screen.

**9) help option -** With almost every command, '--help' option shows usage summary for that command.

```
$ date --help
Usage: date [OPTION]... [+FORMAT] or: date [-u|--utc|--universal]
```

[MMDDhhmm[[CC]YY][.ss]] Display the current time in the given FORMAT, or set the system date.

**10) whatis command -** This command gives a one line description about the command. It can be used as a quick reference for any command.

```
$ whatis date
date (1) - print or set the system date and time
```

**11) Manual Pages**

'--help' option and 'whatis' command do not provide thorough information about the command. For more detailed information, Linux provides man pages and info pages. To see a command's manual page, man command is used.

```
$ man date
```

**12) Info pages**

Info documents are sometimes more elaborate than the man pages. But for some commands, info pages are just the same as man pages. These are like web pages. Internal links are present within the info pages. These links are called nodes. Info pages can be navigated from one page to another through these nodes.

```
$ info date
```

## 2.3 File System Concepts

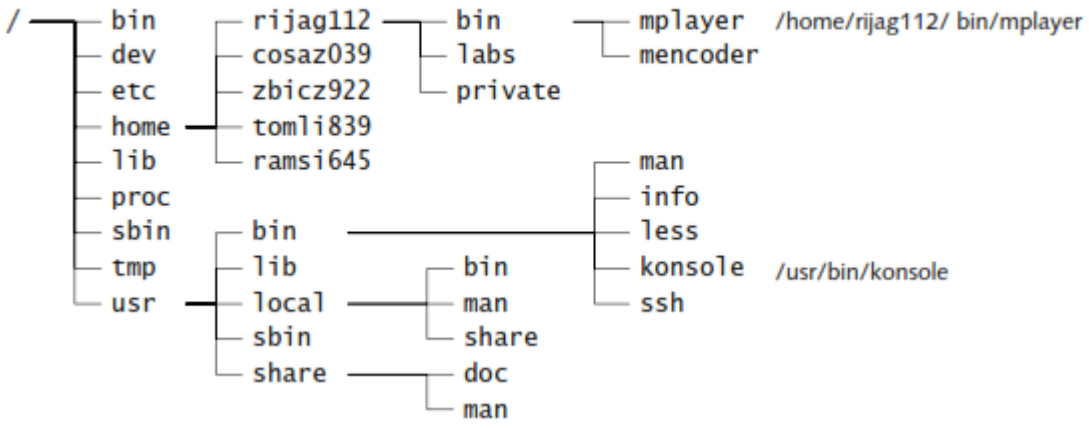
### 2.3.1 Files and directories

Understanding how files and directories are organized and can be manipulated is vital when using or managing a Linux system. All files and directories in Linux are organized in a single tree, regardless of what physical devices are involved (unlike Microsoft Windows, where individual devices typically form separate trees).

The root of the tree is called /, and is known as the root directory or simply the root. The root contains a number of directories, most of which are standard on Linux systems. The following toplevel directories are particularly important:

| Directory | Purpose |
|---|---|
| bin | Commands (binaries) needed at startup. Every Unix command is a separate executable binary file. Commands that are fundamental to operation, and may be needed while the system is starting, are stored in this directory. Other commands go in the /usr directory. |
| dev | Interfaces to hardware and logical devices. Hardware and logical devices are represented by device nodes: special files that are stored in this directory. |
| etc | Configuration files. The /etc directory holds most of the configuration of a system. In many Linux systems, /etc has a subdirectory for each installed software package. |
| home | Home directories. User's home directories are subdirectories of /home. |
| sbin | Administrative commands. The commands in /sbin typically require administrative privileges or are of interest only to system administrators. Commands that are needed when the system is starting go in /sbin. Others go in /usr/sbin. |
| tmp | Temporary (non-persistent) files. The /tmp directory is typically implemented in main |

The figure below shows part of a Unix system.




### 2.3.2 File and path names

There are two ways to reference a file in Unix: using a relative path name or using an absolute path name. An absolute path name always begins with a / and names every directory on the path from the root to the file in question. For example, in the figure above, the Console file has the absolute path name /usr/bin/console. A relative path names a path relative the current working directory.

The current working directory is set using the cd command. For example, if the current working directory is /usr, then the console file could be referenced with the name bin/console.

Note that there is no leading /. If the current working directory were /usr/share, then console could be referenced with ../bin/console. The special name ".." is used to reference the directory above the current working directory.

### 2.3.3 File system permissions

Linux, as many other modern operating systems have methods of administrating permissions or access rights to individual or groups of users. These permissions affect how users can make changes to the file system.

There are three groups of permissions on UNIX type systems:

- The **"user" group** grants permissions for the owner of a file or directory.
- The **"group" group** grants permissions for members of the file or directory's group
- The "**others" group** grants permissions for all other users.

### 2.3.4 Manipulating access rights

The chmod commands are used to manipulating permissions. chmod is used to manipulate individual permissions. Permissions can be specified using either "long" format or a numeric mode

```
% chmod -w foobar
% ls -l foobar
-r-xr-xr-x 1 john users 81 May 26 10:43 foobar
```

### 2.3.5 File System Commands

**1) Changing Directories Command**

$ cd [path-to-directory]

**2) Listing File And Directories Command**

$ ls [files-or-directories]

**3) Creating files and directories Command**

**mkdir command**

To create a directory, the 'mkdir' command is used.

$ mkdir example
$ ls -l
total 4
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 14:09

**touch command**

For creating an empty file, use the touch command.

```
$ touch file1 file2 file3
$ ls -l
total 4
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 14:09 example
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file1
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file2
-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file3
```

**4) Copy, move and remove commands**

**Copy command**

$cp source destination

Copy files and directories. If the source is a file, and the destination (file) name does not exit, then source is copied with new name i.e. with the name provided as the destination.

**Move command**

$ mv source destination

Move files or directories. The 'mv' command works like 'cp' command, except that the original file is removed. But, the mv command can be used to rename the files (or directories).

**To remove or Delete**

$ rmdir

'rmdir' command removes any empty directories, but cannot delete a directory if a file is present in it. To use ‘rmdir’ command, you must first remove all the files present in the directory you wish to remove (and possibly directories if any).

To remove files and directories

$ rm files|directories

**Other file commands**

**1) file command**

The file command determines the file type of a given file. For example:

$ file /etc/passwd
**/etc/passwd: ASCII text**

**2) stat command**

To check the status of a file. This provides more detailed information about a file than 'ls -l' output.

**3) cat command**

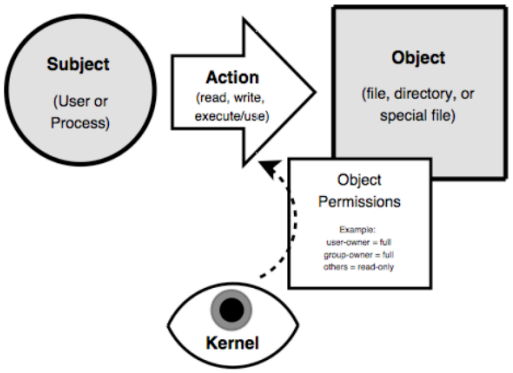
The 'cat' command is actually a concatenator but can be used to view the contents of a file.

## 2.4 **Linux Security Model**

- Traditional security model
  - people or processes with "root" privileges can do anything – other accounts can do much less
  - Goal of hackers – to gain root privilege
- Linux can be run robust and secure
  - many system admin fails to use the security features
  - add-on tools like sudo and Tripwire available
- Crux of the problem – Discretionary Access Control

**Discretionary Access Control:**

- In discretionary access control (DAC), the owner of the object specifies which subjects can access the object.
- In these operating systems, when we create a file, we decide what access privileges you want to give to other users;
- When they access your file, the operating system will make the access control decision based on the access privileges you created.




**Extended DAC:**

- Several of the first extensions to the Linux security model were to enhancements of existing Unix DAC features.
- The proprietary Unix systems evolved their own security enhancements efforts to standardize these.

**Linux Security Modules:**

- The Linux Security Modules (LSM) API implements hooks at all security-critical points within the kernel.
- A user of the framework (an "LSM") can register with the API and receive callbacks from these hooks.
- The LSM API allows different security models to be plugged into the kernel.

**Some of the Linux security modules are as follows kernel:**

- **SELinux**: Security Enhanced Linux (SELinux) is designed to meet a wide range of security requirements.
- **Smack**: provide a simple form of MAC security, in response to the relative complexity of SELinux.
- **AppArmor**: was designed to be simple to manage.
- **TOMOYO**: implements path-based security rather than object labeling.
- **Audit**: was designed to meet government certification requirements, but also actually turns out to be useful.
- **Seccomp**: Secure computing mode (seccomp) is a mechanism which restricts access to system calls by processes.
-

## 2.5 **Vi Editor**

### 2.5.1 **What is Vi Editor?**

The vi editor comes with every version of Linux or Unix. Using vi is similar to using other editors in that you can see your file on the screen.

The vi editor is the most popular editor in linux. The current version is really "vim", but to invoke it simply type "vi".

Before vi the primary editor used on Unix was the line editor - User was able to see/edit only one line of the text at a time. The vi editor is not a text formatter (like MS Word, Word Perfect, etc.) - you cannot set margins - center headings - Etc…

### 2.5.1 History of Vi Editor

Although other stories exist, the true one tells that vi was originally written by Bill Joy in 1976. Who is Bill Joy you ask? - He co-founded Sun Microsystems in 1982 and served as chief scientist until 2003.

### 2.5.2 **Characteristics of vi**

- A very powerful
- It is hard to learn, especially for windows users - Move from point to point in the file, and make changes.
- Available on all UNIX systems

### 2.5.3 **Vi modes**

There are three modes in vi

- Command mode - In this mode all the keys pressed by the user are interpret to the editor command.
- Input mode - Accessed by typing “i” - This mode permits insertion of new text, editing of existing text or replacement of existing text.
- Ex mode( Last Line Mode) -The bottom line of the vi screen is called ex mode. - When you start vi by default it is in command mode -You exit the input mode by pressing the Esc key to get back to the command mode.

### 2.5.4 **Common Vi Commands**

- w - to move one word forward

- b - to move one word backward
- $ - takes you to the end of line
- <enter> takes the cursor the beginning of next line
- - - (minus) moves the cursor to the first character in the current line
- H - takes the cursor to the beginning of the current screen Home position)
- L - moves to the Lower last line
- M - moves to the middle line on the current screen
- Control-d scrolls the screen down (half screen)
- Control-u scrolls the screen up (half screen)
- Control-f scrolls the screen forward (full screen)
- Control-b scrolls the screen backward (full screen).
- x - deletes the current character
- d - is the delete command
- yy - (yank) copy current line to buffer
- nyy - Where n is number of lines
- p - Paste the yanked lines from buffer to the line below
- P - Paste the yanked lines from buffer to the line above

2.6 **Partition Creation**

Preparing a new disk for use on a Linux system can be quick and easy. There are many tools, file system formats, and partitioning schemes that may complicate the process if you have specialized needs, but if you want to get up and running quickly, it's fairly straightforward.

This guide will cover the following process:

- Identifying the new disk on the system.
- Creating a single partition that spans the entire drive (most operating systems expect a partition layout, even if only one filesystem is present)

- Formatting the partition with the Ext4 filesystem (the default in most modern Linux distributions)

- Mounting and setting up Auto-mounting of the filesystem at boot

**Choose a Partitioning Standard**

To do this, we first need to specify the partitioning standard we wish to use. GPT is the more modern partitioning standard, while the MBR standard offers wider support among operating systems. If you do not have any special requirements, it is probably better to use GPT at this point.

To choose the **GPT** standard, pass in the disk you identified like this:

```
sudo parted /dev/sda mklabel gpt
```

If you wish to use the **MBR** format, type this instead:

```
sudo parted /dev/sda mklabel msdos
```

**Create the New Partition**

Once the format is selected, you can create a partition spanning the entire drive by typing:

```
sudo parted -a opt /dev/sda mkpart primary ext4 0% 100%
```

If we check `lsblk`, we should see the new partition available:

```
lsblk
```

**Output**

```
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0   100G  0 disk
└─sda1   8:1    0   100G  0 part
vda    253:0    0    20G  0 disk
└─vda1 253:1    0    20G  0 part /
```

**Create a Filesystem on the New Partition**

Now that we have a partition available, we can format it as an Ext4 filesystem. To do this, pass the partition to the `mkfs.ext4` utility.

We can add a partition label by passing the `-L` flag. Select a name that will help you identify this particular drive:

**Note:**

Make sure you pass in the **partition** and not the entire **disk**. In Linux, disks have names like sda, sdb, hda, etc. The partitions on these disks have a number appended to the end. So we would want to use something like sda1 and **not** sda.

sudo mkfs.ext4 -L datapartition /dev/sda1

If you want to change the partition label at a later date, you can use the `e2label` command:

sudo e2label /dev/sda1 newlabel

Some versions of `lsblk` will print all of this information if we type:

sudo lsblk --fs

If your version does not show all of the appropriate fields, you can request them manually:

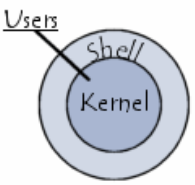
sudo lsblk -o NAME,FSTYPE,LABEL,UUID,MOUNTPOINT

**Output**

```
NAME   FSTYPE LABEL       UUID                                 MOUNTPOINT
sda
└─sda1 ext4   datapartition 4b313333-a7b5-48c1-a957-d77d637e4fda
vda
└─vda1 ext4   DOROOT       050e1e34-39e6-4072-a03e-ae0bf90ba13a /
```

## 2.7 Introduction to Shell

The command interpreter is the interface between the user and the operating system, hence the name "shell".




The shell is an executable file responsible for interpreting the commands, transmitting them to the system, and returning the result.

**Types of Shell:**

- sh ("Bourne shell")
- bash ("Bourne again shell")
- csh ("C Shell")
- Tcsh ("Tenex C shell")
- ksh ("Korn shell")
- zsh ("Zero shell").

**Basic terminology of a Shell:**

Each user has a default shell, which will be launched upon opening of a command prompt.

It is possible to change the shell during a session by simply executing the corresponding executable file.

**Prompt**

The shell is initialized by reading its overall user configuration. The prompt is displayed as follows:

machine:/directory/current$

Where,

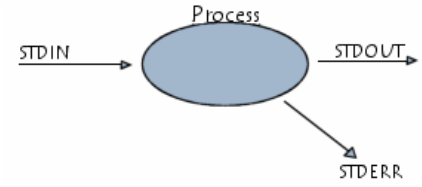
"$" speciies a normal user & "#" specifies the administrator, called "root"

**The command line concept**

A command line is a chain of characters representing a command which corresponds to an executable file of the system. Some of the basic commands are as follows,

- command --help
- command -?
- man command

**Standard input-output**




This process opens three flows:

- stdin, called standard input, in which case the process reads the input data.
- stdout, called standard output, in which case the process writes the output data.
- stderr, called standard error, in which case the process writes the error messages.

**Communication pipes**

- The pipes are a special communication mechanism of all UNIX systems.
- A pipe is symbolized by a vertical bar (character "|"), makes it possible to assign the standard output of a command.

## 2.8 String Processing

**String Length**

${#string}

expr length $string

These are the equivalent of strlen() in C.

expr "$string" : '.*'

**Length of Matching Substring at Beginning of String**

expr match "$string" '$substring' - $substring is a regular expression.

expr "$string" : '$substring' - $substring is a regular expression.

```
stringZ=abcABC123ABCabc
#       |------|
#       12345678
echo `expr match "$stringZ" 'abc[A-Z]*.2'`   # 8
echo `expr "$stringZ" : 'abc[A-Z]*.2'`       # 8
```

**Index**

expr index $string $substring

Numerical position in $string of first character in $substring that matches.

```
stringZ=abcABC123ABCabc
#       123456 ...
```

```
echo `expr index "$stringZ" C12`            # 6
                                            # C position.
echo `expr index "$stringZ" 1c`             # 3
# 'c' (in #3 position) matches before '1'.
```

This is the near equivalent of strchr() in C.

**Substring Extraction**

${string:position} - Extracts substring from $string at $position.

If the $string parameter is "*" or "@", then this extracts the positional parameters, [1] starting at $position.

${string:position:length} - Extracts $length characters of substring from $string at $position.

**Substring Removal**

${string#substring} - Deletes shortest match of $substring from front of $string.

${string##substring} - Deletes longest match of $substring from front of $string.

${string%substring} - Deletes shortest match of $substring from back of $string.

${string%%substring} - Deletes longest match of $substring from back of $string.

## 2.9 Investigation and Managing Processes

A process is an instance of a running program. It is the job of Kernel to manage CPU time and other resources among multiple processes.

Each process is given a unique PID (process ID). PID is allotted by Kernel when a process is born.

**Listing Process:**

# ps (displays the following)

| PID | TTY | TIME | CMD |
|---|---|---|---|
| 22794 | tty1 | 00:00:00 | bash |
| 22809 | tty1 | 00:00:00 | ps |

**Finding Process:**

- # ps axu | grep tty2 (shows all process, running on all terminal by all users. Use grep for filtering)
- # ps -eaf | grep sshd
- # pgrep -u username (to see PID related to specific user)
- #pidof crond (to see PID of specific process)

**Signals**:

- Signals are messages that are sent to processes with a command like kill.
- They can be sent to a process even if it is not attached to a terminal.

**Sending signals to processes:**

- Process normally terminates on their own when they have completed.
- Interactive application may need the user to issue a quit command.
- CTRL + C also terminate the process, which send an interrupt (INT) signal to a process.
- If a process doesn't terminate using above method, you can use KILL signal.

## 2.10 Network Clients

Linux supports many different networking protocols:

**TCP/IP**

- It mainly used for the purpose of interconnecting different-brand computers.

- It is one of the most robust, fast and reliable implementations and is one of the key factors of the success of Linux.

**IPX/SPX**

- IPX/SPX (Internet Packet Exchange/Sequenced Packet Exchange) is a proprietary protocol based on the Xerox Network Systems (XNS) protocol.

**AppleTalk Protocol Suite**

- It allows a peer-to-peer network model which provides basic functionality such as file and printer sharing.
- Linux provides full Appletalk networking.

**PPP, SLIP, PLIP**

- PPP is the most popular way individual users access their ISPs (Internet Service Providers).
- PLIP allows the cheap connection of two machines. It uses a parallel port and a special cable, achieving speeds of 10kBps to 20kBps.

**ATM**

- ATM support for Linux is currently in pre-alpha stage, which supports raw ATM connections (PVCs and SVCs), IP over ATM, LAN emulation...

## 2.11 Installing Application

There are several ways of installing software in Linux and they are as follows

**Online installation:**

**Step 1: Through software manager/software center (ubuntu):**

First open the terminal and run this command to get the **latest version** of the software:

**sudo apt-get update**

then

- Open **software manager/center**. It's in the **menu**.
- search your desired software in the **search box**
- If it's in the list then it will appear before you. **If it's not in the list follow the instructions in the ppa installation section of this tutorial.**
- Now double click on the desired software entry and then click "**install**".
- It will be installed on your system as per your network connection speed.

**Step 2: Through synaptic package manager:**

If it is absent in your Linux distribution then you will have to install it through software manager/center first.

- Open **synaptic package manager**. Click **reload** to get the latest version of the software.
- Search your desired software/s in the **search box**.
- Right click each software you want to install and **mark them for installing**. It will mark additional dependencies on it's own.
- After marking for installing, click **apply.**
- **I**t will download and install the marked software.

**Offline installation:**

**Step 1: Installing .deb packages**

**Through terminal:**

**cd path_to_the_directory_that_contains_the_.deb_file**

**sudo dpkg -i filename.deb**

**Through gdebi package manager:**

If gdebi is not installed then you have to install it through one of the processes #1,#2,#3 (requires internet connection)

- Then double click on the .deb file or open the file with **gdebi package manager** and click **install**.
- it will be installed soon.

**Step 2: installing .rpm packages**

rpm has to be installed in the system, otherwise follow one of the processes #1,#2,#3 to install rpm (requires internet connection)

**Code:**

**cd path_to_the_directory_that_contains_the_.rpm_file**

**sudo rpm -i filename.rpm**