Dr. D. Y. Patil Pratishthan's

Institute for Advanced Computing and Software Development

IACSD

Operating System (Linux)

Dr. D.Y. Patil Educational Complex Sector 29,Near Akurdi Railway
Station ,Nigdi Pradhikarn,Akurdi,Pune-44

INDEX

Introduction to Linux Operating System

Linux is a community of open-source Unix like operating systems that are based on the Linux Kernel. It was initially released by **Linus Torvalds** on September 17, 1991. It is a free and open-source operating system and the source code can be modified and distributed to anyone commercially or noncommercial under the GNU General Public License.
Initially, Linux was created for personal computers and gradually it was used in other machines like servers, mainframe computers, supercomputers, etc. Nowadays, Linux is also used in embedded systems like routers, automation controls, televisions, digital video recorders, video game consoles, smartwatches, etc. The biggest success of Linux is Android (operating system) it is based on the Linux kernel that is running on smartphones and tablets. Due to android Linux has the largest installed base of all general-purpose operating systems. Linux is generally packaged in a Linux distribution.

Linux Distribution

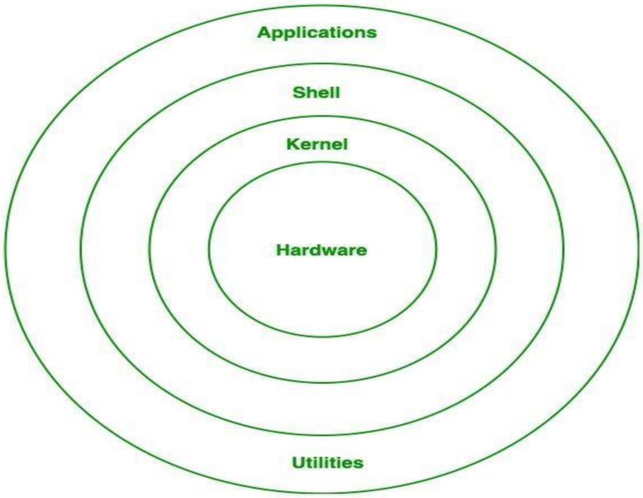
Linux distribution is an operating system that is made up of a collection of software based on Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software. And you can get Linux based operating system by downloading one of the Linux distributions and these distributions are available for different types of devices like embedded devices, personal computers, etc. Around **600 + Linux Distributions** are available and some of the popular Linux distributions are:

- MX Linux
- Manjaro
- Linux Mint
- elementary

- Ubuntu
- Debian
- Solus
- Fedora
- openSUSE
- Deepin

Architecture of Linux

Linux architecture has the following components:



1. **Kernel:** Kernel is the core of the Linux based operating system. It virtualizes the common hardware resources of the computer to provide each process with its virtual resources. This makes the process seem as if it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes.

2. **System Library: Is**the special types of functions that are used to implement the functionality of the operating system.
3. **Shell:** It is an interface to the kernel which hides the complexity of the kernel's functions from the users. It takes commands from the user and executes the kernel's functions.
4. **Hardware Layer:** This layer consists all peripheral devices like RAM/ HDD/ CPU etc.
5. **System Utility:** It provides the functionalities of an operating system to the user.

Advantages of Linux

- The main advantage of Linux, is it is an open-source operating system. This means the source code is easily available for everyone and you are allowed to contribute, modify and distribute the code to anyone without any permissions.
- In terms of security, Linux is more secure than any other operating system. It does not mean that Linux is 100 percent secure it has some malware for it but is less vulnerable than any other operating system. So, it does not require any anti-virus software.
- The software updates in Linux are easy and frequent.
- Various Linux distributions are available so that you can use them according to your requirements or according to your taste.
- Linux is freely available to use on the internet.
- It has large community support.
- It provides high stability. It rarely slows down or freezes and there is no need to reboot it after a short time.
- It maintain the privacy of the user.

- The performance of the Linux system is much higher than other operating systems. It allows a large number of people to work at the same time and it handles them efficiently.
- It is network friendly.
- The flexibility of Linux is high. There is no need to install a complete Linux suit; you are allowed to install only required components.
- Linux is compatible with a large number of file formats.
- It is fast and easy to install from the web. It can also install on any hardware even on your old computer system.
- It performs all tasks properly even if it has limited space on the hard disk.

Disadvantages of Linux

- It is not very user-friendly. So, it may be confusing for beginners.
- It has small peripheral hardware drivers as compared to windows.

Is There Any Difference between Linux and Ubuntu?

The answer is YES. The main difference between Linux and Ubuntu is Linux is the family of open-source operating systems which is based on Linux kernel, whereas Ubuntu is a free open-source operating system and the Linux distribution which is based on Debian. Or in other words, Linux is the core system and Ubuntu is the distribution of Linux. Linux is developed by Linus Torvalds and released in 1991 and Ubuntu is developed by Canonical Ltd. and released in 2004.

Linux vs Windows: What's the Difference?

It's time to make the big switch from your Windows or Mac OS operating system.

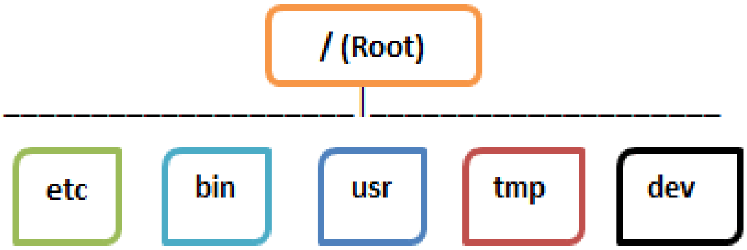
Mac OS uses a UNIX core. Your switch from Mac OS to Linux will be relatively smooth.

It's the Windows users who will need some adjusting.

Windows Vs. Linux File System

In Microsoft Windows, files are stored in folders on different data drives like C: D: E: But, in **Linux, files are ordered in a tree structure starting with the root directory**. This root directory can be considered as the start of the file system, and it further branches out various other subdirectories. The root is denoted with a forward slash '/'.

A general tree file system on your UNIX may look like this.



KEY DIFFERENCE

- Linux is an open source operating system so user can change source code as per requirement whereas Windows OS is a commercial operating system so user doesn’t have access to source code.
- Linux is very well secure as it is easy to detect bugs and fix whereas Windows has a huge user base, so it becomes a target of hackers to attack windows system.
- Linux runs faster even with older hardware whereas windows are slower compared to Linux.
- Linux peripherals like hard drives, CD-ROMs, printers are considered files whereas Windows, hard drives, CD-ROMs, printers are considered as devices

- Linux files are ordered in a tree structure starting with the root directory whereas in Windows, files are stored in folders on different data drives like C: D: E:
- In Linux you can have 2 files with the same name in the same directory while in Windows, you cannot have 2 files with the same name in the same folder.
- In Linux you would find the system and program files in different directories whereas in Windows, system and program files are usually saved in C: drive.

Types of Files

In Linux and UNIX, everything is a file. Directories are files, files are files, and devices like Printer, mouse, keyboard etc.are files.

Let's look into the File types in more detail.

General Files

General Files also called as Ordinary files. They can contain image, video, program or simply text. They can be in ASCII or a Binary format. These are the most commonly used files by Linux Users.

Directory Files

These files are a warehouse for other file types. You can have a directory file within a directory (sub-directory).You can take them as 'Folders' found in Windows operating system.

Device Files:

In MS Windows, devices like Printers, CD-ROM, and hard drives are represented as drive letters like G: H:. In Linux, there are represented as files.For example, if the first SATA hard drive had three primary partitions, they would be named and numbered as /dev/*sda1*, /dev/sda2 and /dev/sda3.

Note: All device files reside in the directory /dev/

All the above file types (including devices) have permissions, which allow a user to read, edit or execute (run) them. This is a powerful Linux/Unix feature. Access restrictions can be applied for different kinds of users, by changing permissions.

Windows Vs. Linux: Users

There are 3 types of users in Linux.

1. Regular
2. Administrative(root)
3. Service

Regular User

A regular user account is created for you when you install Ubuntu on your system. All your files and folders are stored in /home/ which is your home directory. As a regular user, you do not have access to directories of other users.

Root User

Other than your regular account another user account called root is created at the time of installation. The root account is a **superuser** who can access restricted files, install software and has administrative privileges. Whenever you want to install software, make changes to system files or perform any administrative task on Linux; you need to log in as a root user. Otherwise, for general tasks like playing music and browsing the internet, you can use your regular account.

Service user

Linux is widely used as a Server Operating System. Services such as Apache, Squid, email, etc. have their own individual service accounts. Having service accounts increases the security of your computer. Linux can allow or deny access to various resources depending on the service.

Note:

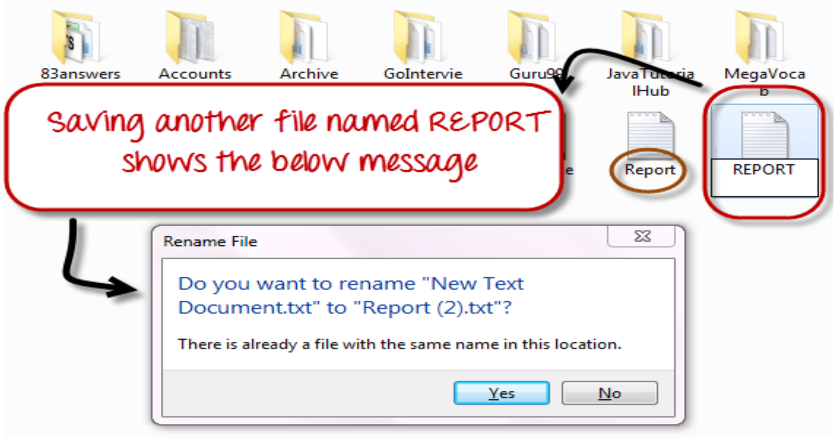
1. You will not see service accounts in Ubuntu Desktop version.
2. Regular accounts are called standard accounts in Ubuntu Desktop

In Windows, there are 4 types of user account types.

1. Administrator
2. Standard
3. Child
4. Guest

Windows Vs. Linux: File Name Convention

In Windows, you cannot have 2 files with the same name in the same folder. See below -



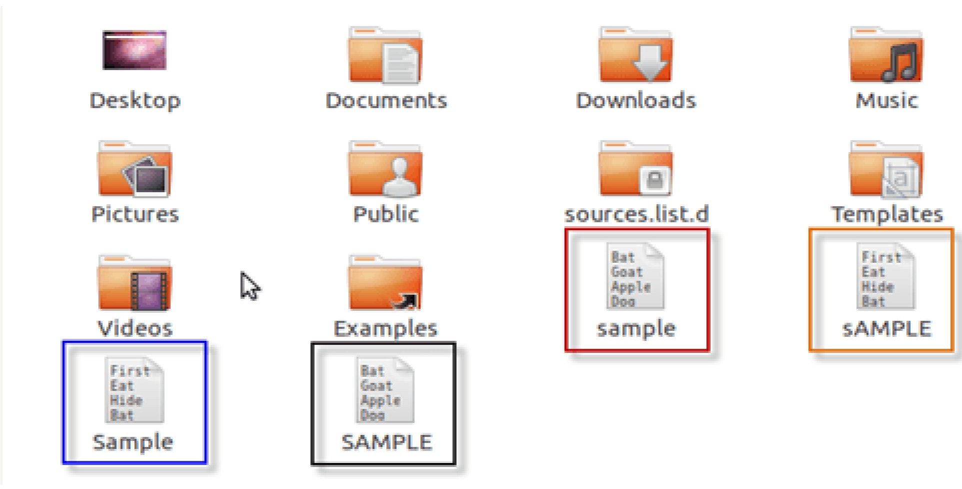
saving another file named REPORT shows the below message

Rename File

Do you want to rename "New Text Document.txt" to "Report (2).txt"?

There is already a file with the same name in this location.

While in Linux, you can have 2 files with the same name in the same directory, provided they use different cases.



Windows Vs. Linux: HOME Directory

For every user in Linux, a directory is created as **/home/**

Consider, a regular user account "Tom". He can store his personal files and directories in the directory "/home/tom". He can't save files outside his user directory and does not have access to directories of other users. For instance, he cannot access directory "/home/jerry" of another user account"Jerry".

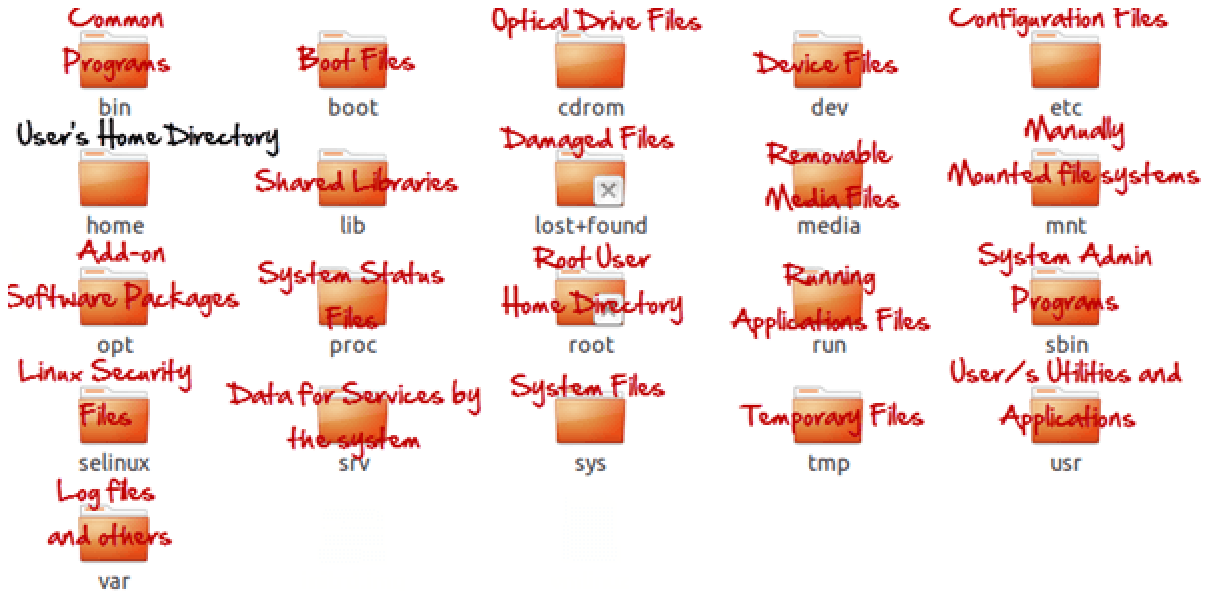
The concept is similar to C:\Documents and Settings in Windows.

When you boot the Linux operating system, your user directory (from the above example /home/tom) is the **default working directory**. Hence the directory "/home/tom is also called the **Home directory** which is a misnomer.

The working directory can be changed using some commands which we will learn later.

Windows Vs. Linux: Other Directories

In Windows, System and Program files are usually saved in C: drive. But, in Linux, you would find the system and program files in different directories. For example, the boot files are stored in the /boot directory, and program and software files can be found under /bin, device files in /dev. Below are important Linux Directories and a short description of what they contain.



These are most striking differences between Linux and other Operating Systems. There are more variations you will observe when switching to Linux and we will discuss them as we move along in our tutorials.

Windows Vs. Linux:

Windows	Linux
Windows uses different data drives like C: D: E to stored files and folders.	Unix/Linux uses a tree like a hierarchical file system.
Windows has different drives like C: D: E	There are no drives in Linux

Hard drives, CD-ROMs, printers are considered as devices	Peripherals like hard drives, CD-ROMs, printers are also considered files in Linux/Unix
There are 4 types of user account types 1) Administrator, 2) Standard, 3) Child, 4) Guest	There are 3 types of user account types 1) Regular, 2) Root and 3) Service Account
Administrator user has all administrative privileges of computers.	Root user is the super user and has all administrative privileges.
In Windows, you cannot have 2 files with the same name in the same folder	Linux file naming convention is case sensitive. Thus, sample and SAMPLE are 2 different files in Linux/Unix operating system.
In windows, My Documents is default home directory.	For every user /home/username directory is created which is called his home directory.

Linux File System

A Linux file system is a structured collection of files on a disk drive or a partition. A partition is a segment of memory and contains some specific data. In our machine, there can be various partitions of the memory. Generally, every partition contains a file system.

The general-purpose computer system needs to store data systematically so that we can easily access the files in less time. It stores the data on hard disks (HDD) or some equivalent storage type. There may be below reasons for maintaining the file system:

- Primarily the computer saves data to the RAM storage; it may lose the data if it gets turned off. However, there is non-volatile RAM (Flash RAM and SSD) that is available to maintain the data after the power interruption.
- Data storage is preferred on hard drives as compared to standard RAM as RAM costs more than disk space. The hard disks costs are dropping gradually comparatively the RAM.

The Linux file system contains the following sections:

- The root directory (/)
- A specific data storage format (EXT3, EXT4, BTRFS, XFS and so on)
- A partition or logical volume having a particular file system.

What is the Linux File System?

Linux file system is generally a built-in layer of a Linux operating system used to handle the data management of the storage. It helps to arrange the file on the disk storage. It manages the file name, file size, creation date, and much more information about a file.

If we have an unsupported file format in our file system, we can download software to deal with it.

Linux File System Structure

Linux file system has a hierarchal file structure as it contains a root directory and its subdirectories. All other directories can be accessed from the root directory. A partition usually has only one file system, but it may have more than one file system.

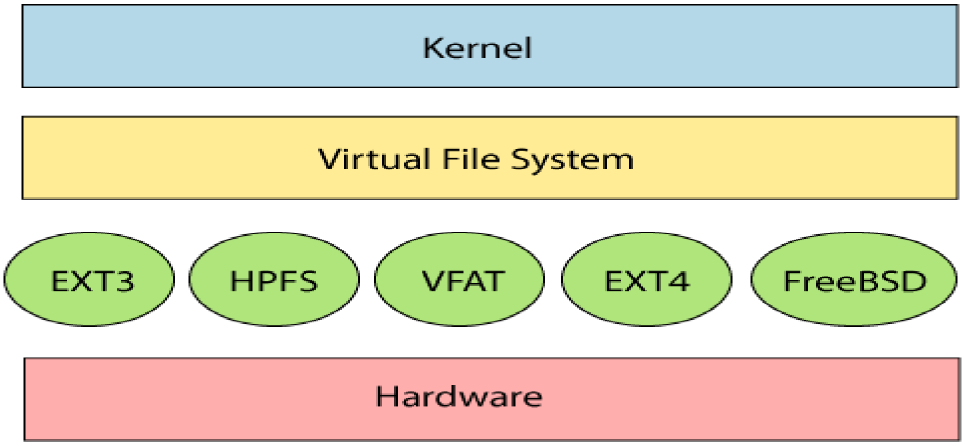
A file system is designed in a way so that it can manage and provide space for non-volatile storage data. All file systems required a namespace that is a naming and organizational methodology. The namespace defines the naming process, length of the file name, or a subset of characters that can be used for the file name. It also defines the logical structure of files on a memory segment, such as the use of directories for organizing the specific files. Once a namespace is described, a Metadata description must be defined for that particular file.

The data structure needs to support a hierarchical directory structure; this structure is used to describe the available and used disk space for a particular block. It also has the other details about the files such as file size, date & time of creation, update, and last modified.

Also, it stores advanced information about the section of the disk, such as partitions and volumes.

The advanced data and the structures that it represents contain the information about the file system stored on the drive; it is distinct and independent of the file system metadata.

Linux file system contains two-part file system software implementation architecture. Consider the below image:



The file system requires an API (Application programming interface) to access the function calls to interact with file system components like files and directories. API facilitates tasks such as creating, deleting, and copying the files. It facilitates an algorithm that defines the arrangement of files on a file system.

The first two parts of the given file system together called a **Linux virtual file system**. It provides a single set of commands for the kernel and developers to access the file system. This virtual file system requires the specific system driver to give an interface to the file system.

Linux File System Features

In Linux, the file system creates a tree structure. All the files are arranged as a tree and its branches. The topmost directory called the **root (/) directory**. All other directories in Linux can be accessed from the root directory.

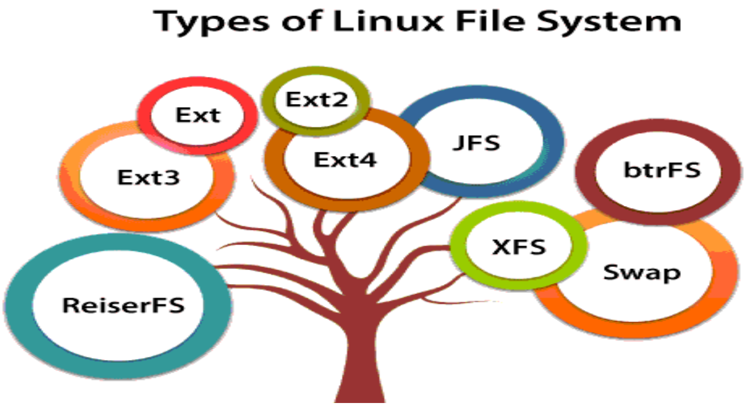
Some key features of Linux file system are as following:

- **Specifying paths:** Linux does not use the backslash (\) to separate the components; it uses forward slash (/) as an alternative. For example, as in Windows, the data may be stored in C:\ My Documents\ Work, whereas, in Linux, it would be stored in /home/ My Document/ Work.

- **Partition, Directories, and Drives:** Linux does not use drive letters to organize the drive as Windows does. In Linux, we cannot tell whether we are addressing a partition, a network device, or an "ordinary" directory and a Drive.
- **Case Sensitivity:** Linux file system is case sensitive. It distinguishes between lowercase and uppercase file names. Such as, there is a difference between test.txt and Test.txt in Linux. This rule is also applied for directories and Linux commands.
- **File Extensions:** In Linux, a file may have the extension '.txt,' but it is not necessary that a file should have a file extension. While working with Shell, it creates some problems for the beginners to differentiate between files and directories. If we use the graphical file manager, it symbolizes the files and folders.
- **Hidden files:** Linux distinguishes between standard files and hidden files, mostly the configuration files are hidden in Linux OS. Usually, we don't need to access or read the hidden files. The hidden files in Linux are represented by a dot (.) before the file name (e.g., .ignore). To access the files, we need to change the view in the file manager or need to use a specific command in the shell.

Types of Linux File System

When we install the Linux operating system, Linux offers many file systems such as **Ext, Ext2, Ext3, Ext4, JFS, ReiserFS, XFS, btrfs,** and **swap**.



Let's understand each of these file systems in detail:

1. Ext, Ext2, Ext3 and Ext4 file system

The file system Ext stands for **Extended File System**. It was primarily developed for **MINIX OS**. The Ext file system is an older version, and is no longer used due to some limitations.

Ext2 is the first Linux file system that allows managing two terabytes of data. Ext3 is developed through Ext2; it is an upgraded version of Ext2 and contains backward compatibility. The major drawback of Ext3 is that it does not support servers because this file system does not support file recovery and disk snapshot.

Ext4 file system is the faster file system among all the Ext file systems. It is a very compatible option for the SSD (solid-state drive) disks, and it is the default file system in Linux distribution.

2. JFS File System

JFS stands for **Journaled File System**, and it is developed by **IBM for AIX Unix**. It is an alternative to the Ext file system. It can also be used in place of Ext4, where stability is needed with few resources. It is a handy file system when CPU power is limited.

3. ReiserFS File System

ReiserFS is an alternative to the Ext3 file system. It has improved performance and advanced features. In the earlier time, the ReiserFS was used as the default file system in SUSE Linux, but later it has changed some policies, so SUSE returned to Ext3. This file system dynamically supports the file extension, but it has some drawbacks in performance.

4. XFS File System

XFS file system was considered as high-speed JFS, which is developed for parallel I/O processing. NASA still using this file system with its high storage server (300+ Terabyte server).

5. Btrfs File System

Btrfs stands for the **B tree file system**. It is used for fault tolerance, repair system, fun administration, extensive storage configuration, and more. It is not a good suit for the production system.

6. Swap File System

The swap file system is used for memory paging in Linux operating system during the system hibernation. A system that never goes in hibernate state is required to have swap space equal to its RAM size.

Linux Command Line Tutorial

To manage your files, you can either use

1. Terminal (Command Line Interface - CLI)
2. File manager (Graphical User Interface -GUI)

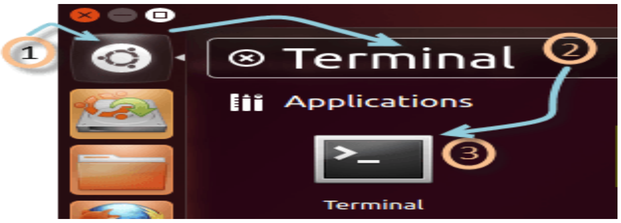
You must learn to use both GUI(File Manager) and CLI (Terminal)

GUI of a Linux based OS is similar to any other OS. Hence, we will focus on CLI and learn some useful commands.

Launching the CLI on Ubuntu

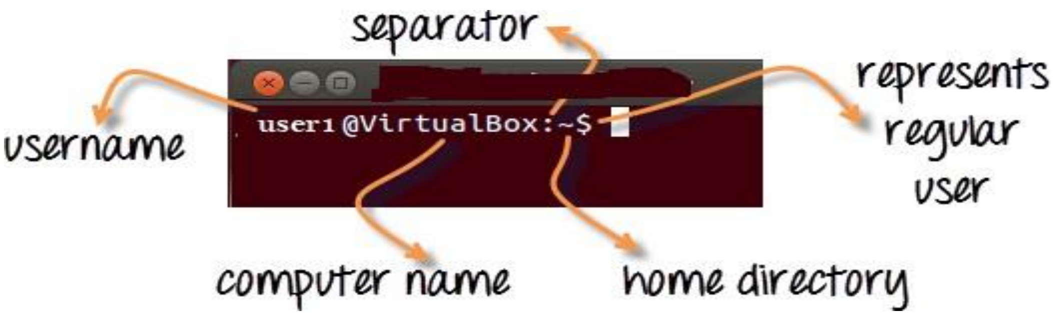
There are 2 ways to launch the terminal.

1) Go to the Dash and type terminal



2) Or you can press **CTRL + Alt + T** to launch the Terminal

Once you launch the CLI (Terminal), you would find something as



```
user1 @VirtualBox:~$
```

1) The first part of this line is the name of the **user** (bob, tom, ubuntu, home...)

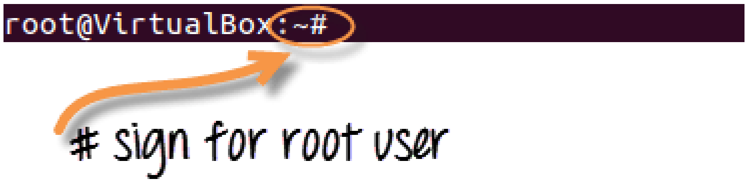
2) The second part is the computer name or the host name. The hostname helps identify a computer over the network. In a server environment, host-name becomes important.

3) The **':'** is a simple separator

4) The tilde '~' sign shows that the user in working in the **home directory**. If you change the directory, this sign will vanish.

In the above illustration, we have moved from the /home directory to /bin using the **'cd' command**. The ~ sign does not display while working in /bin directory. It appears while moving back to the home directory.

5) The '$' sign suggests that you are working as a regular user in Linux. While working as a root user, '#' is displayed.



```
root@VirtualBox:~#
```

sign for root user

Present Working Directory

The directory that you are currently browsing is called the Present working directory. You log on to the home directory when you boot your PC. If you want to determine the directory you are presently working on, use the command -

```
pwd
```

```
u1@debian:~$ pwd
/home/u1
u1@debian:~$
```

pwd command stands for **p**rint **w**orking **d**irectory

Above figure shows that /home/u1 is the directory we are currently working on.

Changing Directories

If you want to change your current directory use the '**cd**' command.

```
cd /tmp
```

Consider the following example.

```
u1@debian:~$ cd /tmp

u1@debian:/tmp$ cd /bin

u1@debian:/bin$ cd /usr

u1@debian:/usr$ cd /tmp

u1@debian:/tmp$
```

Here, we moved from directory /tmp to /bin to /usr and then back to /tmp.

Navigating to home directory

If you want to navigate to the home directory, then type **cd**.

```
cd
```

```
u1@debian:~/Documents/test$ cd
u1@debian:~$
```

You can also use the **cd ~** command.

```
cd ~
```

Moving to root directory

The root of the file system in Linux is denoted by '/'. Similar to 'c:\' in Windows.

Note: In Windows, you use backward slash "\" while in UNIX/Linux, forward slash is used "/"

Type 'cd /' to move to the root directory.

```
cd /
```

TIP: Do not forget space between **cd** and **/**. Otherwise, you will get an error.

Navigating through multiple directories

You can navigate through multiple directories at the same time by specifying its complete path.

Example: If you want to move the /cpu directory under /dev, we do not need to break this operation in two parts.

Instead, we can type '/dev/cpu' to reach the directory directly.

```
cd /dev/cpu
```

Moving up one directory level

For navigating up one directory level, try.

```
cd ..
```

Here by using the 'cd ..' command, we have moved up one directory from '/dev/cpu' to '/dev'.

Then by again using the same command, we have jumped from '/dev' to '/' root directory.

Relative and Absolute Paths

A path in computing is the address of a file or folder.

Example - **In Windows**

```
C:\documentsandsettings\user\downloads
```

In Linux

```
/home/user/downloads
```

There are two kinds of paths:

1. Absolute Path:

Let's say you have to browse the images stored in the Pictures directory of the home folder 'u1'.

The absolute file path of Pictures directory **/home/u1/Pictures**

To navigate to this directory, you can use the command.

```
cd  /home/u1/Pictures
```

This is called absolute path as you are specifying the full path to reach the file.

2. Relative Path:

The Relative path comes in handy when you have to browse another subdirectory within a given directory.

It saves you from the effort to type complete paths all the time.

Suppose you are currently in your Home directory. You want to navigate to the Downloads directory.

You do no need to type the absolute path

```
cd /home/u1/Downloads
```

Instead, you can simply type **'cd Downloads'** and you would navigate to the Downloads directory as you are already present within the **'/home/u1'** directory.

```
cd Downloads
```

This way you do not have to specify the complete path to reach a specific location within the same directory in the file system.

Command	Description
cd or cd ~	Navigate to HOME directory
cd ..	Move one level up
cd	To change to a particular directory
cd /	Move to the root directory

Linux Create File

Linux file system considers everything as a file in Linux; whether it is text file images, partitions, compiled programs, directories, or hardware devices. If it is not a file, then it must be a process. To manage the data, it forms a tree structure.

Linux files are case sensitive, so **test.txt** and **Test.txt** will be considered as two different files. There are multiple ways to create a file in Linux. Some conventional methods are as follows:

- using cat command
- using touch command
- using redirect '>' symbol
- using echo command
- using printf command
- using a different text editor like vim, nano, vi

Apart from all of the above methods, we can also create a file from the desktop file manager. Let's understand the above methods in detail:

1. Using cat command

The cat command is one of the most used commands in Linux. It is used to **create a file, display the content of the file, concatenate the contents of multiple files, display the line numbers,** and more.

Here, we will see how to create files and add content to them using cat command.

First of all, create a directory and named it as **New_directory**, execute the **mkdir** command as follows:

1. **mkdir New_directory**

Change directory to it:

```
cd  New_directory
```

Now execute the cat command to create a file:

```
cat > test.txt
```

The above command will create a text file and will enter in the editor mode. Now, enter the desired text and press **CTRL + D** key to save and exit the file and it will return to the command line.

```
u1@debian:~/New_directory$ cat > test.txt
This is a text file
created using cat command
```

To display the content of the file, execute the cat command as follows:

```
cat test.txt
```

Consider the below output:

2. Using the touch command

The **touch** command is also one of the popular commands in Linux. It is used to **create a new file, update the time stamp on existing files and directories**. It can also create empty files in Linux.

The touch command is the simplest way to create a new file from the command line. We can create multiple files by executing this command at once.

To create a file, execute the touch command followed by the file name as given below:

```
touch test1.txt
```

To list the information of the created file, execute the below command:

1. ls - l test1.txt

To create multiple files at once, specify files and their extensions after the touch command along with single space. Execute the below command to create three files at once:

1. touch test1.txt test2.txt test3.txt

To create two different types of file, execute the command as follows:

1. touch test4.txt test.odt

The above command will create two different files named as **test4.txt** and **test.odt**.

3. Using the redirect (>) symbol

We can also create a file using the redirect symbol (>) on the command line. To create a file, we just have to type a redirect symbol (>) followed by the file name. This symbol is mostly used to redirect the output. There are two ways to redirect the output. If we use > **operator**, it will overwrite the existing file, and >> operator will append the output.

To create a file with redirect (>) operator, execute the command as follows:

1. > test5.txt

The above command will create a file, to display the existence of the created file, execute the below command:

4. Using echo command

The **echo** command is used to create a file, but we should specify the file content on the command line.

To create the file with the echo command, execute the command as follows:

1. **echo " File content" > test6.txt**

The above command will create the **test6** file. To display the existence of the file, execute the below command:

5. Using printf command

We can also create a file using **printf** command. For this we need to specify the file content on the command line.

To create a file with the printf command, execute the command as follows:

printf " File content" > test7.txt

To display the file details, execute the ls command as follows:

6. Using Text Editor

We can also create a file using the different text editors like **vim, nano, vi,** and more.

- **Using Vim text editor**

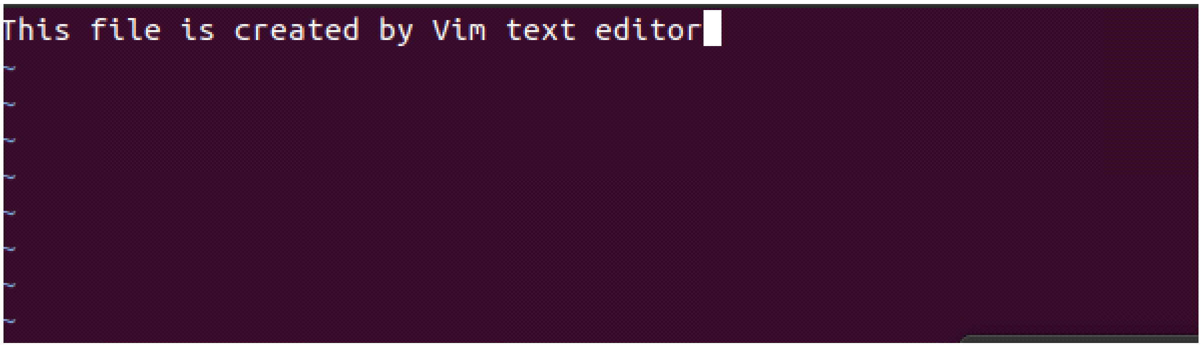
We can create a file using the **Vim text editor**. If you do not have the vim editor installed on your machine, execute the below command:

sudo apt install vim

vim test8.txt

The above command will open the text editor, press i key to go to the insert mode of the editor.

Enter the file content, press **Esc key** preceded by **:wq** to save and exit the file. The text editor looks like as follows:



```
This file is created by Vim text editor
~
~
~
~
~
~
~
~
```

To display the file information, execute the **ls** command as follows:

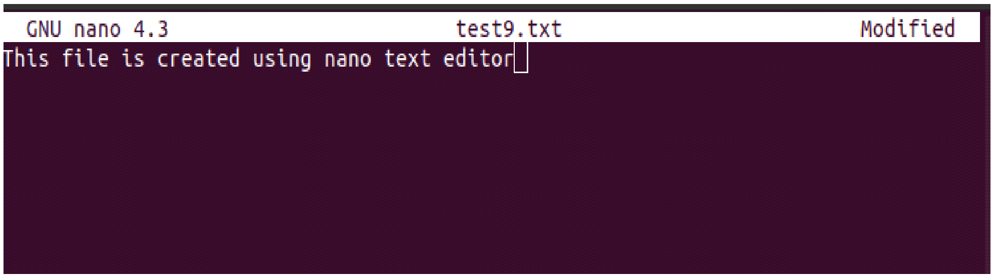
7. Using Nano editor

We can create a file using the **nano** text editor. To create a file, execute the below command:

nano test9.txt

The above command will open the nano text editor. Enter the desired text and press **CTRL + X** then type y for confirmation of the file changes. Press **Enter key** to exit from the editor.

The nano text editor looks like the below image:



```
  GNU nano 4.3                    test9.txt                    Modified
This file is created using nano text editor
```

To display the file information, execute the below command:

Using Vi editor

To create a file with Vi editor, execute the below command:

vi test10.txt

The above command will open the Vi editor. Press i key for the insert mode and enter the file content. Press Esc key and :wq to save and exit the file from the editor.

To display the file information, execute the below command

Linux file command

file command is used to determine the file type. It does not care about the extension used for file. It simply uses file command and tell us the file type. It has several options.

Syntax:

file <filename>

Example:

```
u1@debian:~$ file a.txt
a.txt: ASCII text
```

Note: File command tell us the file type with the help of a magic file that contains all the patterns to recognize a file type. Path of magic file is /usr/share/file/magic. For more information enter the command 'man 5 magic'.

Linux File Command Options

Option	Function
file -s	Used for special files.
file *	Used to list types of all the files.
file /directory name/*	Used to list types of all the files from mentioned directory.
file [range]*	It will list out all the files starting from the alphabet present within the given range.

Deleting Files

The 'rm' command removes files from the system without confirmation.

To remove a file use syntax -

```
rm filename
```

Moving and Re-naming files

To move a file, use the command.

```
mv filename new_file_location
```

Suppose we want to move the file "sample2" to location /home/u1/Documents. Executing the command

```
mv sample2 /home/u1/Documents
```

mv command needs super user permission. Currently, we are executing the command as a standard user. Hence we get the above error. To overcome the error use command.

```
sudo command_you_want_to_execute
```

Sudo program allows regular users to run programs with the security privileges of the superuser or root.

Sudo command will ask for password authentication. Though, you do not need to know the root password. You can supply your own password. After authentication, the system will invoke the requested command.

Sudo maintains a log of each command run. System administrators can trackback the person responsible for undesirable changes in the system.

```
u1@VirtualBox:~$ sudo mv sample2 /home/u1/Documents
[sudo] password for u1: ****
u1@VirtualBox:~$
```

For renaming file:

```
mv filename newfilename
```

NOTE: By default, the password you entered for sudo is retained for 15 minutes per terminal. This eliminates the need of entering the password time and again.

You only need root/sudo privileges, only if the command involves files or directories not owned by the user or group running the commands

Directory Manipulations

Enough with File manipulations! Let's learn some directory manipulation Linux basic commands.

Creating Directories

Directories can be created on a Linux operating system using the following command

```
mkdir directoryname
```

This command will create a subdirectory in your present working directory, which is usually your "Home Directory".

For example,

```
mkdir mydirectory
```

If you want to create a directory in a different location other than 'Home directory', you could use the following command -

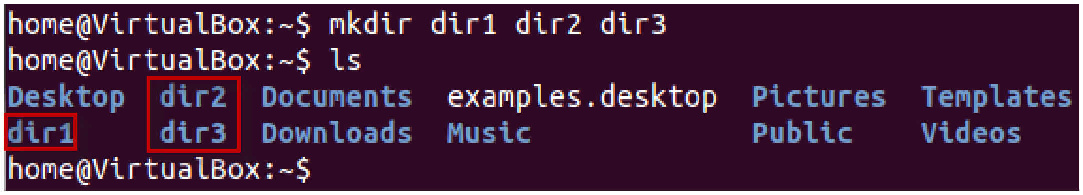
```
mkdir
```

For example:

mkdir /tmp/MUSIC

will create a directory 'Music' under '/tmp' directory

You can also create more than one directory at a time.



```
home@VirtualBox:~$ mkdir dir1 dir2 dir3
home@VirtualBox:~$ ls
Desktop  dir2  Documents  examples.desktop  Pictures  Templates
dir1     dir3  Downloads  Music             Public    Videos
home@VirtualBox:~$
```

removing Directories

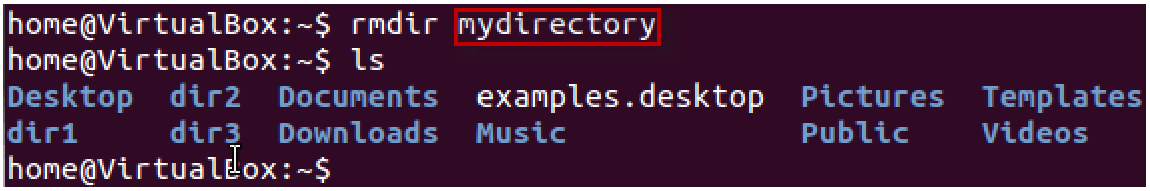
To remove a directory, use the command -

rmdir directoryname

Example

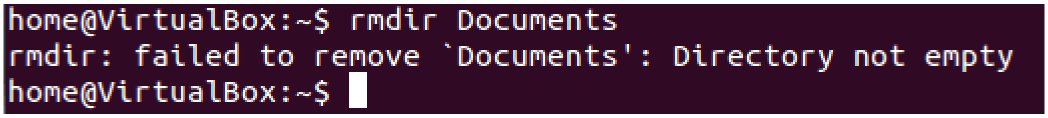
rmdir mydirectory

will delete the directory mydirectory



```
home@VirtualBox:~$ rmdir mydirectory
home@VirtualBox:~$ ls
Desktop  dir2  Documents  examples.desktop  Pictures  Templates
dir1     dir3  Downloads  Music             Public    Videos
home@VirtualBox:~$
```

Tip: Ensure that there is no file / sub-directory under the directory that you want to delete. Delete the files/sub-directory first before deleting the parent directory.



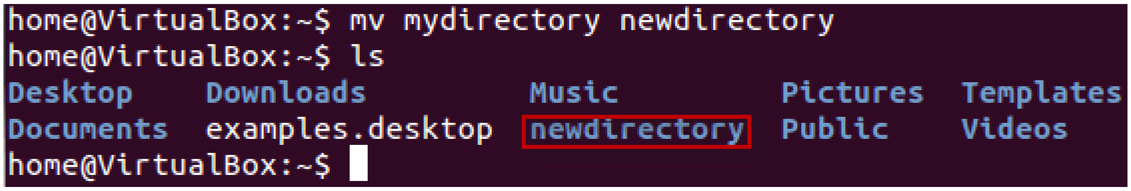
```
home@VirtualBox:~$ rmdir Documents
rmdir: failed to remove `Documents': Directory not empty
home@VirtualBox:~$
```

Renaming Directory

The 'mv' (move) command (covered earlier) can also be used for renaming directories. Use the below-given format:

mv directoryname newdirectoryname

Let us try it:



```
home@VirtualBox:~$ mv mydirectory newdirectory
home@VirtualBox:~$ ls
Desktop    Downloads        Music         Pictures  Templates
Documents  examples.desktop newdirectory  Public    Videos
home@VirtualBox:~$ 
```

How to rename a directory using Linux/Unix Commands

The 'Man' command

Man stands for manual which is a reference book of a Linux operating system. It is similar to HELP file found in popular software.

To get help on any command that you do not understand, you can type

man

The terminal would open the manual page for that command.

For an example, if we type *man man* and hit enter; terminal would give us information on man command

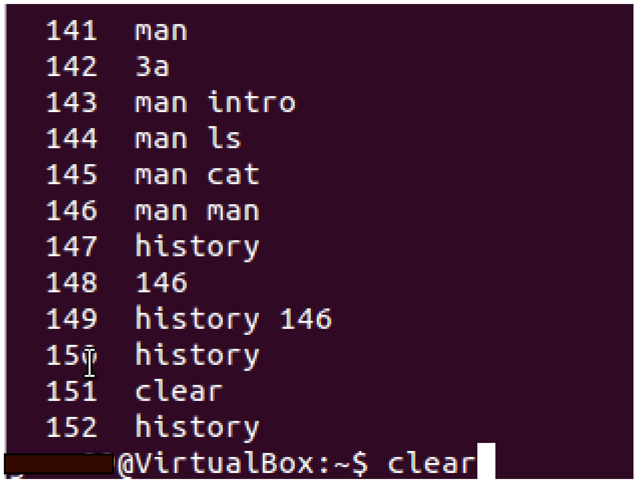
The History Command

History command shows all the basic commands in Linux that you have used in the past for the current terminal session. This can help you refer to the old commands you have entered and re-used them in your operations again.

```
u1@debian:~$ history
   1  ip a
   2  df -u
   3  mount
   4  cd /media/cdrom
   5  ls
   6  cp VMwareTools-10.3.10-12406962.tar.gz /home/u1/
   7  cd
   8  ls
   9  tar -xvf VMwareTools-10.3.10-12406962.tar.gz
  10  ls
  11  cd vmware-tools-distrib/
  12  ls
  13  ./vmware-install.pl
  14  sudo ./vmware-install.pl
  15  history
```

The clear command

This command clears all the clutter on the terminal and gives you a clean window to work on, just like when you launch the terminal.



```
  141  man
  142  3a
  143  man intro
  144  man ls
  145  man cat
  146  man man
  147  history
  148  146
  149  history 146
  150  history
  151  clear
  152  history
[redacted]@VirtualBox:~$ clear
```

The window gets cleared

Pasting commands into the terminal

Many times you would have to type in long commands on the Terminal. Well, it can be annoying at times, and if you want to avoid such a situation then copy, pasting the commands can come to rescue.

For copying, the text from a source, you would use **Ctrl + c,** but for pasting it on the Terminal, you need to use **Ctrl + Shift + p**. You can also try **Shift + Insert or select Edit>Paste on the menu**

NOTE: With Linux upgrades, these shortcuts keep changing. You can set your preferred shortcuts via Terminal> Edit> Keyboard Shortcuts.

Installing Software

.using Linux/Unix basic commands, installation files in Linux are distributed as packages. But the package contains only the program itself. Any dependent components will have to be installed separately which are usually available as packages themselves.

You can use the **apt** commands to install or remove a package. Let's update all the installed packages in our system using command -

```
sudo apt-get update
```

Linux Mail Command

For sending mails through a terminal, you will need to install packages 'mailutils'.

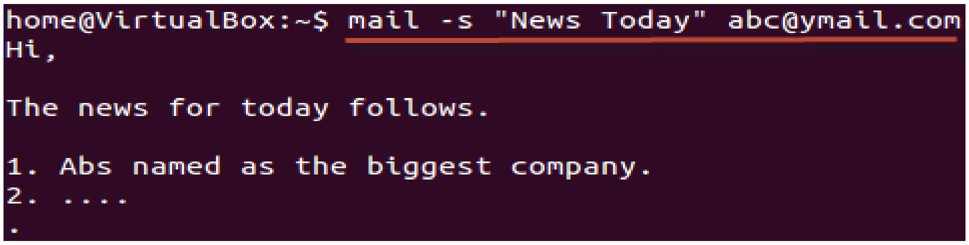
The command syntax is -

```
sudo apt-get install packagename
```

Once done, you can then use the following syntax for sending an email.

```
mail -s 'subject' -c 'cc-address' -b 'bcc-address' 'to-address'
```

This will look like:



```
home@VirtualBox:~$ mail -s "News Today" abc@ymail.com
Hi,

The news for today follows.

1. Abs named as the biggest company.
2. ....
.
```

Press Cntrl+D you are finished writing the mail. The mail will be sent to the mentioned address.

Linux Command List

Below is a Cheat Sheet of Linux commands list we have learned in this Linux commands tutorial

Command	Description
ls	Lists all files and directories in the present working directory
ls - R	Lists files in sub-directories as well
ls - a	Lists hidden files as well
ls - al	Lists files and directories with detailed information like permissions, size, owner, etc.
Lpr	Print files
sort	sorts the file content in an alphabetical order.
more	displays the first section of the file. By pressing the "ENTER" key, you can scroll line by line, all the way to the bottom of the file.
cat > filename	Creates a new file
cat filename	Displays the file content
cat file1 file2 > file3	Joins two files (file1, file2) and stores the output in a new file (file3)
touch file	Create or update file
pwd	Print Preset Working directory
head	it prints the first 10 lines of the specified files.

tail	it prints the last 10 lines of the specified files.
date	Show the current date and time
time	used to execute a command and prints a summary of real-time, user CPU time and system CPU time spent by executing a command when it terminates
uptime	Show current uptime
cal	Show this month's calender
bc	basic calculator by using which we can do basic mathematical calculations
banner	writes ASCII character Strings to standard output in large letters
echo	used to display line of text/string that are passed as an argument
w	Display who is on line
whoami	Who you are logged in as
whatis	used to get a one-line manual page descriptions.
whereis	used to find the location of source/binary file of a command and manuals sections for a specified file in Linux system.
locate	used to find the files by name
ping host	Ping host and output results
mv file "new file path"	Moves the files to the new location

mv filename new_file_name	Renames the file to a new filename
diff	allows you to compare two files line by line
sudo	Allows regular users to run programs with the security privileges of the super user or root
rm filename	Deletes a file
file	used to determine the type of a file.
man	Gives help information on a command
df	Show the disk usage
du	Show directory space usage
free	Show memory and swap usage
grep pattern file	Search for pattern in file
cp file1 file2	Copy the contents of file1 to file2
cd ..	Move up one directory. For example, if you are in /home/vic and you type “cd ..”, you will end up in /home
cd -	Return to previous directory. An easy way to get back to your previous location!
cd ~	“~” is an alias for your home directory. It can be used as a shortcut to your “home”, or other directories relative to your home.

file	Find out what kind of file it is. For example, “file /bin/ls” tells us that it is a Linux executable file.

history	Gives a list of all past basic Linux commands list typed in the current terminal session
clear	Clears the terminal
mkdir directoryname	Creates a new directory in the present working directory or a at the specified path
rmdir	Deletes a directory
mv	Renames a directory
ln	for creating links between files. By default, the ln command creates hard links
ln –s	for creating soft links between files
gzip	compresses files
zip	a compression and file packaging utility
unzip	to perform various operations on a ZIP archive file's contents.
gunzip	command-line tool for decompressing Gzip files.
tar	tar' stands for tape archive, is used to create Archive and extract the Archive

zcat	for viewing the contents of a compressed file without literally uncompressing it.
pr -x	Divides the file into x columns
pr -h	Assigns a header to the file
pr -n	Denotes the file with Line Numbers
lp -nc lpr c	Prints "c" copies of the File
lp -d lpr -P	Specifies name of the printer
find	command line utility for walking a file hierarchy
apt-get	Command used to install and update packages
mail -s 'subject' -c 'cc-address' -b 'bcc-address' 'to-address'	Command to send email

File Permissions

Linux is a clone of UNIX, the multi-user operating system which can be accessed by many users simultaneously. Linux can also be used in mainframes and servers without any modifications. But this raises security concerns as an unsolicited or malign user can corrupt, change or remove crucial data. For effective security, Linux divides authorization into 2 levels.

1. Ownership
2. Permission

The concept of Linux File **permission** and **ownership** is crucial in Linux. Here, we will explain Linux permissions and ownership and will discuss both of them. Let us start with the **Ownership.**

Ownership of Linux files

Every file and directory on your Unix/Linux system is assigned 3 types of owner, given below.

User

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.

Group

A user- group can contain multiple users. All users belonging to a group will have the same Linux group permissions access to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

Other

Any other user who has access to a file. This person has neither created the file, nor he belongs to a usergroup who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for the world.

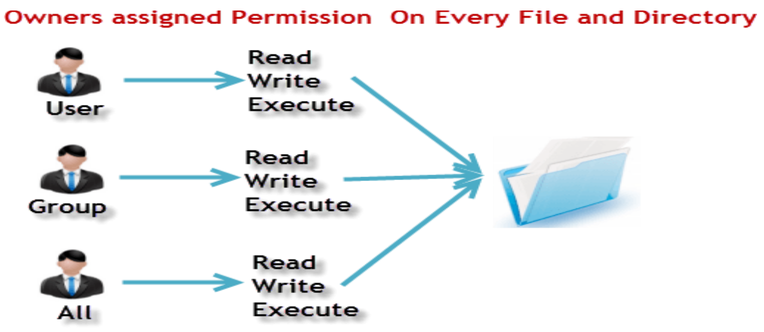
Now, the big question arises how does **Linux distinguish** between these three user types so that a user 'A' cannot affect a file which contains some other user 'B's' vital information/data. It is like you do not want your colleague, who works on your Linux computer, to view your images. This is where **Permissions** set in, and they define **user behavior**.

Let us understand the **Permission system** on Linux.

Permissions

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

- **Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
- **Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.
- **Execute:** In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.

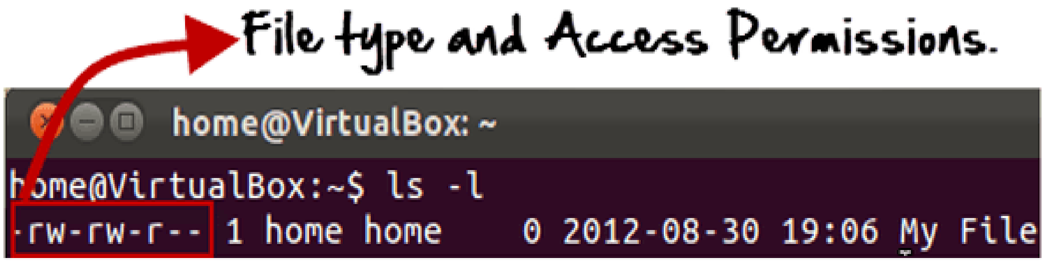


File Permissions in Linux/Unix

Let's see file permissions in Linux with examples:

ls - l on terminal gives

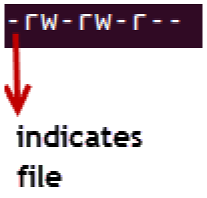
```
ls - l
```



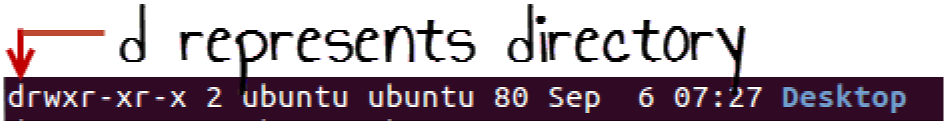
```
home@VirtualBox: ~
home@VirtualBox:~$ ls -l
-rw-rw-r-- 1 home home     0 2012-08-30 19:06 My File
```

Here, we have highlighted **'-rw-rw-r--'**and this weird looking code is the one that tells us about the Unix permissions given to the owner, user group and the world.

Here, the first '**-**' implies that we have selected a file.p>



Else, if it were a directory, **d** would have been shown.



```
drwxr-xr-x 2 ubuntu ubuntu 80 Sep  6 07:27 Desktop
```

The characters are pretty easy to remember.

r = read permission

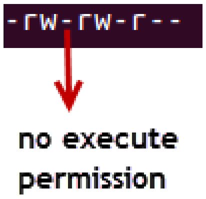
w = write permission

x = execute permission

- = no permission

Let us look at it this way.

The first part of the code is **'rw-'**. This suggests that the owner 'Home' can:



```
-rw-rw-r--
```

- Read the file
- Write or edit the file
- He cannot execute the file since the execute bit is set to '-'.

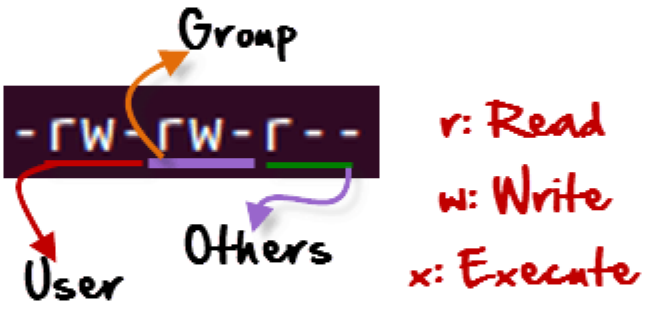
By design, many Linux distributions like Fedora, CentOS, Ubuntu, etc. will add users to a group of the same group name as the user name. Thus, a user 'tom' is added to a group named 'tom'.

The second part is **'rw-'.** It for the user group 'Home' and group-members can:

- Read the file
- Write or edit the file

The third part is for the world which means any user. It says **'r--'.** This means the user can only:

- Read the file



Changing file/directory permissions with 'chmod' command

Say you do not want your colleague to see your personal images. This can be achieved by changing file permissions.

We can use the '**chmod'** command which stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world. **Syntax:**

```
chmod permissions filename
```

There are 2 ways to use the command -

1. **Absolute mode**
2. **Symbolic mode**

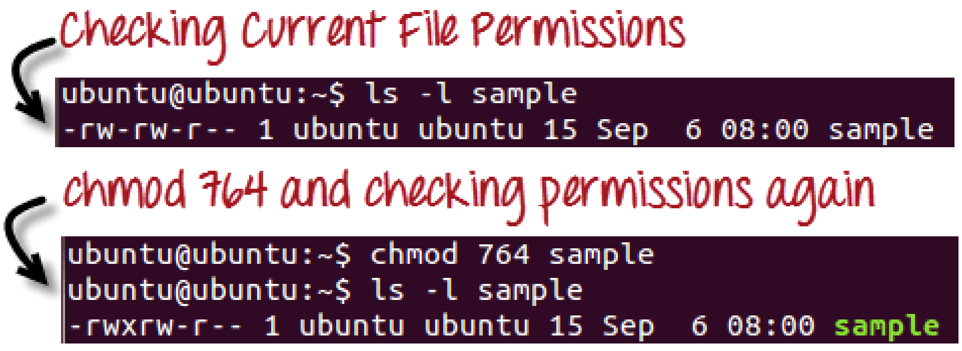
Absolute(Numeric) Mode

In this mode, file **permissions are not represented as characters but a three-digit octal number**.

The table below gives numbers for all for permissions types.

Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

Let's see the chmod permissions command in action.



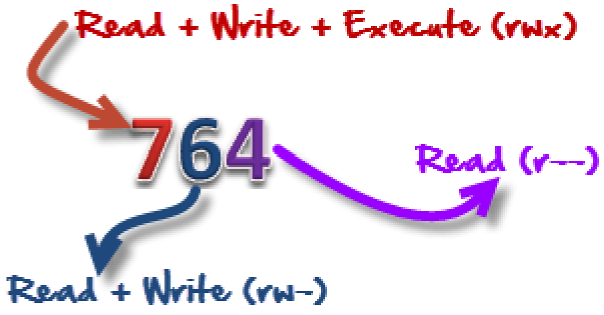
Checking Current File Permissions

```
ubuntu@ubuntu:~$ ls -l sample
-rw-rw-r-- 1 ubuntu ubuntu 15 Sep  6 08:00 sample
```

chmod 764 and checking permissions again

```
ubuntu@ubuntu:~$ chmod 764 sample
ubuntu@ubuntu:~$ ls -l sample
-rwxrw-r-- 1 ubuntu ubuntu 15 Sep  6 08:00 sample
```

In the above-given terminal window, we have changed the permissions of the file 'sample to '764'.



'764' absolute code says the following:

- Owner can read, write and execute
- Usergroup can read and write
- World can only read

This is shown as '-rwxrw-r--

This is how you can change user permissions in Linux on file by assigning an absolute number.

Symbolic Mode

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

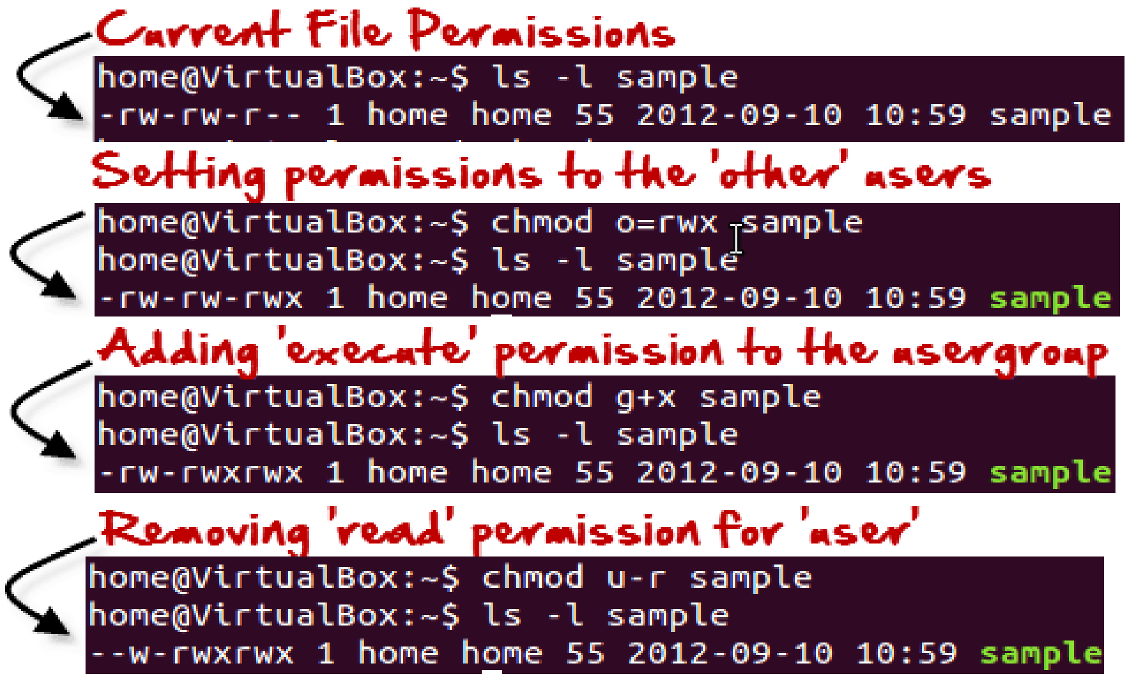
Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission

=	Sets the permission and overrides the permissions set earlier.

The various owners are represented as -

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

We will not be using permissions in numbers like 755 but characters like rwx. Let's look into an example



Current File Permissions

```
home@VirtualBox:~$ ls -l sample
-rw-rw-r-- 1 home home 55 2012-09-10 10:59 sample
```

Setting permissions to the 'other' users

```
home@VirtualBox:~$ chmod o=rwx sample
home@VirtualBox:~$ ls -l sample
-rw-rw-rwx 1 home home 55 2012-09-10 10:59 sample
```

Adding 'execute' permission to the usergroup

```
home@VirtualBox:~$ chmod g+x sample
home@VirtualBox:~$ ls -l sample
-rw-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

Removing 'read' permission for 'user'

```
home@VirtualBox:~$ chmod u-r sample
home@VirtualBox:~$ ls -l sample
--w-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

Changing Ownership and Group

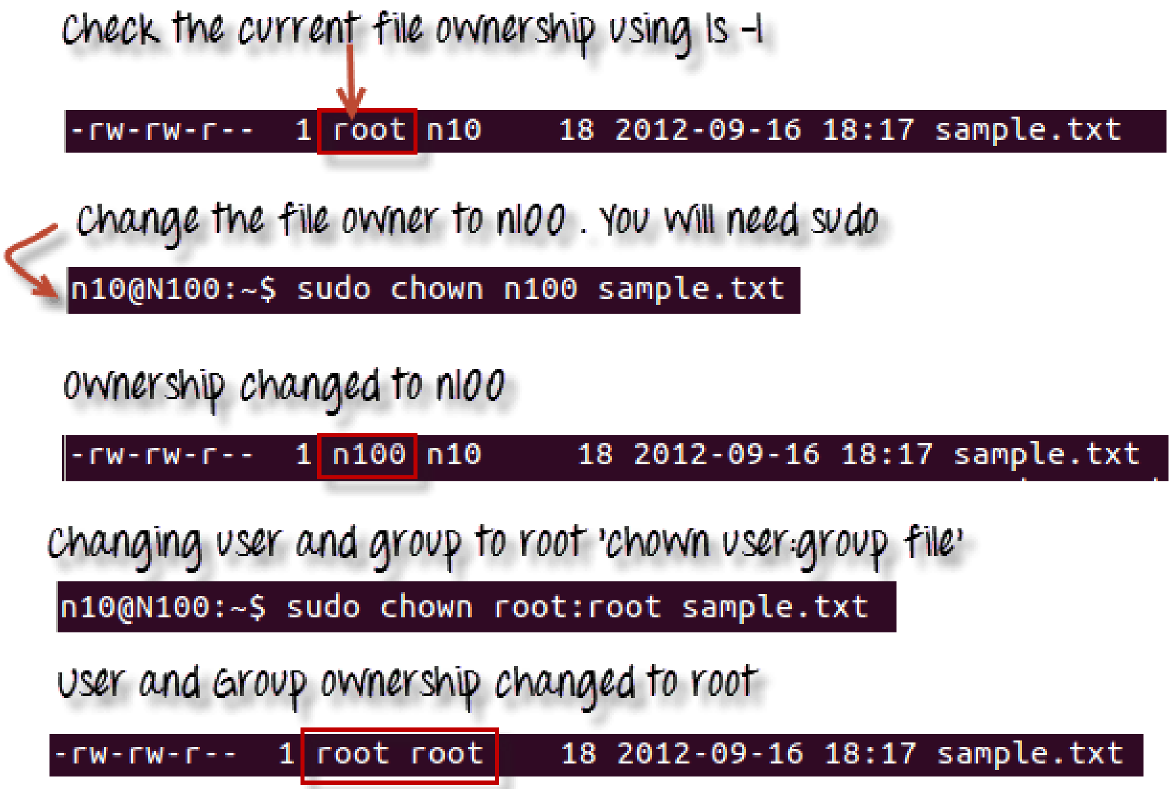
For changing the ownership of a file/directory, you can use the following command:

chown user

In case you want to change the user as well as group for a file or directory use the command

chown user:group filename

Let's see this in action



Check the current file ownership using ls -l

```
-rw-rw-r--  1 root n10    18 2012-09-16 18:17 sample.txt
```

Change the file owner to n100 . You will need sudo

```
n10@N100:~$ sudo chown n100 sample.txt
```

ownership changed to n100

```
-rw-rw-r--  1 n100 n10     18 2012-09-16 18:17 sample.txt
```

Changing user and group to root 'chown user:group file'

```
n10@N100:~$ sudo chown root:root sample.txt
```

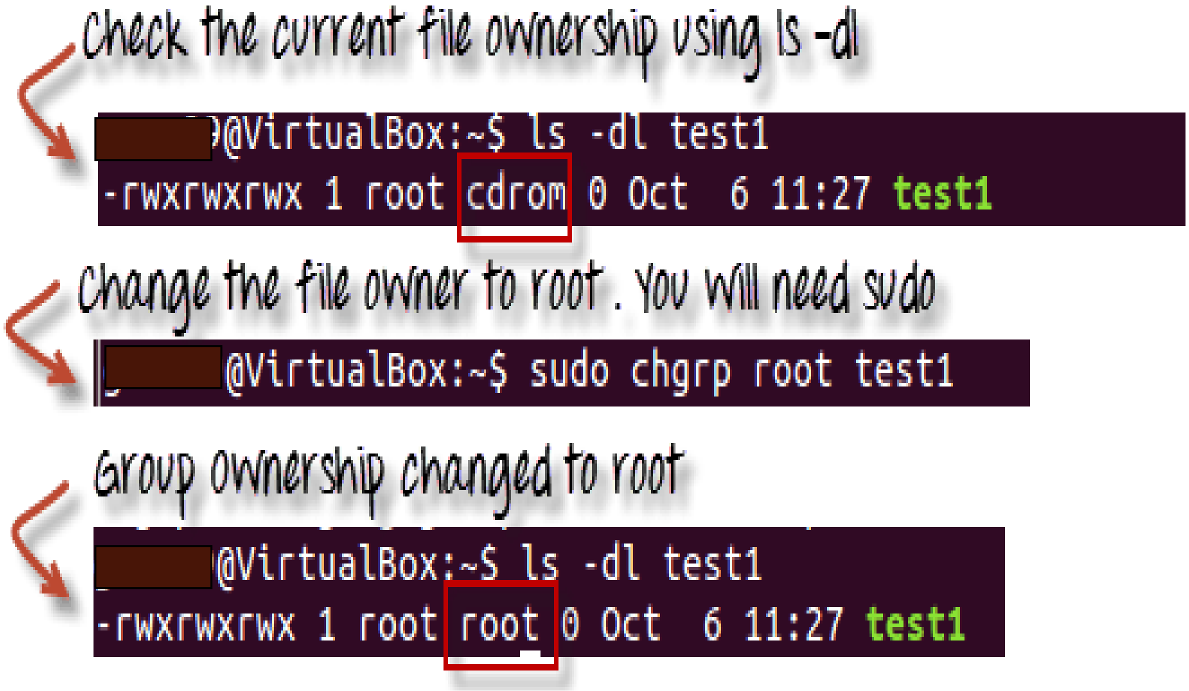
User and Group ownership changed to root

```
-rw-rw-r--  1 root root    18 2012-09-16 18:17 sample.txt
```

In case you want to change group-owner only, use the command

chgrp group_name filename

'**chgrp'** stands for change group.



Check the current file ownership using ls -dl

```
[redacted]@VirtualBox:~$ ls -dl test1
-rwxrwxrwx 1 root cdrom 0 Oct  6 11:27 test1
```

Change the file owner to root. You will need sudo

```
[redacted]@VirtualBox:~$ sudo chgrp root test1
```

Group ownership changed to root

```
[redacted]@VirtualBox:~$ ls -dl test1
-rwxrwxrwx 1 root root 0 Oct  6 11:27 test1
```

Tip

- The file /etc/group contains all the groups defined in the system
- You can use the command "groups" to find all the groups you are a member of

```
u1@debian:~$ groups
u1 cdrom floppy audio dip video plugdev netdev bluetooth lpadmin
scanner
u1@debian:~$
```

- You can use the command newgrp to work as a member a group other than your default group

```
u1@debian:~$ newgrp cdrom
u1@debian:~$
u1@debian:~$ touch c.txt
u1@debian:~$ ls -ld c.txt
-rw-r--r-- 1 u1 cdrom 0 Mar  5 01:25 c.txt
```

- You cannot have 2 groups owning the same file.
- You do not have nested groups in Linux. One group cannot be sub-group of other
- x- eXecuting a directory means Being allowed to "enter" a dir and gain possible access to sub-

The ftp Utility

ftp stands for File Transfer Protocol. This utility helps you upload and download your file from one computer to another computer.

Syntax

```
$ftp hostname or ip-address
```

The above command would prompt you for the login ID and the password. Once you are authenticated, you can access the home directory of the login account and you would be able to perform various commands.

The following tables lists out a few important commands −

Sr.No.	Command & Description
1	put filename Uploads filename from the local machine to the remote machine.
2	get filename Downloads filename from the remote machine to the local machine.
3	mput file list Uploads more than one file from the local machine to the remote machine.
4	mget file list Downloads more than one file from the remote machine to the local machine.
5	prompt off Turns the prompt off. By default, you will receive a prompt to upload or download files using mput or mget commands.
6	prompt on

	Turns the prompt on.
7	dir Lists all the files available in the current directory of the remote machine.
8	cd dirname Changes directory to dirname on the remote machine.
9	lcd dirname Changes directory to dirname on the local machine.
10	quit Helps logout from the current login.

The telnet Utility

There are times when we are required to connect to a remote Unix machine and work on that machine remotely. **Telnet** is a utility that allows a computer user at one site to make a connection, login and then conduct work on a computer at another site.

Once you login using Telnet, you can perform all the activities on your remotely connected machine.

The finger Utility

The **finger** command displays information about users on a given host. The host can be either local or remote.

Finger may be disabled on other systems for security reasons.

Input Output Redirection in Linux

What is Redirection?

Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices. The basic workflow of any Linux command is that it takes an input and give an output.

- The standard input (stdin) device is the keyboard.
- The standard output (stdout) device is the screen.

With redirection, the above standard input/output can be changed.

Output Redirection

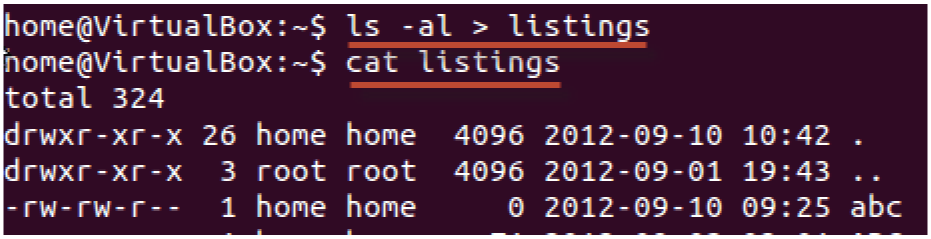
The **'>'** symbol is used for output (STDOUT) redirection.

> Output Redirection

Example:

ls -al > listings

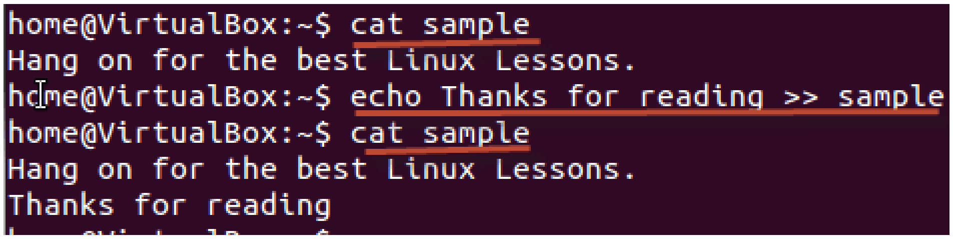
Here the output of command ls -al is re-directed to file "listings" instead of your screen.



```
home@VirtualBox:~$ ls -al > listings
home@VirtualBox:~$ cat listings
total 324
drwxr-xr-x 26 home home   4096 2012-09-10 10:42 .
drwxr-xr-x  3 root root   4096 2012-09-01 19:43 ..
-rw-rw-r--  1 home home      0 2012-09-10 09:25 abc
```

Note: Use the correct file name while redirecting command output to a file. If there is an existing file with the same name, the redirected command will delete the contents of that file and then it may be overwritten."

If you do not want a file to be overwritten but want to add more content to an existing file, then you should use **'>>'** operator.



```
home@VirtualBox:~$ cat sample
Hang on for the best Linux Lessons.
home@VirtualBox:~$ echo Thanks for reading >> sample
home@VirtualBox:~$ cat sample
Hang on for the best Linux Lessons.
Thanks for reading
```

You can redirect standard output, to not just files, but also devices!

$ cat music.mp3 > /dev/audio

The cat command reads the file music.mp3 and sends the output to /dev/audio which is the audio device. If the sound configurations in your PC are correct, this command will play the file music.mp3

Input redirection

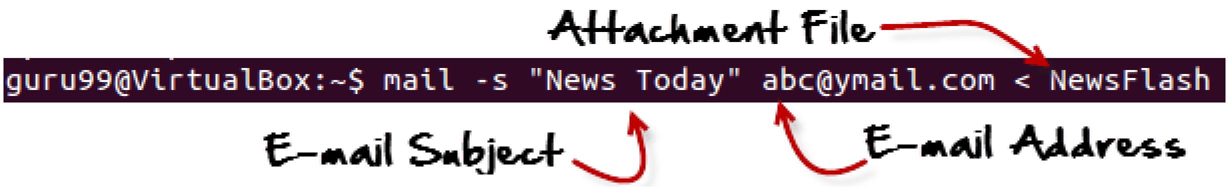
The **'<'** symbol is used for input(STDIN) redirection

< Input Redirection

Example: The mail program in Linux can help you send emails from the Terminal.

You can type the contents of the email using the standard device keyboard. But if you want to attach a File to email you can use the input re-direction operator in the following format.

Mail -s "Subject" to-address < Filename



```
guru99@VirtualBox:~$ mail -s "News Today" abc@ymail.com < NewsFlash
```

This would attach the file with the email, and it would be sent to the recipient.

The above examples were simple. Let's look at some advance re-direction techniques which make use of File Descriptors.

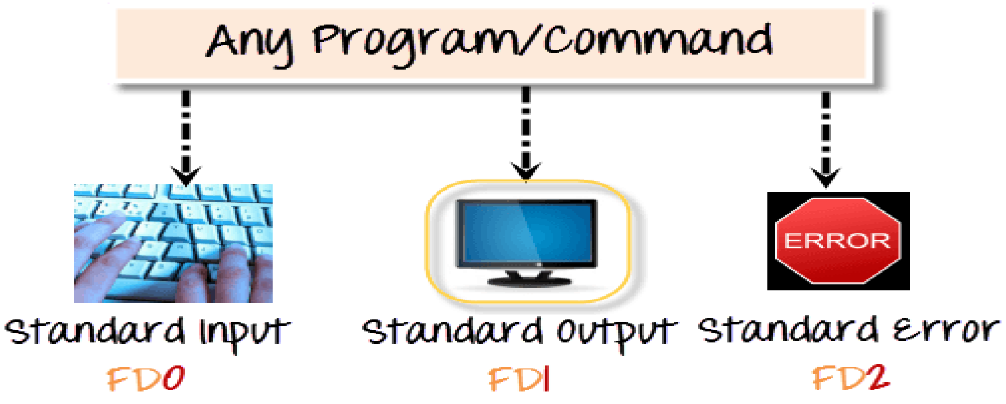
File Descriptors (FD)

In Linux/Unix, everything is a file. Regular file, Directories, and even Devices are files. Every File has an associated number called File Descriptor (FD).

Your screen also has a File Descriptor. When a program is executed the output is sent to File Descriptor of the screen, and you see program output on your monitor. If the output is sent to File Descriptor of the printer, the program output would have been printed.

Error Redirection

Whenever you execute a program/command at the terminal, 3 files are always open, viz., standard input, standard output, standard error**.**



These files are always present whenever a program is run. As explained before a file descriptor, is associated with each of these files.

File	**File Descriptor**
Standard Input STDIN	**0**
Standard Output STDOUT	**1**
Standard Error STDERR	**2**

By default, error stream is displayed on the screen. Error redirection is routing the errors to a file other than the screen.

Why Error Redirection?

Error re-direction is one of the very popular features of Unix/Linux.

Frequent UNIX users will reckon that many commands give you massive amounts of errors.

- For instance, while searching for files, one typically gets permission denied errors. These errors usually do not help the person searching for a particular file.
- While executing shell scripts, you often do NOT want error messages cluttering up the normal program output.

The solution is to re-direct the error messages to a file.

Example 1

```
$ myprogram 2>errorsfile
```

Above we are executing a program names myprogram.

The file descriptor for standard error is 2.

Using "2>" we re-direct the error output to a file named "errorfile"

Thus, program output is not cluttered with errors.

Example 2

Here is another example which uses find statement -

```
find . -name 'my*' 2>error.log
```

Using the "find" command, we are searching the "." current directory for a file with "name" starting with "my"

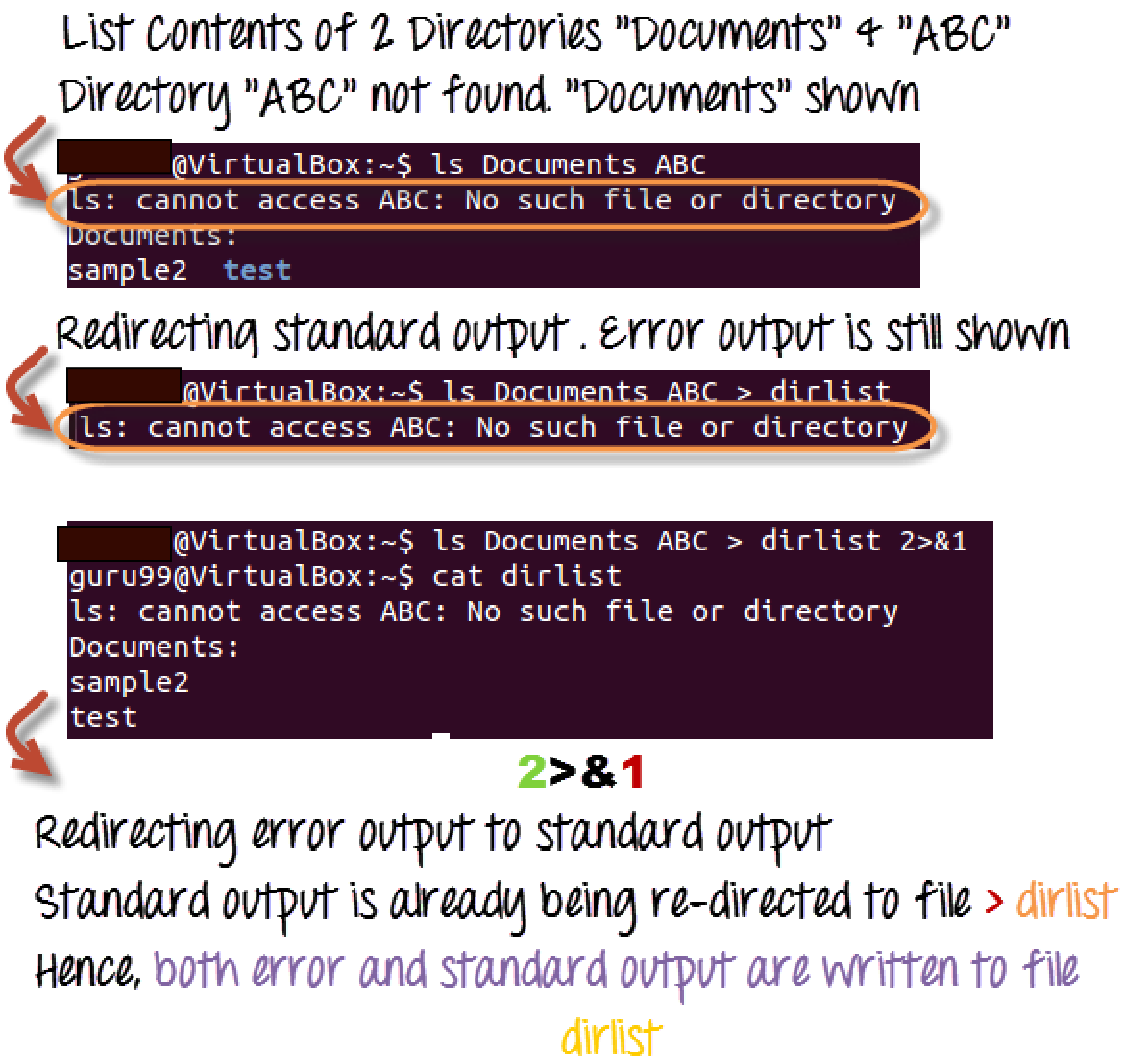
Example 3 Let's see a more complex example,

Server Administrators frequently, list directories and store both error and standard output into a file, which can be processed later. Here is the command.

```
ls Documents ABC> dirlist 2>&1
```

Here,

- which writes the output from one file to the input of another file. 2>&1 means that STDERR redirects to the target of STDOUT (which is the file dirlist)
- We are redirecting error output to standard output which in turn is being re-directed to file dirlist. Hence, both the output is written to file dirlist



List Contents of 2 Directories "Documents" + "ABC"

Directory "ABC" not found. "Documents" shown

```
[redacted]@VirtualBox:~$ ls Documents ABC
ls: cannot access ABC: No such file or directory
Documents:
sample2  test
```

Redirecting standard output . Error output is still shown

```
[redacted]@VirtualBox:~$ ls Documents ABC > dirlist
ls: cannot access ABC: No such file or directory
```

```
[redacted]@VirtualBox:~$ ls Documents ABC > dirlist 2>&1
guru99@VirtualBox:~$ cat dirlist
ls: cannot access ABC: No such file or directory
Documents:
sample2
test
```

2>&1

Redirecting error output to standard output

Standard output is already being re-directed to file > dirlist

Hence, both error and standard output are written to file dirlist

Pipe, Grep and Sort Command

What is a Pipe in Linux?

The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next. In short, the output of each process directly as input to the next one like a pipeline. The symbol '|' denotes a pipe.

Pipes help you mash-up two or more commands at the same time and run them consecutively. You can use powerful commands which can perform complex tasks in a jiffy.

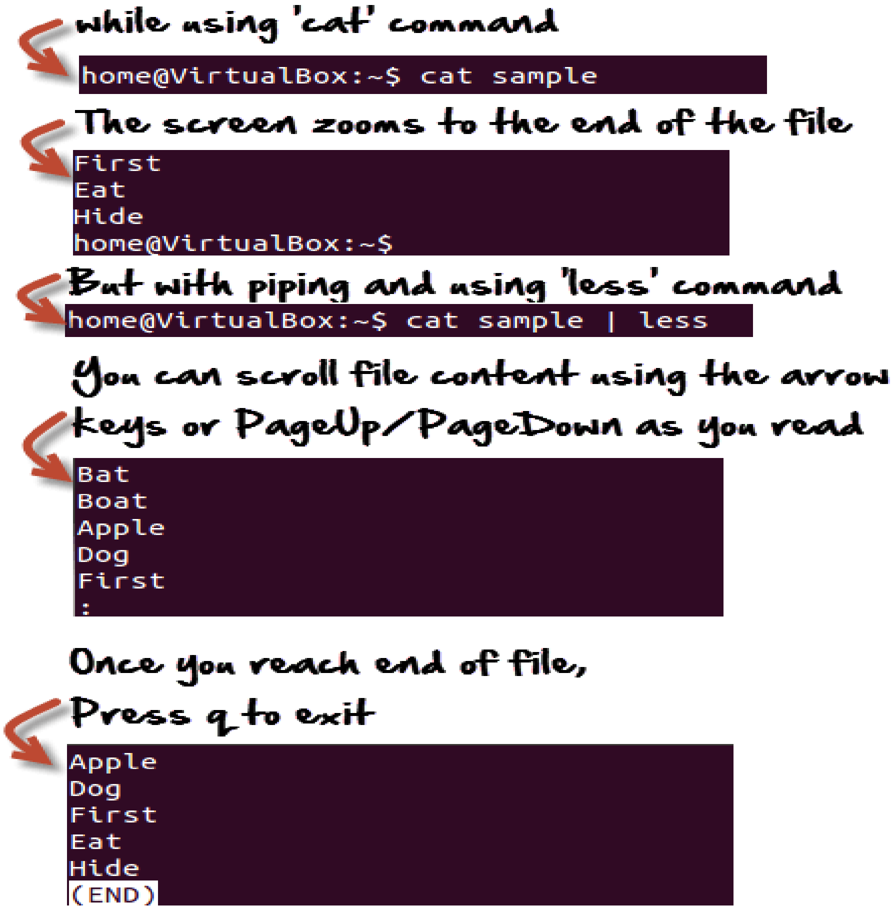
Let us understand this with an example.

When you use 'cat' command to view a file which spans multiple pages, the prompt quickly jumps to the last page of the file, and you do not see the content in the middle.

To avoid this, you can pipe the output of the 'cat' command to 'less' which will show you only one scroll length of content at a time.

```
cat  filename | less
```

An illustration would make it clear.



while using 'cat' command

```
home@VirtualBox:~$ cat sample
```

The screen zooms to the end of the file

```
First
Eat
Hide
home@VirtualBox:~$
```

But with piping and using 'less' command

```
home@VirtualBox:~$ cat sample | less
```

You can scroll file content using the arrow keys or PageUp/PageDown as you read

```
Bat
Boat
Apple
Dog
First
:
```

Once you reach end of file, Press q to exit

```
Apple
Dog
First
Eat
Hide
(END)
```

'pg' and 'more' commands

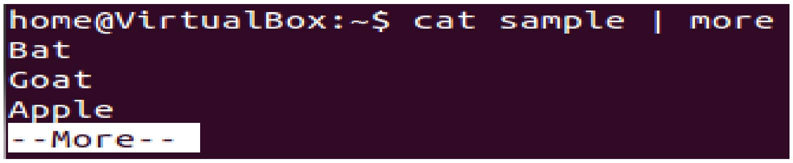
Instead of 'less', you can also use.

```
cat Filename | pg
```

or

```
cat Filename | more
```

And, you can view the file in digestible bits and scroll down by simply hitting the enter key.



```
home@VirtualBox:~$ cat sample | more
Bat
Goat
Apple
--More--
```

The 'grep' command

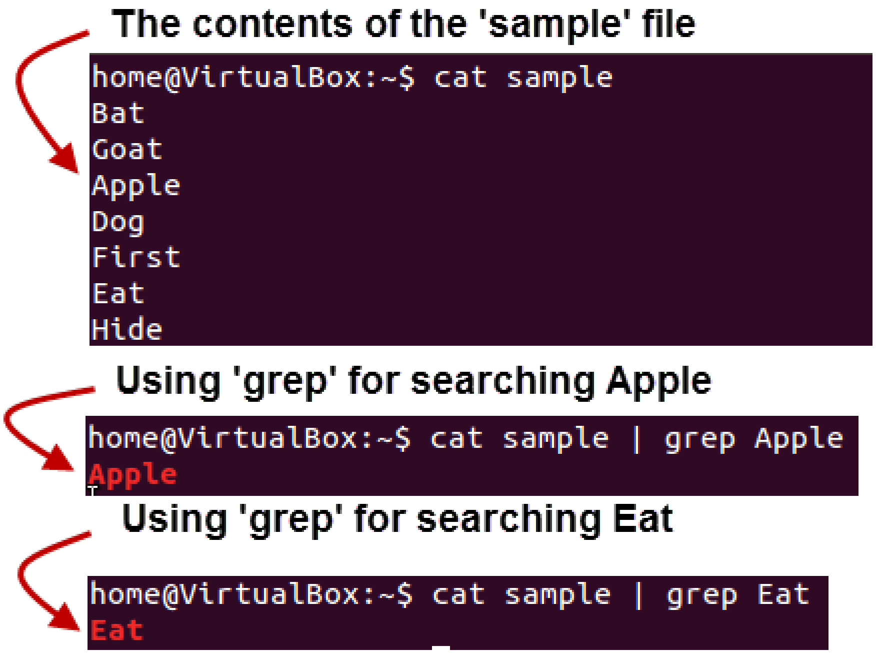
Suppose you want to search a particular information the postal code from a text file.

You may manually skim the content yourself to trace the information. A better option is to use the grep command. It will scan the document for the desired information and present the result in a format you want.

Syntax:

```
grep search_string
```

Let's see it in action -



The contents of the 'sample' file

```
home@VirtualBox:~$ cat sample
Bat
Goat
Apple
Dog
First
Eat
Hide
```

Using 'grep' for searching Apple

```
home@VirtualBox:~$ cat sample | grep Apple
Apple
```

Using 'grep' for searching Eat

```
home@VirtualBox:~$ cat sample | grep Eat
Eat
```

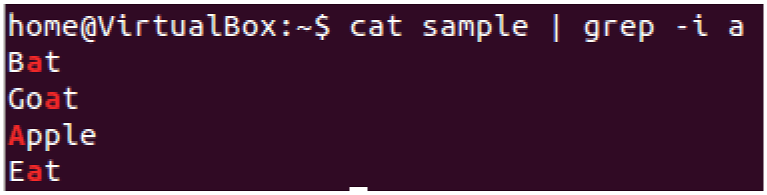
Here, **grep** command has searched the file 'sample', for the string 'Apple' and 'Eat'.

Following options can be used with this command.

Option	**Function**
-v	Shows all the lines that do not match the searched string
-c	Displays only the count of matching lines
-n	Shows the matching line and its number
-i	Match both (upper and lower) case
-l	Shows just the name of the file with the string

Let us try the first option **'-i'** on the same file use above -

Using the 'i' option grep has filtered the string 'a' (case-insensitive) from the all the lines.



```
home@VirtualBox:~$ cat sample | grep -i a
Bat
Goat
Apple
Eat
```

The 'sort' command

This command helps in **sorting out the contents of a file alphabetically.**

The syntax for this command is:

sort Filename

```
u1@debian:~$ cat b
a
c
b
e
d

u1@debian:~$ sort b
a
b
c
d
e
```

There are **extensions** to this command as well, and they are listed below.

Option	Function
-r	Reverses sorting

-n	Sorts numerically
-f	Case insensitive sorting

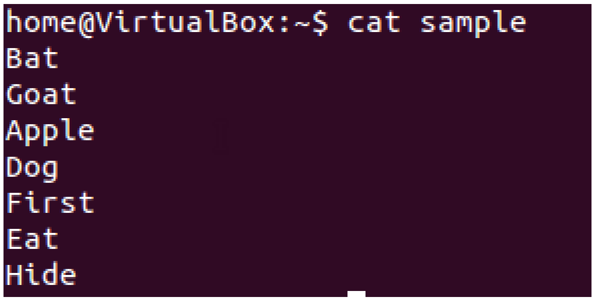
What is a Filter?

Linux has a lot of filter commands like awk, grep, sed, spell, and wc. A filter takes input from one command, does some processing, and gives output.

When you pipe two commands, the "filtered " output of the first command is given to the next.

Let's understand this with the help of an example.

We have the following file 'sample'



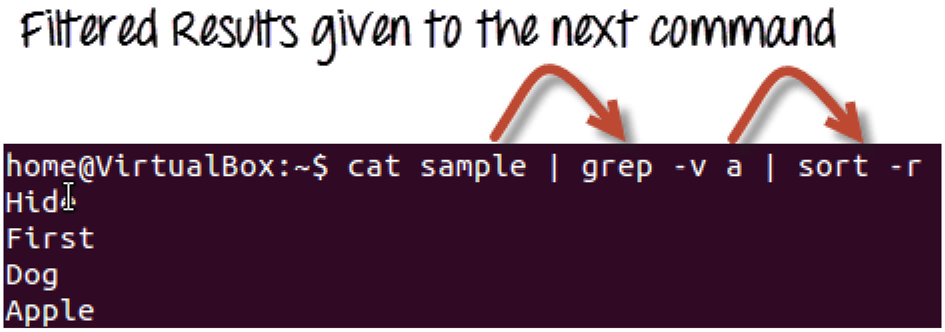
```
home@VirtualBox:~$ cat sample
Bat
Goat
Apple
Dog
First
Eat
Hide
```

We want to highlight only the lines that do not contain the character 'a', but the result should be in reverse order.

For this, the following syntax can be used.

```
cat sample | grep -v a | sort - r
```

Let us look at the result.



Filtered Results given to the next command

```
home@VirtualBox:~$ cat sample | grep -v a | sort -r
Hide
First
Dog
Apple
```

Linux Regular Expression

What are Linux Regular Expressions?

Linux Regular Expressions are special characters which help search data and matching complex patterns. Regular expressions are shortened as 'regexp' or 'regex'. They are used in many Linux programs like grep, bash, rename, sed, etc.

Types of Regular expressions

For ease of understanding let us learn the different types of Regex one by one.

- Basic Regular expressions
- Interval Regular expressions
- Extended regular expressions

Basic Regular expressions

Some of the commonly used commands with Regular expressions are tr, sed, vi and grep. Listed below are some of the basic Regex.

Symbol	Descriptions
.	replaces any character
^	matches start of string

$	matches end of string
*	matches up zero or more times the preceding character
\	Represent special characters
()	Groups regular expressions
?	Matches up exactly one character

Let's see an example.

Execute cat sample to see contents of an existing file

```
u1@debian:~$ cat b

apple
bat
on
off
eat
```

Search for content containing letter 'a'.

```
u1@debian:~$ cat b | grep a

apple
bat
eat
```

'**^**' matches the start of a string. Let's search for content that STARTS with a

```
u1@debian:~$ cat b | grep ^a


apple
```

Only lines that start with character are filtered. Lines which do not contain the character 'a' at the start are ignored.

Select only those lines that end with t using **$**

```
u1@debian:~$ cat b | grep t$
bat
eat
```

Interval Regular expressions

These expressions tell us about the number of occurrences of a character in a string. They are

Expression	Description
{n}	Matches the preceding character appearing 'n' times exactly
{n,m}	Matches the preceding character appearing 'n' times but not more than m
{n, }	Matches the preceding character only when it appears 'n' times or more

We want to check that the character 'p' appears exactly 2 times in a string one after the other. For this the syntax would be:

```
cat sample | grep -E p\{2}
```

Note: You need to add -E with these regular expressions.

Extended regular expressions

These regular expressions contain combinations of more than one expression. Some of them are:

Expression	Description
\+	Matches one or more occurrence of the previous character
\?	Matches zero or one occurrence of the previous character

Example:

Suppose we want to filter out lines where character 'a' precedes character 't'

We can use command like

```
cat  sample | grep "a\+t"
```

Brace expansion

The syntax for brace expansion is either a sequence or a comma separated list of items inside curly braces "{}". The starting and ending items in a sequence are separated by two periods "..".

Some examples:

```
u1@debian:~$ echo {aa,bb,cc,dd}
aa bb cc dd

u1@debian:~$ echo {0..11}
0 1 2 3 4 5 6 7 8 9 10 11

u1@debian:~$ echo {a..z}
a b c d e f g h i j k l m n o p q r s t u v w x y z

u1@debian:~$ echo  {0..9}b
0b 1b 2b 3b 4b 5b 6b 7b 8b 9b
```

In the above examples, the echo command creates strings using the brace expansion.

Users And Group Management

Users are accounts that can be used to login into a system.

Each user is identified by a ***unique identification number*** or ***UID*** by the system.
All the information of users in a system are stored in **/etc/passwd** file.

The hashed passwords for users are stored in **/etc/shadow** file.

Users can be divided into two categories on the basis of the level of access:

1. **Superuser/root/administrator** : Access to all the files on the system.
2. **Normal users** : Limited access.

When a new user is created, by default system takes following actions:

- Assigns UID to the user.
- Creates a home directory /home/.
- Sets the default shell of the user to be /bin/sh.
- Creates a private user group, named after the username itself.
- Contents of /etc/skel are copied to the home directory of the new user.
- *.bashrc*, *.bash_profile* and *.bash_logout* are copied to the home directory of new user.
- These files provide environment variables for this user’s session.

Description of contents of /etc/passwd File

This file is readable by any user but only root as read and write permissions for it. This file consists of the following colon separated information about users in a system:

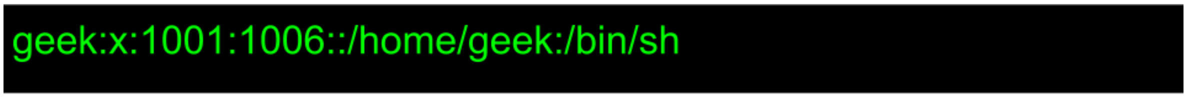
1. Username field
2. Password field
   - An `x` in this field denotes that the encrypted password is stored in the /etc/shadow file.
3. The user ID number (UID)

4. User's group ID number (GID)
5. Additional information field such as the full name of the user or comment (GECOS)
6. Absolute path of user's home directory
7. Login shell of the user

Syntax:

[username]:[password]:[UID]:[GID]:[GECOS]:[home_dir]:[shell_path]

Example:



```
geek:x:1001:1006::/home/geek:/bin/sh
```

Description of contents of the /etc/shadow File

This file is readable and writable by only by root user. This file consists of the following colon separated information about password of users in a system:

1. User name field
2. Password field
3. Contains an encrypted password.
    - A blank entry, {:: }, indicates that a password is not required to login into that user's account.
    - An asterisk, {:*:}, indicates the account has been disabled.
4. Last Password Change
    - This field denotes the number of days since the date of last password change counted since UNIX time (1-Jan-1970).
5. The minimum number of days after which the user can change his password.
6. Password validity
    - Denotes the number of days after which the password will expire.
7. Warning period

- Denotes the number of days before the password expiry date, from which the user will start receiving warning notification for password change.

8. Account validity

- Denotes the number of days after which the account will be disabled, once the password is expired.

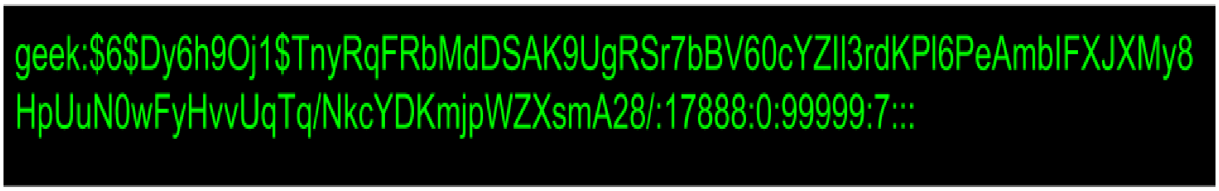
9. Account disability

- This field denotes the number of days since which the account had been disabled counted from UNIX time (1-Jan-1970).

Syntax:

[username]:[enc_pwd]:[last_pwd_change]:[pwd_validity]:[warn_date]:[acc_validity]:[acc_disablity]

Example:



```
geek:$6$Dy6h9Oj1$TnyRqFRbMdDSAK9UgRSr7bBV60cYZIl3rdKPl6PeAmbIFXJXMy8
HpUuN0wFyHvvUqTq/NkcYDKmjpWZXsmA28/:17888:0:99999:7:::
```

Creating a User

Using Terminal

Step 1) Use command sudo adduser

```
sudo adduser user1
Adding user `user1' ...
Adding new group `user1' (1002) ...
Adding new user `user1' (1002) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
New password:
```

Step 2) Enter password for the new account and confirm

```
Retype new password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
        Full Name []: user1
```

Step 3) Enter details of the new user and press Y

```
Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

Deleting, disabling account

Terminal

For disabling an account using Terminal, remove the password set on the account.

```
sudo passwd -l 'username'
```

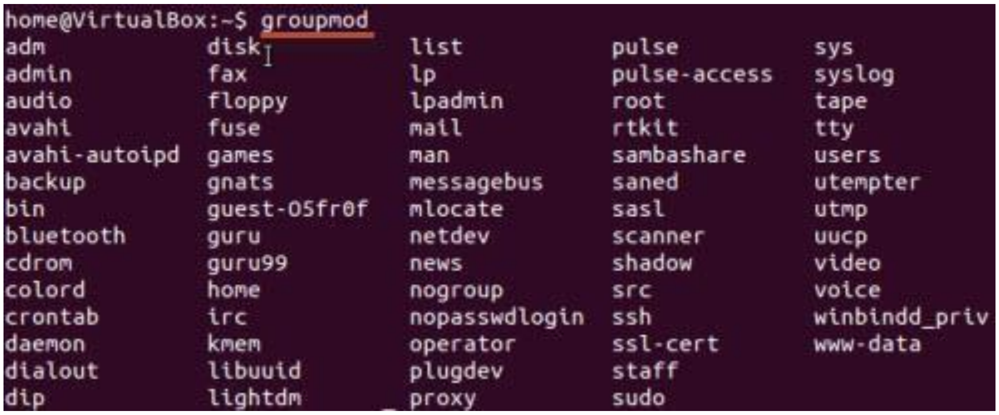
To delete an account, use the command -

```
sudo userdel -r 'username'
```

Adding users to the usergroups

You can view the existing groups on your Linux operating system by entering the following Linux user commands:

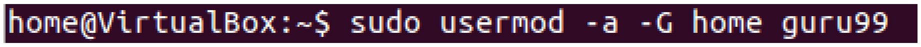
```
groupmod "Press Tab key twice"
```



```
home@VirtualBox:~$ groupmod
adm             disk            list            pulse           sys
admin           fax             lp              pulse-access    syslog
audio           floppy          lpadmin         root            tape
avahi           fuse            mail            rtkit           tty
avahi-autoipd   games           man             sambashare      users
backup          gnats           messagebus      saned           utempter
bin             guest-O5fr0f    mlocate         sasl            utmp
bluetooth       guru            netdev          scanner         uucp
cdrom           guru99          news            shadow          video
colord          home            nogroup         src             voice
crontab         irc             nopasswdlogin   ssh             winbindd_priv
daemon          kmem            operator        ssl-cert        www-data
dialout         libuuid         plugdev         staff
dip             lightdm         proxy           sudo
```

Now to add a user to a group, use the following syntax:

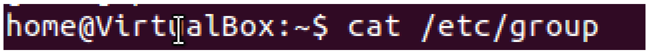
sudo usermod -a -G GROUPNAME USERNAME



```
home@VirtualBox:~$ sudo usermod -a -G home guru99
```

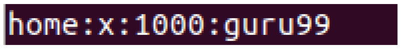
The system would ask for authentication and then it would add the user to the group.

You can check whether the user is in a group by this command.



```
home@VirtualBox:~$ cat /etc/group
```

And it would show it as

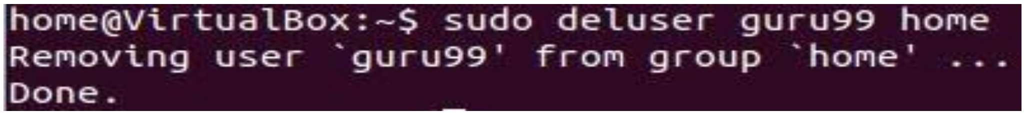


```
home:x:1000:guru99
```

Removing a user from Usergroup

Use the following syntax for removing a user.

sudo deluser USER GROUPNAME



```
home@VirtualBox:~$ sudo deluser guru99 home
Removing user `guru99' from group `home' ...
Done.
```

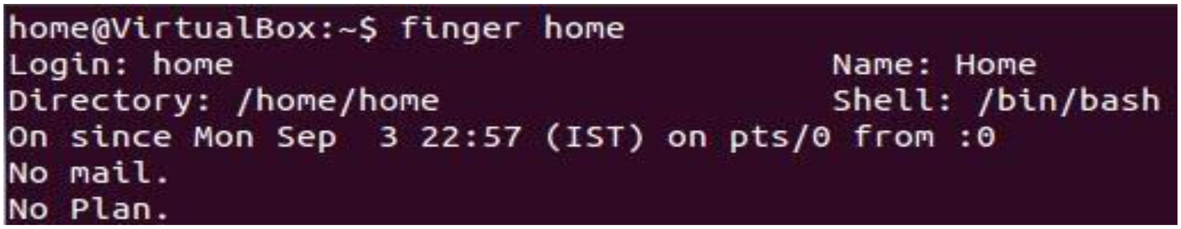
Finger

This command is used to **procure information of the users on a Linux machine**. You can use it on both local & remote machines

The syntax 'finger' gives data on all the logged users on the remote and local machine.

```
u1@debian:~$ finger

Login    Name     Tty     Idle    Login Time      Office          Office Phone

  u1      u1      tty2     1d     Mar 4 21:41     (tty2)

  u1      u1      pts/1   23:55   Mar 5 00:50     (192.168.221.1)
```

The syntax 'finger username' specifies the information of the user in User administration in Linux.



```
home@VirtualBox:~$ finger home
Login: home                                 Name: Home
Directory: /home/home                       Shell: /bin/bash
On since Mon Sep  3 22:57 (IST) on pts/0 from :0
No mail.
No Plan.
```

Linux/Unix user management commands

User management in Linux is done by using Linux administration commands. Here is a list of user management commands in Linux:

Command	Description
sudo adduser username	Adds a user

sudo passwd -l 'username'	Disable a user
sudo userdel -r 'username'	Delete a user
sudo usermod -a -G GROUPNAME USERNAME	Add user a to a usergroup
sudo deluser USER GROUPNAME	Remove user from a user group
Finger	Gives information on all logged in user
finger username	Gives information of a particular user

Groups

In Linux, groups can be defined as a way to organize users that need the same type of access to a directory or file.

To create a new group named finances, do

```
addgroup finances
```

To remove it from the system, use

```
delgroup finances
```

The information for the new group is stored in /etc/group, where each line shows the name of the group and the user accounts that are associated with it.

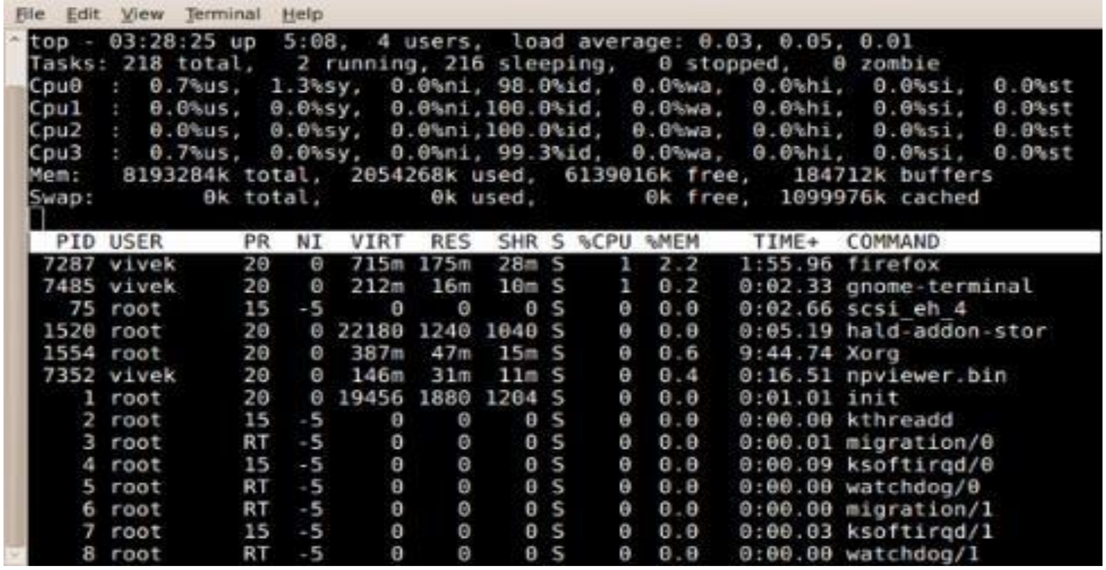
MONITORING & MANAGING PERFORMANCE/PROCESS

A process is a running instance of a launched, executable program. So being a System Engineer it is not easy to monitor Linux server/system performances and keep them up and running.

REAL- TIME PROCESS MONITORING BY ‘ TOP ‘:

The ‘ top ‘ command line, top program show a dynamic view of system processes and displaying a summary or it is used to dipslay all the running and active real-time processes in ordered list. It display **CPU usage**, **Memory usage**, **Swap Memory**, **Cache Size**, **Buffer Size**, **Process PID**, **User**, **Command**s and much more. It also shows high **memory** and **cpu** utilization of a running processess. And ‘ top ‘ command continuously refreshes at a configurable interval and provides above summary in a column view. The top command is more useful for system administrator to monitor and take correct action when required.

[u1@debian ~] $ top



```
top - 03:28:25 up  5:08,  4 users,  load average: 0.03, 0.05, 0.01
Tasks: 218 total,   2 running, 216 sleeping,   0 stopped,   0 zombie
Cpu0  :  0.7%us,  1.3%sy,  0.0%ni, 98.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :  0.7%us,  0.0%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   8193284k total,  2054268k used,  6139016k free,   184712k buffers
Swap:        0k total,        0k used,        0k free,  1099976k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7287 vivek     20   0  715m 175m  28m S    1  2.2   1:55.96 firefox
 7485 vivek     20   0  212m  16m  10m S    1  0.2   0:02.33 gnome-terminal
   75 root      15  -5     0    0    0 S    0  0.0   0:02.66 scsi_eh_4
 1520 root      20   0 22180 1240 1040 S    0  0.0   0:05.19 hald-addon-stor
 1554 root      20   0  387m  47m  15m S    0  0.6   9:44.74 Xorg
 7352 vivek     20   0  146m  31m  11m S    0  0.4   0:16.51 npviewer.bin
    1 root      20   0 19456 1880 1204 S    0  0.0   0:01.01 init
    2 root      15  -5     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S    0  0.0   0:00.01 migration/0
    4 root      15  -5     0    0    0 S    0  0.0   0:00.09 ksoftirqd/0
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/0
    6 root      RT  -5     0    0    0 S    0  0.0   0:00.00 migration/1
    7 root      15  -5     0    0    0 S    0  0.0   0:00.03 ksoftirqd/1
    8 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/1
```

To quit press **q**, for help press **h**.

1. To Save Process Snapshot to a file by type the following command:

```
[u1@debian ~]$ top -b -n1 > /tmp/process.log
```

Or you can email result to yourself:

```
[u1@debian ~] $  top -b -n1 | mail -s  'Process snapshot'  you@example.com
```

LINUX PROCESS MONITORING BY ‘ HTOP ‘ :

“htop” is a third party tool and doesn’t included in Linux systems, you need to install it by using **YUM** package manager tool for Redhat and for Ubuntu need to install it by **apt-get.** It is much similar to Linux **top command** but it has some advanced features like **user friendly interface to manage process**, **shortcut keys** etc.
It allows to scroll the list vertically and horizontally to see all processes and their full command lines. Tasks related to processes (killing, renicing) can be done without entering their PIDs.

```
[u1@debian ~] $  sudo   apt-get    install    htop
 [u1@debian ~] $ htop
```

LINUX PROCESS MONITORING BY ‘ ATOP ‘:

‘ **atop** ‘ command or program is an interactive monitor to view the **load on a Linux system**. It shows the occupation of the most **critical hardware resources (from a performance point of view) on system level, i.e. cpu, memory, disk and network.** It also shows which processes are responsible for the indicated load with respect to **cpu- and memory load on process level; disk- and network load is only shown per process if a kernel patch has been installed**.

[u1@debian ~] $ atop

VIRTUAL MEMORY STATISTICS – VMSTAT COMMAND:

Linux **VmStat** command used to display statistics of **system processes**, **I/O blocks**, **interrupts**, **CPU activity**, **virtual memory**, **kernerl threads**, **disks**, **etc**. By default vmstat command is not available under Linux systems you need to install a package called **sysstat** that includes a vmstat program.

[u1@debian ~] $ vmstat

LIST OPEN FILES – LSOF

‘ lsof ‘ command used in many **Linux/Unix** like system that is used to display list of all the open files and the processes. The open files included are **disk files**, **network sockets**, **pipes**, **devices** and **processes**.

One of the main reason for using this command is when a disk cannot be unmounted and displays the error that files are being used or opened. With this command you can easily identify which files are in use.

[u1@debian ~] $ lsof

TO MONITOR LINUX DISK I/O – IOTOP COMMAND

‘ iotop ‘ command is also much similar to **top command** and **htop program**, but it has accounting function to monitor and display real time **Disk I/O** and **processes**. This tool is much useful for finding the exact process and high used disk read/writes of the processes.

[u1@debian ~] $ iotof

SHOW ALL RUNNING PROCESSES IN LINUX – ‘ PS’ COMMAND:

We need to use the ps command. It provide information about the currently running processes, including their process identification numbers (PIDs).
The ‘ **ps** ‘ command gives a snapshot of the current processes. If you want a repetitive update of this status, use **top, atop, and/or htop** command as described above.

1. **top command : Display and update sorted information about processes.**
2. **atop : Advanced System & Process Monitor.**
3. **htop : Interactive process viewer.**

Type the following **ps command** to display all running process:

```
[u1@debian ~] $ ps aux | less
```

Where,

- -A: select all processes
- a: select all processes on a terminal, including those of other users
- x: select processes without controlling ttys

1. **To see everyprocess on the system**

```
[u1@debian ~] $ ps    -A

[u1@debian~]$ ps –e
```

2. **To see every process except those running as u1**

```
[u1@debian ~]$ ps -U u1 -u u1 –N
```

3. **To see process run by user abc**

```
[u1@debian ~]$ ps -u abc
```

DISPLAY A TREE OF PROCESSES- ‘PSTREE’

‘ pstree ‘ shows running processes as a tree. The tree is rooted at either pid or init if pid is omitted. If a user name is specified, all process trees rooted at processes owned by that user are shown.

```
[u1@debian ~]$ pstree
```

1. **To Print a process tree using ‘ps’**

```
[u1@debian ~]$ ps -ejH
[u1@debian ~]$ ps axjf
```

2. To Get info about threads by the following command:

```
[u1@debian ~]$ ps -eLf
[u1@debian ~]$ ps axms
```

LOOKUP PROCESS BY PGREP:

‘ **pgrep** ‘ command, looks through the currently running processes and lists the process IDs which matches the selection criteria to screen. For example display **httpd** process id:

```
[u1@debian ~]$ pgrep httpd
```

Sample outputs:

```
3356
```

Following command will list the **process called sshd which is owned by a user called root:**

```
[u1@debian ~]$ pgrep -u root sshd
```

Secure SSH Service

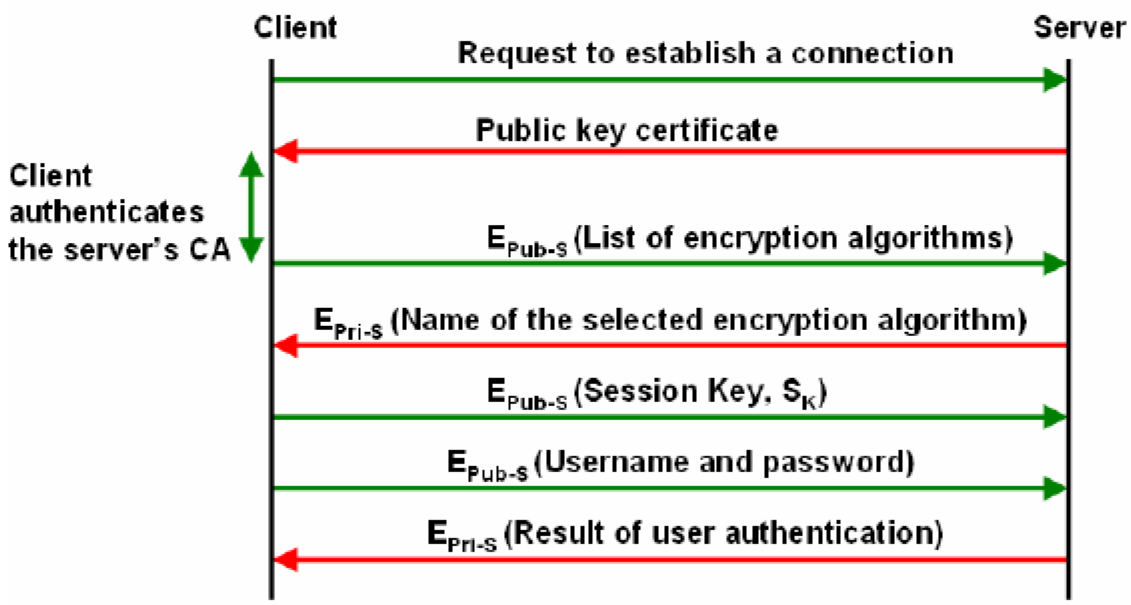
SSH – An Introduction:

SSH stands for Secure Shell, a cryptographic network protocol used for connecting to Linux/Unix servers remotely via a command line interface.

The default port on which SSH service works is 22 (which is configurable) to exchange data between the remote users and the server. The SSH program was developed to improve the security of applications which were used earlier like **telnet** or **rsh**.

The following events occur when two parties connect via SSH:

1. A request to establish a connection is made, so that the client can verify his communication with the correct server.
2. The connection between the client and the server is encrypted on the transport layer.
3. The server checks the information provided by the client for authentication.
4. Once the encrypted connection is established, the client and the server are now ready to exchange information.


Client
Server
Request to establish a connection
Public key certificate
Client authenticates the server's CA
E_Pub-S (List of encryption algorithms)
E_Pri-S (Name of the selected encryption algorithm)
E_Pub-S (Session Key, S_K)
E_Pub-S (Username and password)
E_Pri-S (Result of user authentication)

Security measures necessary to safeguard SSH service:

Here are some security measures which can be useful in securing the SSH service in Linux:

1. **Configure the SSH port:**

   The default port is 22, as stated above. If an attacker attempts to request access on the given port, he is already one step ahead. Now he only has to do hit and trial for the password. If we configure the SSH port to be other than 22, we can secure it against possible attacks.

   Run the command below to change the default port in the SSH configuration file:

```
# sed -i 's/#Port [0-9]*/Port 1337/' /etc/ssh/sshd_config
```

It will change any Port xxx, where xxx is – any number to Port 1337. To verify, run the below command:

```
# cat  /etc/ssh/sshd_config  |  grep Port
```

Now that you've changed the port, add the port in the firewall, so that the SSH service listens on port 1337. Run the below commands to achieve that:

```
#iptables -A INPUT  -i  eth0 -p tcp –dport 1337 -j ACCEPT
#service iptables save
#service  iptables  restart
```

2. Disable root login:

Disable root login! How are we supposed to login to our server?

Default user or the user through which we login to the server is the root and this gives an attacker an edge to access the complete system. He already knows the username. He only has to work on the password. We will create a user in our server- a user which will be used only for SSH login.

Run the command below to create a user:

```
# useradd 1345345
```

Set the user password to something complex:

```
#passwd 1345345
Changing password for user 1345345.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

We will now change our SSH configuration to disable the root login.

```
# sed -i ‘s/#PermitRootLogin yes/PermitRootLogin no/’ /etc/ssh/sshd_config
```

Once the root login has been disabled, we will now allow the users we created earlier.

```
# echo “AllowUsers 1345345” >> /etc/ssh/sshd_config
```

Finally, to allow the user to run su – command after logging in to the server, add the user to the wheel group to make things easier to manage.

```
# usermod -aG wheel 1345345
```

Now after making all the required changes, restart the SSH service with below command:

```
# /etc/init.d/sshd restart
```

or

```
# service  sshd restart
```

or if you are using Centos 7/ Red Hat Enterprise Linux 7, use this to restart:

```
# systemctl restart  sshd.service
```

3. **Allow access from specific IP addresses:**

   If you have a static IP address, such as your corporate network, you are suggested to allow SSH from that IP only. To do that, run the commands below, where the public IP address is as per your network:

```
# echo “sshd : 192.168.1.1 : ALLOW” >> /etc/hosts.allow
# echo “sshd : 192.168.1.2 : ALLOW” >> /etc/hosts.allow
# echo “sshd : ALL : DENY” >> /etc/hosts.allow
```

4. **Physical Access/User Security:**

   Make sure your machines are well protected against viruses and are not for public use, if you use Windows to access SSH. Change passwords regularly and store them at a secure location.

Linux Logs

How to View Linux Logs

Linux system administrators often need to look at log files for troubleshooting purposes. In fact, this is the first thing any sysadmin would do.

Linux and the applications that run on it can generate all different types of messages, which are recorded in various log files. Linux uses a set of configuration files, directories, programs, commands and daemons to create, store and recycle these log messages. Knowing where the system keeps its log files and how to make use of related commands can therefore help save valuable time during troubleshooting.

Default Log File Location

The default location for log files in Linux is /var/log.

You can view the list of log files in this directory with a simple ls -l /var/log command.

```
[root@TestLinux ~]# ls -l /var/log
total 1472
-rw ---- --. 1 root root   4524 Nov 15 16:04 anaconda.ifcfg.log
-rw ---- --. 1 root root  59041 Nov 15 16:04 anaconda.log
-rw ---- --. 1 root root  42763 Nov 15 16:04 anaconda.program.log
-rw ---- --. 1 root root 299910 Nov 15 16:04 anaconda.storage.log
-rw ---- --. 1 root root  40669 Nov 15 16:04 anaconda.syslog
-rw ---- --. 1 root root  57061 Nov 15 16:04 anaconda.xlog
-rw ---- --. 1 root root   1829 Nov 15 16:04 anaconda.yum.log
drwxr-x---. 2 root root   4096 Nov 15 16:11 audit
-rw-r--r--  1 root root   2252 Dec  9 10:27 boot.log
-rw-------  1 root utmp    384 Dec  9 10:31 btmp
-rw ---- --. 1 root utmp   1920 Nov 28 09:28 btmp-20131202
drwxr-xr-x  2 root root   4096 Nov 29 15:47 ConsoleKit
-rw-------  1 root root   2288 Dec  9 11:01 cron
-rw ---- --. 1 root root   8809 Dec  2 17:09 cron-20131202
-rw-r--r--  1 root root  21510 Dec  9 10:27 dmesg
```

```
-rw-r--r--  1 root root  21351 Dec  6 16:37 dmesg.old
-rw-r--r--. 1 root root 165665 Nov 15 16:04 dracut.log
-rw-r--r--. 1 root root 146876 Dec  9 10:44 lastlog
-rw-------  1 root root    950 Dec  9 10:27 maillog
-rw ---- --. 1 root root   4609 Dec  2 17:00 maillog-20131202
-rw-------  1 root root 123174 Dec  9 10:27 messages
-rw ---- --. 1 root root 458481 Dec  2 17:00 messages-20131202
-rw-------  1 root root   2644 Dec  9 10:44 secure
-rw ---- --. 1 root root  15984 Dec  2 17:00 secure-20131202
-rw-------  1 root root      0 Dec  2 17:09 spooler
-rw ---- --. 1 root root      0 Nov 15 16:02 spooler-20131202
-rw ---- --. 1 root root      0 Nov 15 16:02 tallylog
-rw-rw-r--. 1 root utmp  89856 Dec  9 10:44 wtmp
-rw-------  1 root root   3778 Dec  6 16:48 yum.log
```

Viewing Log File Contents

Here are some common log files you will find under /var/log:

- wtmp
- utmp
- dmesg
- messages
- maillog or mail.log
- spooler
- auth.log or secure

The wtmp and utmp files keep track of users logging in and out of the system. You cannot directly read the contents of these files using cat– there are specific commands for that.

We will now use some of these commands.

To see who is currently logged in to the Linux server, simply use the who command. This command gets its values from the /var/run/utmp file (for CentOS and Debian) or /run/utmp (for Ubuntu).

Here is an example from CentOS:

```
[root@TestLinux ~]# who
root     tty1         2013-12-09 10:44
root     pts/0        2013-12-09 10:29 (10.0.2.2)
sysadmin pts/1        2013-12-09 10:31 (10.0.2.2)
joeblog  pts/2        2013-12-09 10:39 (10.0.2.2)
```

In this particular case, I am the sole user of the system. I was running the server from an Oracle VirtualBox and accessing it as root from both the console and an SSH session. Two other user accounts (sysadmin and joebolg) were also accessing the system.

The last command tells us the login history of users:

```
[root@TestLinux ~]# last | grep sysadmin
sysadmin pts/1        10.0.2.2         Mon Dec  9 10:31   still logged in
sysadmin pts/0        10.0.2.2         Fri Nov 29 15:42 - crash  (00:01)
sysadmin pts/0        10.0.2.2         Thu Nov 28 17:06 - 17:13  (00:06)
sysadmin pts/0        10.0.2.2         Thu Nov 28 16:17 - 17:05  (00:48)
sysadmin pts/0        10.0.2.2         Thu Nov 28 09:29 - crash  (06:04)
sysadmin pts/0        10.0.2.2         Wed Nov 27 16:37 - down   (00:29)
sysadmin tty1                          Wed Nov 27 14:05 - down   (00:36)
sysadmin tty1                          Wed Nov 27 13:49 - 14:04  (00:15)
```

In this example, I am trying to find the login history of the user sysadmin. As you can see, there were couple of instances where he managed to crash the system.

To find out when was the system last rebooted, we can run the following command:

```
[root@TestLinux ~]# last reboot
```

The result may look like this

```
reboot   system boot  2.6.32-358.el6.x Mon Dec  9 10:27 - 10:47  (00:19)
reboot   system boot  2.6.32-358.el6.x Fri Dec  6 16:37 - 10:47 (2+18:10)
reboot   system boot  2.6.32-358.el6.x Fri Dec  6 16:28 - 16:36  (00:08)
reboot   system boot  2.6.32-358.el6.x Fri Dec  6 11:06 - 16:36  (05:29)
reboot   system boot  2.6.32-358.el6.x Mon Dec  2 17:00 - 16:36 (3+23:36)
reboot   system boot  2.6.32-358.el6.x Fri Nov 29 16:01 - 16:36 (7+00:34)
reboot   system boot  2.6.32-358.el6.x Fri Nov 29 15:43 - 16:36 (7+00:53)
...
...
wtmp begins Fri Nov 15 16:11:54 2013
```

To see when did someone last log in to the system, use lastlog:

```
[root@TestLinux ~]# lastlog
```

In my system, the output looked like this:

```
Username         Port        From             Latest
root             tty1                         Mon Dec  9 10:44:30 +1100 2013
bin                                          **Never logged in**
daemon                                       **Never logged in**
adm                                          **Never logged in**
lp                                           **Never logged in**
sync                                         **Never logged in**
shutdown                                     **Never logged in**
halt                                         **Never logged in**
mail                                         **Never logged in**
uucp                                         **Never logged in**
operator                                     **Never logged in**
games                                        **Never logged in**
gopher                                       **Never logged in**
ftp                                          **Never logged in**
nobody                                       **Never logged in**
vcsa                                         **Never logged in**
saslauth                                     **Never logged in**
postfix                                      **Never logged in**
sshd                                         **Never logged in**
```

```
sysadmin         pts/1    10.0.2.2         Mon Dec  9 10:31:50 +1100 2013
dbus                                        **Never logged in**
joeblog          pts/2    10.0.2.2         Mon Dec  9 10:39:24 +1100 2013
```

For other text-based log files, you can use cat, head or tail commands to read the contents.

In the example below, I am trying to look at the last ten lines of /var/log/messages file in a Debian box:

```
debian@debian:~$ sudo tail /var/log/messages
```

Output:

```
Dec 16 01:21:08 debian kernel: [    9.584074] Bluetooth: BNEP (Ethernet
Emulation) ver 1.3
Dec 16 01:21:08 debian kernel: [    9.584074] Bluetooth: BNEP filters: protocol
multicast
Dec 16 01:21:08 debian kernel: [    9.648220] Bridge firewalling registered
Dec 16 01:21:08 debian kernel: [    9.696728] Bluetooth: SCO (Voice Link) ver 0.6
Dec 16 01:21:08 debian kernel: [    9.696728] Bluetooth: SCO socket layer
initialized
Dec 16 01:21:08 debian kernel: [    9.832215] lp: driver loaded but no devices
found
Dec 16 01:21:08 debian kernel: [   9.868897] ppdev: user-space parallel port
driver
Dec 16 01:21:11 debian kernel: [   12.748833] [drm] Initialized drm 1.1.0
20060810
Dec 16 01:21:11 debian kernel: [   12.754412] pci 0000:00:02.0: PCI INT A ->
Link[LNKB] -> GSI 11 (level, low) -> IRQ 11
Dec 16 01:21:11 debian kernel: [   12.754412] [drm] Initialized vboxvideo 1.0.0
20090303 for 0000:00:02.0 on minor 0
```

The rsyslog Daemon

At the heart of the logging mechanism is the rsyslog daemon. This service is responsible for listening to log messages from different parts of a Linux system and routing the message to an appropriate log file in the /var/log directory. It can also forward log messages to another Linux server.

The rsyslog Configuration File

The rsyslog daemon gets its configuration information from the rsyslog.conf file. The file is located under the /etc directory.

Basically, the rsyslog.conf file tells the rsyslog daemon where to save its log messages. This instruction comes from a series of two-part lines within the file.

This file can be found at rsyslog.d/50-default.conf on ubuntu.

The two part instruction is made up of a *selector* and an *action*. The two parts are separated by white space.

The selector part specifies what's the source and importance of the log message and the action part says what to do with the message.

The selector itself is again divided into two parts separated by a dot (.). The first part before the dot is called *acility (the origin of the message) and the second part after the dot is called priority (the severity of the message).

Together, the facility/priority and the action pair tell rsyslog what to do when a log message matching the criteria is generated.

Here is excerpt from a CentOS rsyslog.conf file:

```
# rsyslog v5 configuration file
...
...
# Include all config files in /etc/rsyslog.d/
IncludeConfig /etc/rsyslog.d/*.conf
```

```
#### RULES ####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*  /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog



# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
...
...
```

To understand what this all means, let’s consider the different types of facilities recognized by Linux. Here is a list:

- **auth** or **authpriv**: Messages coming from authorization and security related events
- **kern**: Any message coming from the Linux kernel
- **mail**: Messages generated by the mail subsystem
- **cron**: Cron daemon related messages

- **daemon**: Messages coming from daemons
- **news**: Messages coming from network news subsystem
- **lpr**: Printing related log messages
- **user**: Log messages coming from user programs
- **local0 to local7**: Reserved for local use

And here is a list of priorities in ascending order:

- **debug**: Debug information from programs
- **info**: Simple informational message - no intervention is required
- **notice**: Condition that may require attention
- **warn**: Warning
- **err**: Error
- **crit**: Critical condition
- **alert**: Condition that needs immediate intervention
- **emerg**: Emergency condition

So now let's consider the following line from the file:

```
cron.*                /var/log/cron
```

This just tells the rsyslog daemon to save all messages coming from the cron daemon in a file called /var/log/cron. The asterix (*) after the dot (.) means messages of all priorities will be logged. Similarly, if the facility was specified as an asterix, it would mean all sources.

Facilities and priorities can be related in a number of ways.

In its default form, when there is only one priority specified after the dot, it means all events equal to or greater than that priority will be trapped. So the following directive causes any messages coming from the mail subsystem with a priority of warning or higher to be logged in a specific file under /var/log:

```
mail.warn             /var/log/mail.warn
```

This will log every message equal to or greater than the warn priority, but leave everything below it. So messages with err, crit, alert or emerg will also be recorded in this file.

Using an equal sign (=) after the dot (.) will cause only the specified priority to be logged. So if we wanted to trap only the info messages coming from the mail subsystem, the specification would be something like the following:

```
mail.=info          /var/log/mail.info
```

Again, if we wanted to trap everything from mail subsystem except info messages, the specification would be something like the following

```
mail.!info          /var/log/mail.info
```

or

```
mail.!=info         /var/log/mail.info
```

In the first case, the mail.info file will contain everything with a priority lower than info. In the second case, the file will contain all messages with a priority above info.

Multiple facilities in the same line can be separated by commas.

Multiple sources (*facility.priority*) in the same line is separated by semicolon.

When an action is marked as an asterix (*), it means all users. This entry in my CentOS rsyslog.conf file is saying exactly that:

```
# Everybody gets emergency messages
*.emerg                                                  *
```

Try to see what's the rsyslog.conf is saying in your Linux system. Here is an excerpt from the Debian server I am running:

```
#  /etc/rsyslog.conf    Configuration file for rsyslog.
#
#           For more information see
#           /usr/share/doc/rsyslog-doc/html/rsyslog_conf.html
...
...
auth,authpriv.*         /var/log/auth.log
*.*;auth,authpriv.none      -/var/log/syslog
#cron.*             /var/log/cron.log
daemon.*            -/var/log/daemon.log
kern.*              -/var/log/kern.log
lpr.*               -/var/log/lpr.log
mail.*              -/var/log/mail.log
user.*              -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info           -/var/log/mail.info
mail.warn           -/var/log/mail.warn
mail.err            /var/log/mail.err
#
# Logging for INN news system.
#
news.crit           /var/log/news/news.crit
news.err            /var/log/news/news.err
news.notice         -/var/log/news/news.notice
```

As you can see, Debian saves all security/authori zation level messages in /var/log/auth.log whereas CentOS saves it under /var/log/secure.

The configurations for rsyslog can come from other custom files as well. These custom configuration files are usually located in different directories under /etc/rsyslog.d. The rsyslog.conf file includes these directories using $IncludeConfig directive.

Here is what it looks like

```
#  Default logging rules can be found in /etc/rsyslog.d/50-default.conf
....
....
$IncludeConfig /etc/rsyslog.d/*.conf
```

The contents under the /etc/rsyslog.d directory looks like the following:

```
-rw-r--r-- 1 root root  311 Mar 17  2012 20-ufw.conf
-rw-r--r-- 1 root root  252 Apr 11  2012 21-cloudinit.conf
-rw-r--r-- 1 root root 1655 Mar 30  2012 50-default.conf
```

Now the destination for a log message does not necessarily have to be a log file; the message can be sent to a user's console. In this case, the action field will contain the username. If more than one user needs to receive the message, their usernames are separated by commas. If the message needs to be broadcast to every user, it's specified by an asterix (*) in the action field.

Because of being part of a network operating system, rsyslog daemon can not only save log messages locally, it can also forward them to another Linux server in the network or act as a repository for other systems. The daemon listens for log messages in UDP port 514. The example below will forward kernel critical messages to a server called "texas".

```
kern.crit           @texas
```

Network configuration

Basic network configuration includes setting a static or dynamic IP address, adding a gateway, DNS server information. There are different ways to configure the network on Debian OS.

Method 1: Use ifconfig and route command

In this method, we will see how to configure network settings. However, remember, these settings will not be permanent. Once you reboot your system, the settings will be removed.

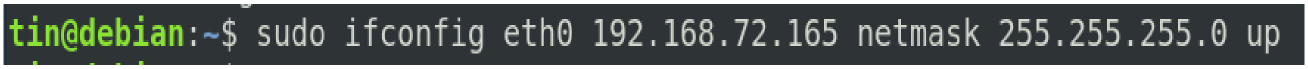
1. Assign an IP address to the interface

We will use ifconfig to assign an IP address to our network interface. Below is the syntax of the command:

```
$ sudo ifconfig <interface> <IP_address> netmask <subnetmask> up
```

In the following example, the command assigns the IP address 192.168.72.165 to the network interface eth0. The network mask is 24 (255.255.255.0) bits.

```
$ sudo ifconfig  eth0 192.168.72.165  netmask 255.255.255.0  up
```



```
tin@debian:~$ sudo ifconfig eth0 192.168.72.165 netmask 255.255.255.0 up
```

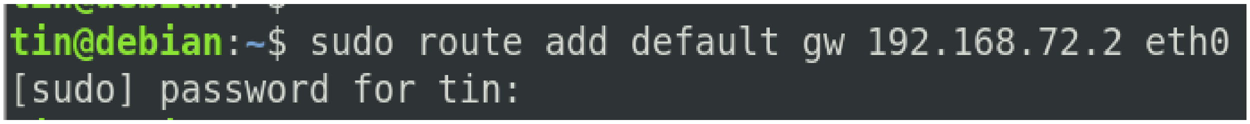
2. Set the Default Gateway

The default gateway is the address used to communicate with the outside network. To configure the default gateway, use the following command syntax:

```
$ sudo route add default gw <IP_address> <interface>
```

In the following example, I am using 192.68.72.2 as my default gateway address.

```
$ sudo route add default gw 192.168.72.2 eth0
```



```
tin@debian:~$ sudo route add default gw 192.168.72.2 eth0
[sudo] password for tin:
```

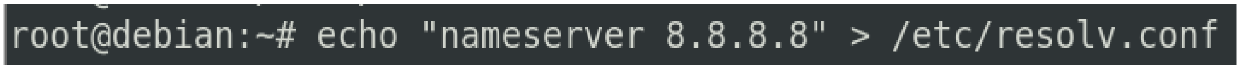
3. Set Your DNS server

DNS server resolves a domain name to an IP address so the browser can load Internet resources. To configure the DNS name server address, use the following command syntax:

```
$ echo “nameserver <IP_address>” > /etc/resolv.conf
```

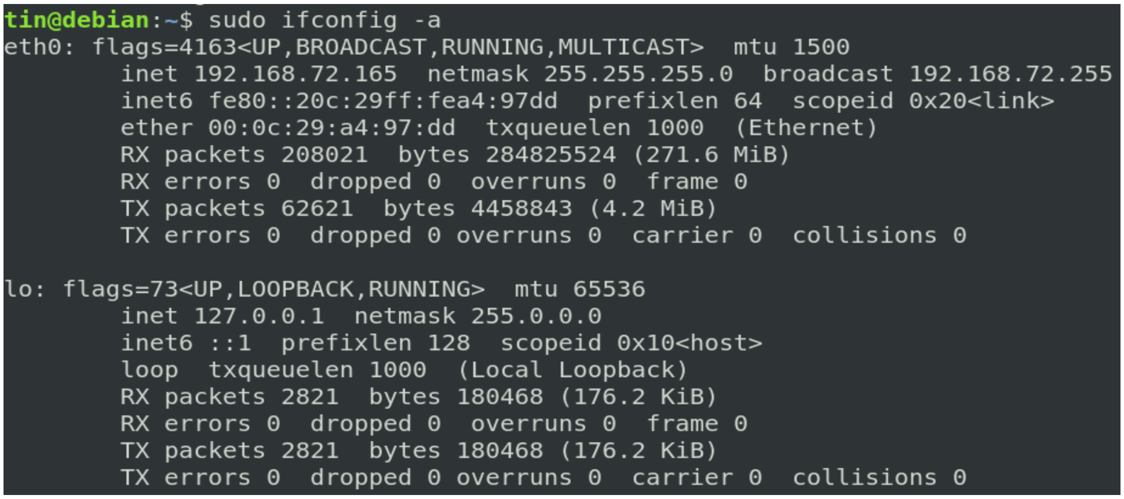
In the following example, I am setting Google’s public DNS IP address as my nameservers address that is 8.8.8.8.

```
$ echo “nameserver 8.8.8.8” > /etc/resolv.conf
```



```
root@debian:~# echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

Once done, you can test your configuration by running the ifconfig command as follows:



```
tin@debian:~$ sudo ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.72.165  netmask 255.255.255.0  broadcast 192.168.72.255
        inet6 fe80::20c:29ff:fea4:97dd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:a4:97:dd  txqueuelen 1000  (Ethernet)
        RX packets 208021  bytes 284825524 (271.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 62621  bytes 4458843 (4.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2821  bytes 180468 (176.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2821  bytes 180468 (176.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Remove IP address from a network interface

To remove an IP address from a network interface, run the following command in Terminal:

```
$ ip address del <IP_address> dev <interface>
```

Method 2: Change network settings by using the interfaces file

In this method, we will configure permanent network settings that your system will remember even after a reboot. For that, we will have to edit **/etc/network/interfaces** file using any text editor. Run the following command in terminal to do so:

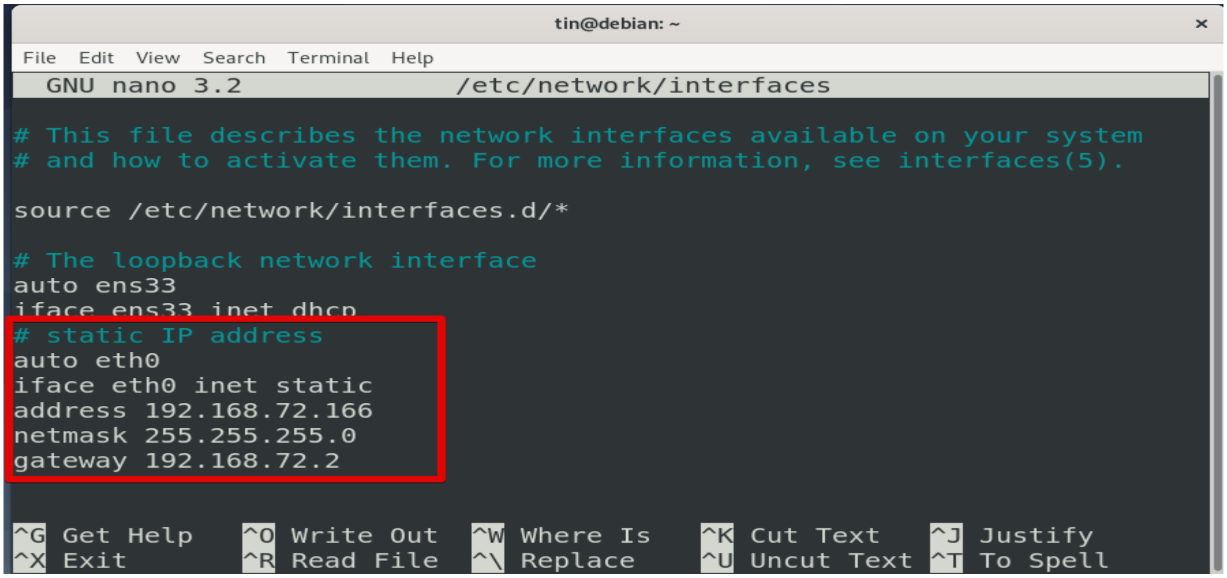
```
$ sudo nano /etc/network/interfaces
```

Then add the following lines in it:

```
auto eth0
```

```
iface eth0 inet static
address 192.168.72.165
netmask 255.255.255.0
gateway 192.168.72.2
```

Now press **Ctrl+O** and then **Ctrl+X** to save and exit the file.



```
  GNU nano 3.2             /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto ens33
iface ens33 inet dhcp
# static IP address
auto eth0
iface eth0 inet static
address 192.168.72.166
netmask 255.255.255.0
gateway 192.168.72.2


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell
```

Please note that the address, netmask and gateway line must start with leading whitespace! In case, you want to dynamically assign the address, use the following lines:

```
auto eth0
iface eth0 inet dhcp
```

Defining the (DNS) Nameservers

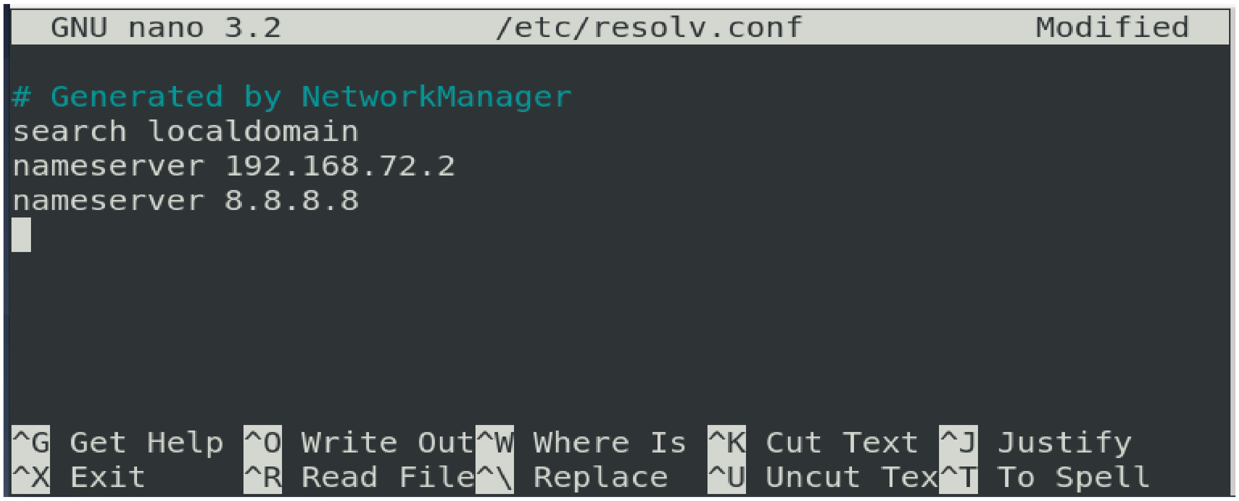
To add DNS server information, we will need to edit the **/etc/resolv.conf** file. Run the following command to do so:

```
$ nano /etc/resolv.conf
```

I am adding here two Nameservers. One is Google's public DNS server address and the other is my router's IP address.

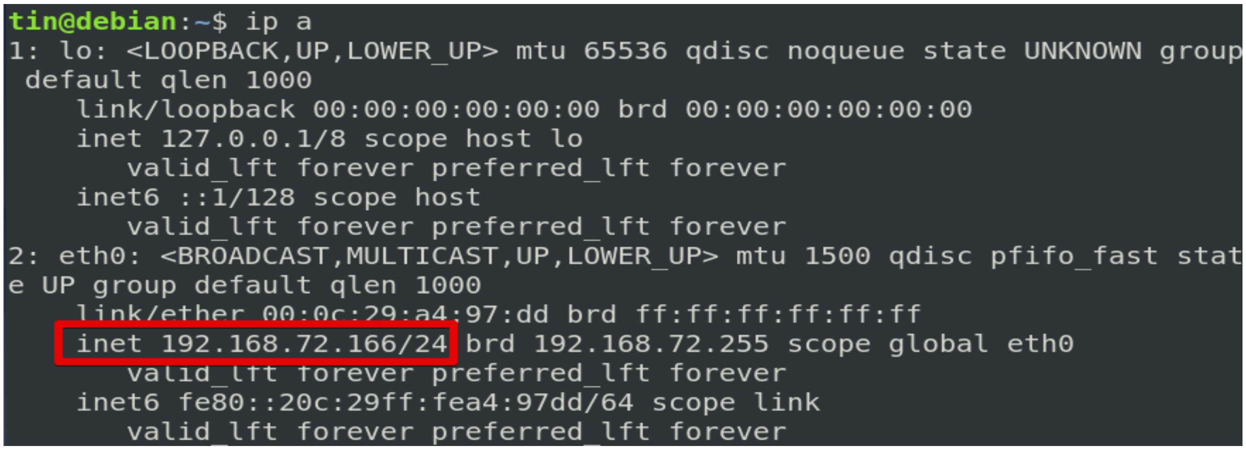
```
nameserver 8.8.8.8
nameserver 192.168.72.2
```

Now press **Ctrl+O** and then **Ctrl+X** to save and exit the file.



```
  GNU nano 3.2              /etc/resolv.conf                   Modified

# Generated by NetworkManager
search localdomain
nameserver 192.168.72.2
nameserver 8.8.8.8






^G Get Help  ^O Write Out^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File^\ Replace   ^U Uncut Tex^T To Spell
```

Once done, you can verify the IP address using **ip a** or **ifconfig** command.

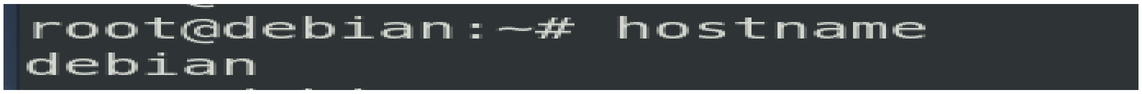


```
tin@debian:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
 default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast stat
e UP group default qlen 1000
    link/ether 00:0c:29:a4:97:dd brd ff:ff:ff:ff:ff:ff
    inet 192.168.72.166/24 brd 192.168.72.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fea4:97dd/64 scope link
       valid_lft forever preferred_lft forever
```

Setting up Hostname

Just like the IP address, a unique hostname is also used to recognize a system on a network. To find the current hostname of your system, run the below command in Terminal:

```
$ hostname
```

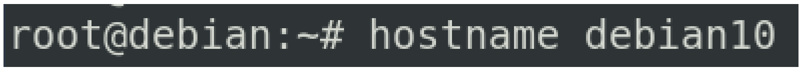


```
root@debian:~# hostname
debian
```

To change the hostname of the system, you can run the below command. But once you reboot your system, your original hostname will be restored.

```
$ hostname host_name
```

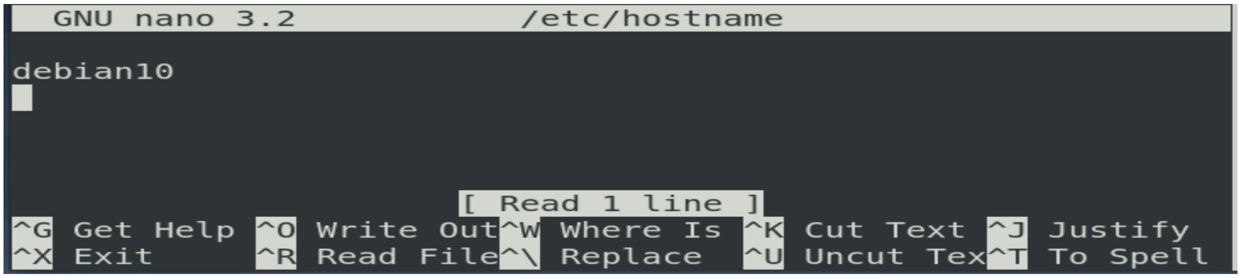
I am changing here my hostname from Debian to Debian10.



```
root@debian:~# hostname debian10
```

To permanently change the host name, you perform will need to edit hostname file located at **/etc/hostname**. Enter the below command to do so:

```
$ sudo nano /etc/hostname
```



```
  GNU nano 3.2            /etc/hostname

debian10


                      [ Read 1 line ]
^G Get Help  ^O Write Out^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File^\ Replace   ^U Uncut Tex^T To Spell
```

This file contains only the hostname of the file, change the old name to your desired name, and then press **Ctrl+O** and **Ctrl+X** to save and exit.
Some other useful commands you might require while setting up a network in a Debian OS:

Ping

It can be used to test connectivity between two systems on aLAN or WAN. To test connectivity to a device, type ping followed by IP or host name of that device:

```
$ ping <IP or hostname>
```

Arp:

Arp is used to translate IP addresses into Ethernet addresses. To print arp table, type:

```
$ arp –a
```

Route

It is used to display the routing table of a Linux system.

```
$ route
```

Host

It translates host names to IP addresses and vice versa.

To find IP against a specified domain:

```
$ host domain_name
```

To find a domain name against the specified IP address.

```
$ host IP_address
```

Enable and disable the interface

To enable up the interface, use:

```
$ ifup <interface>
```

To bring down the interface, use:

```
$ ifdown <interface>
```

Disk Management

BIOS / UEFI

When we power on a computer, the first program which starts is either a BIOS or a UEFI. This is the starting point of computer. It performs a series of diagnostic test to detect and connect CPU, Memory, Keyboard, Hard disks and other peripherals. This process is known as Power on Self-Test (POST). If all peripherals are connected without any issue, BIOS/UEFI will find and execute the boot loader program.

MBR partition scheme

- can be used in a hard disk which is 8TiB or less in size.
- To use a hard disk which is greater than 8TiB or to create a partition which is larger than 2TiB we need GPT partition scheme.

GPT Partition scheme

- In GPT maximum size of a partition is 8 ZiB.
- GPT allows maximum 128 partitions. GPT uses a 128 bit global unique ID (GUID) to identify the partition.
- GPT provides a lot of partitions (128) and much bigger space in each partition there is no need to divide the partitions in primary, extended and logi cal partitions.

SWAP Space

- Swap space is the special space in hard disk that is used as a temporary memory.
- This space can be allocated as a separate swap partition.
- Swap space is used only if a shortage of physical memory occurs.
- In shortage situation system moves recently unused data from memory to swap space.

- When requires, system moves back this data from swap to memory. This is the convenient way to improve kernel memory usage.

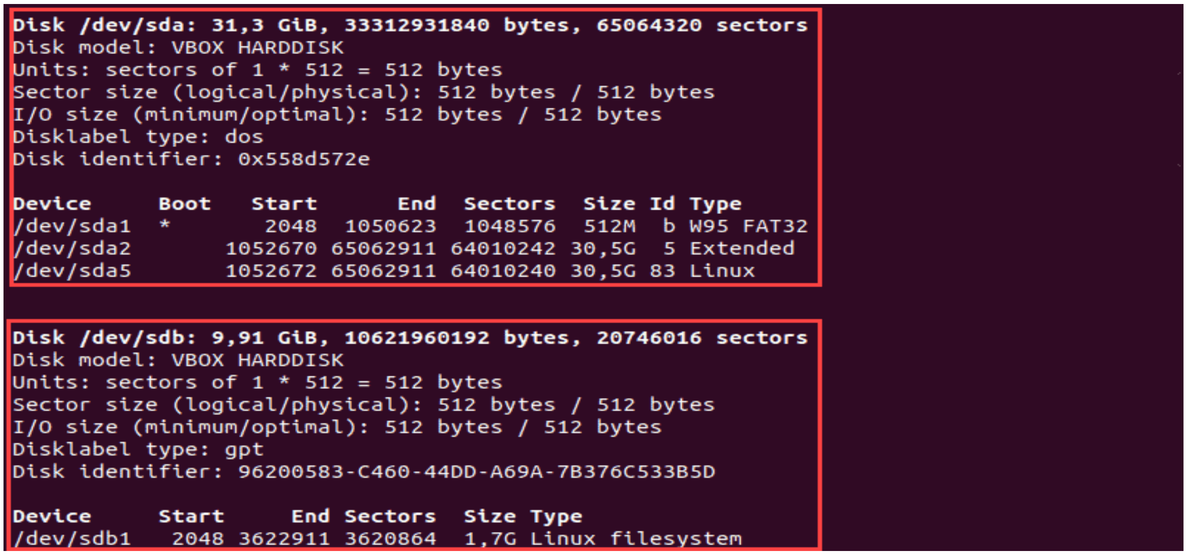
Follow the steps below to partition a disk in Linux by using the **fdisk** command.

Step 1: List Existing Partitions

Run the following command to list all existing partitions:

```
sudo fdisk -l
```

The output contains information about storage disks and partitions:



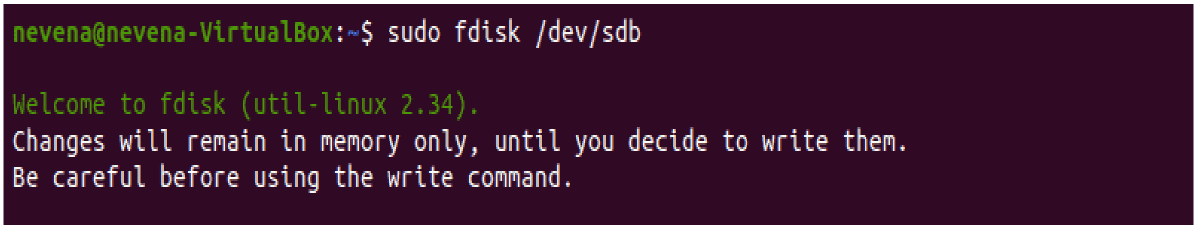
```
Disk /dev/sda: 31,3 GiB, 33312931840 bytes, 65064320 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x558d572e

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sda1  *       2048  1050623  1048576  512M  b W95 FAT32
/dev/sda2       1052670 65062911 64010242 30,5G  5 Extended
/dev/sda5       1052672 65062911 64010240 30,5G 83 Linux

Disk /dev/sdb: 9,91 GiB, 10621960192 bytes, 20746016 sectors
Disk model: VBOX HARDDISK
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: gpt
Disk identifier: 96200583-C460-44DD-A69A-7B376C533B5D

Device     Start     End Sectors  Size Type
/dev/sdb1   2048 3622911 3620864  1,7G Linux filesystem
```

Step 2: Select Storage Disk

Select the storage disk you want to create partitions on by running the following command:

```
sudo fdisk /dev/sdb
```

The **/dev/sdb** storage disk is open:

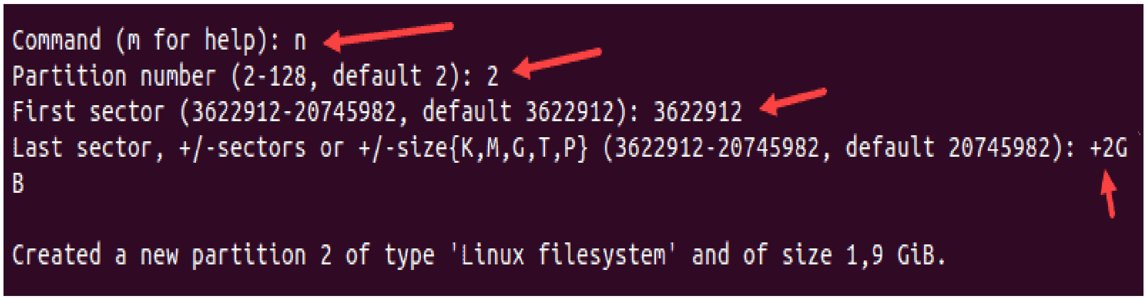


```
nevena@nevena-VirtualBox:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
```

Step 3: Create a New Partition

1. Run the **n** command to create a new partition.
2. Select the partition number by typing the default number (2).
3. After that, you are asked for the starting and ending sector of your hard drive. It is best to type the default number in this section (3622912).
4. The last prompt is related to the size of the partition. You can choose to have several sectors or to set the size in megabytes or gigabytes. Type **+2GB** to set the size of the partition to 2GB.

A message appears confirming that the partition is created.

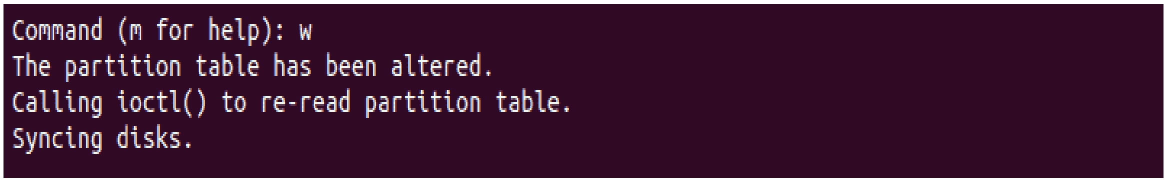


```
Command (m for help): n
Partition number (2-128, default 2): 2
First sector (3622912-20745982, default 3622912): 3622912
Last sector, +/-sectors or +/-size{K,M,G,T,P} (3622912-20745982, default 20745982): +2G
B

Created a new partition 2 of type 'Linux filesystem' and of size 1,9 GiB.
```

Step 4: Write on Disk

The system created the partition, but the changes are not written on the disk.

1. To write the changes on disk, run the **w** command:

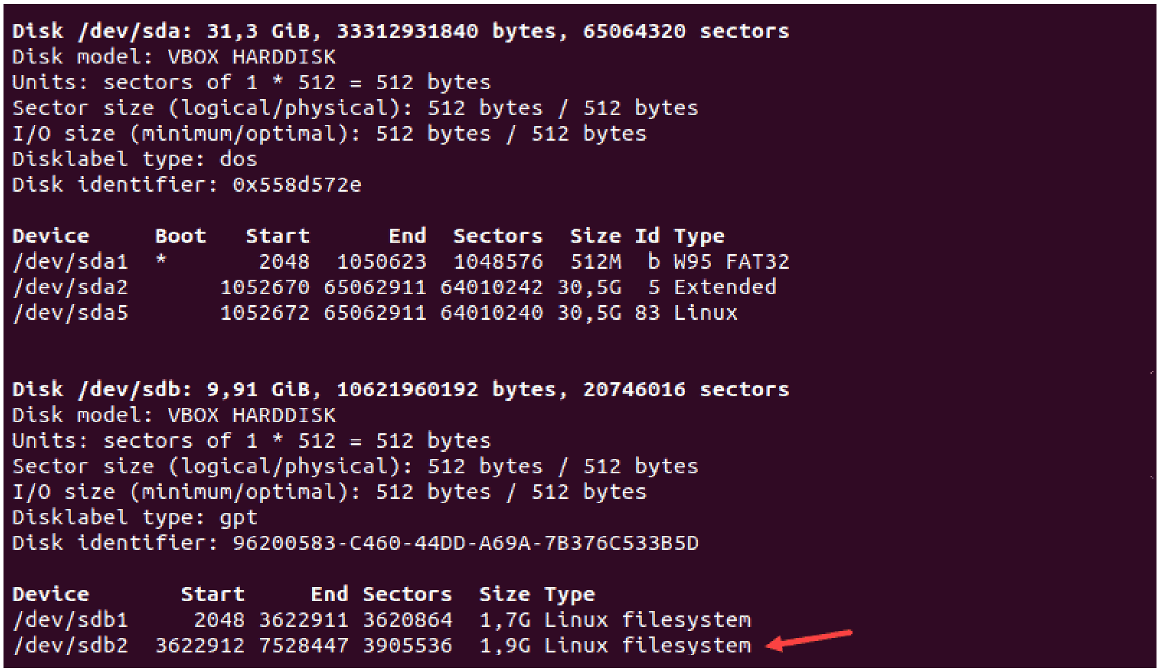


```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

2. Verify that the partition is created by running the following command:

```
sudo fdisk -l
```

As you can see, the partition **/dev/sdb2** has been created.



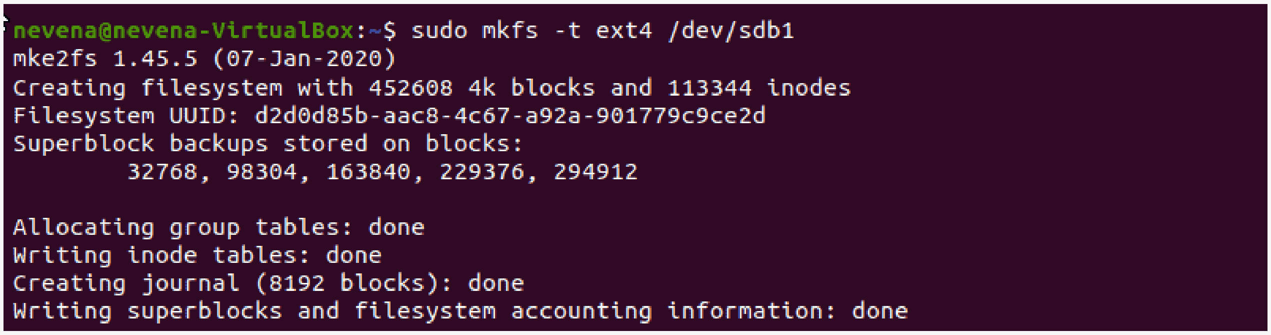
```
Disk /dev/sda: 31,3 GiB, 33312931840 bytes, 65064320 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x558d572e

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *        2048  1050623  1048576  512M  b W95 FAT32
/dev/sda2        1052670 65062911 64010242 30,5G  5 Extended
/dev/sda5        1052672 65062911 64010240 30,5G 83 Linux


Disk /dev/sdb: 9,91 GiB, 10621960192 bytes, 20746016 sectors
Disk model: VBOX HARDDISK
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: gpt
Disk identifier: 96200583-C460-44DD-A69A-7B376C533B5D

Device       Start     End Sectors  Size Type
/dev/sdb1     2048 3622911 3620864  1,7G Linux filesystem
/dev/sdb2  3622912 7528447 3905536  1,9G Linux filesystem
```

Format the Partition

Once a partition has been created with the **parted** of **fdisk** command, format it before using it.

Format the partition by running the following command:

sudo mkfs -t ext4 /dev/sdb1



```
nevena@nevena-VirtualBox:~$ sudo mkfs -t ext4 /dev/sdb1
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 452608 4k blocks and 113344 inodes
Filesystem UUID: d2d0d85b-aac8-4c67-a92a-901779c9ce2d
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Mount the Partition

To begin interacting with the disk, create a **mount point** and **mount the partition** to it.

1. Create a mount point by running the following command:

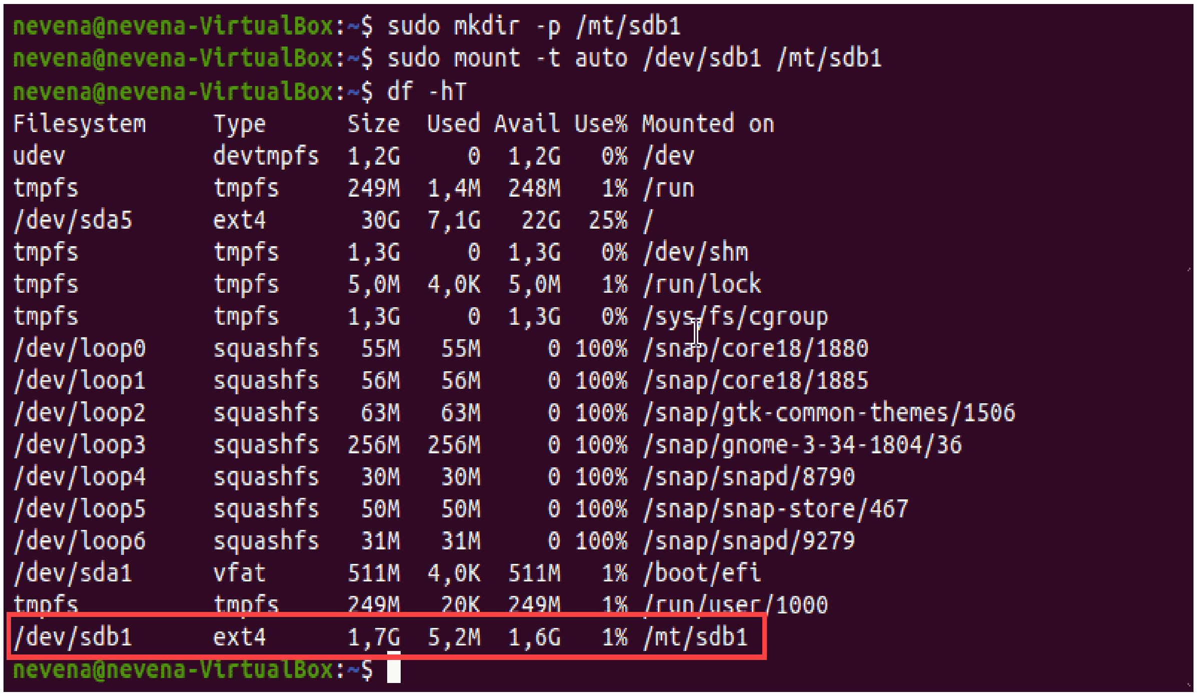
```
sudo mkdir -p /mt/sdb1
```

2. After that, mount the partition by entering:

```
sudo mount -t auto /dev/sbd1 /mt/sdb1
```

The terminal does not print out an output if the commands are executed successfully.

3. Verify if partition is mounted by using the **df hT** command:



```
nevena@nevena-VirtualBox:~$ sudo mkdir -p /mt/sdb1
nevena@nevena-VirtualBox:~$ sudo mount -t auto /dev/sdb1 /mt/sdb1
nevena@nevena-VirtualBox:~$ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
udev           devtmpfs  1,2G     0  1,2G   0% /dev
tmpfs          tmpfs     249M  1,4M  248M   1% /run
/dev/sda5      ext4       30G  7,1G   22G  25% /
tmpfs          tmpfs     1,3G     0  1,3G   0% /dev/shm
tmpfs          tmpfs     5,0M  4,0K  5,0M   1% /run/lock
tmpfs          tmpfs     1,3G     0  1,3G   0% /sys/fs/cgroup
/dev/loop0     squashfs   55M   55M     0 100% /snap/core18/1880
/dev/loop1     squashfs   56M   56M     0 100% /snap/core18/1885
/dev/loop2     squashfs   63M   63M     0 100% /snap/gtk-common-themes/1506
/dev/loop3     squashfs  256M  256M     0 100% /snap/gnome-3-34-1804/36
/dev/loop4     squashfs   30M   30M     0 100% /snap/snapd/8790
/dev/loop5     squashfs   50M   50M     0 100% /snap/snap-store/467
/dev/loop6     squashfs   31M   31M     0 100% /snap/snapd/9279
/dev/sda1      vfat      511M  4,0K  511M   1% /boot/efi
tmpfs          tmpfs     249M   20K  249M   1% /run/user/1000
/dev/sdb1      ext4      1,7G  5,2M  1,6G   1% /mt/sdb1
nevena@nevena-VirtualBox:~$ 
```

Configure LVM in Linux

Logical Volume Management (LVM) creates a layer of abstraction over physical storage, allowing you to create logical storage volumes. With LVM in place, you are not bothered with physical disk sizes because the hardware storage is hidden from the software so it can be resized and moved without stopping applications or unmounting file systems. You can think of LVM as dynamic partitions.

For example, if you are running out of disk space on your server, you can just add another disk and extend the logical volume on the fly.

Below are some advantages of using Logical volumes over using physical storage directly:

- **Resize storage pools:** You can extend the logical space as well as reduce it without reformatting the disks.
- **Flexible storage capacity:** You can add more space by adding more disks and adding them to the pool of physical storage, thus you have a flexible storage capacity.

LVM have 3 concepts

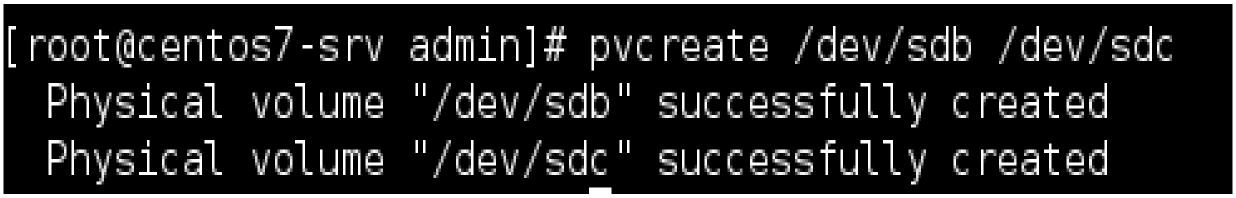
- **Physical Volume (PV): it** is a whole disk or a partition of a disk
- **Volume Group (VG):** corresponds to one or more PV
- **Logical Volume (LV):** represents a portion of a VG. A LV can only belong to one VG. It’s on a LV that we can create a file system.

1) Create Physical Volume

Physical volume is the actual storage device that will be used in the LVM configuration. It can be an entire disk, a partition on disk or a LUN on the SAN.

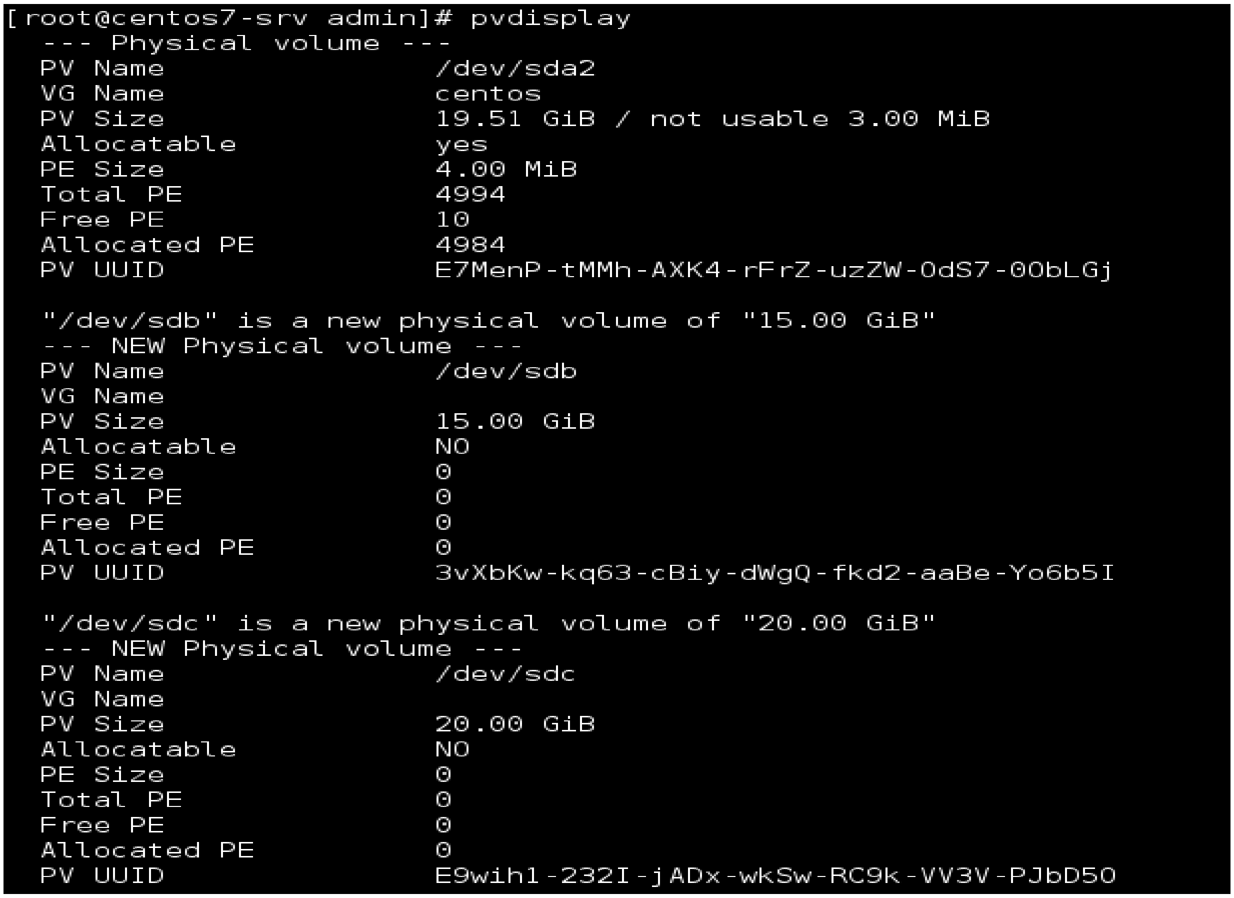
You can use pvcreate to create the physical volume. In this example I have added two disks `/dev/sdb` and `/dev/sdc` of 1 GB each. I will be using these for examples.

pvcreate command initialize these disks so that they can be a part in forming volume groups.



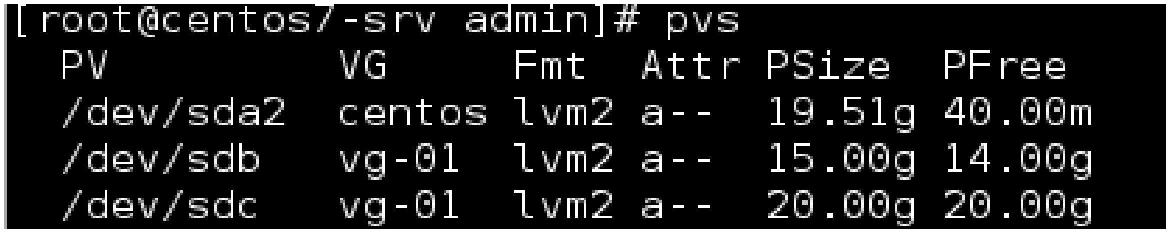
```
[root@centos7-srv admin]# pvcreate /dev/sdb /dev/sdc
  Physical volume "/dev/sdb" successfully created
  Physical volume "/dev/sdc" successfully created
```

Display the physical volumes:



```
[root@centos7-srv admin]# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sda2
  VG Name               centos
  PV Size               19.51 GiB / not usable 3.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              4994
  Free PE               10
  Allocated PE          4984
  PV UUID               E7MenP-tMMh-AXK4-rFrZ-uzZW-0dS7-0ObLGj

  "/dev/sdb" is a new physical volume of "15.00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb
  VG Name
  PV Size               15.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               3vXbKw-kq63-cBiy-dWgQ-fkd2-aaBe-Yo6b5I

  "/dev/sdc" is a new physical volume of "20.00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdc
  VG Name
  PV Size               20.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               E9wih1-232I-jADx-wkSw-RC9k-VV3V-PJbD5O
```

You can also use pvs command that will display the output in a configurable form.



```
[root@centos7-srv admin]# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sda2  centos lvm2 a--  19.51g 40.00m
  /dev/sdb   vg-01  lvm2 a--  15.00g 14.00g
  /dev/sdc   vg-01  lvm2 a--  20.00g 20.00g
```

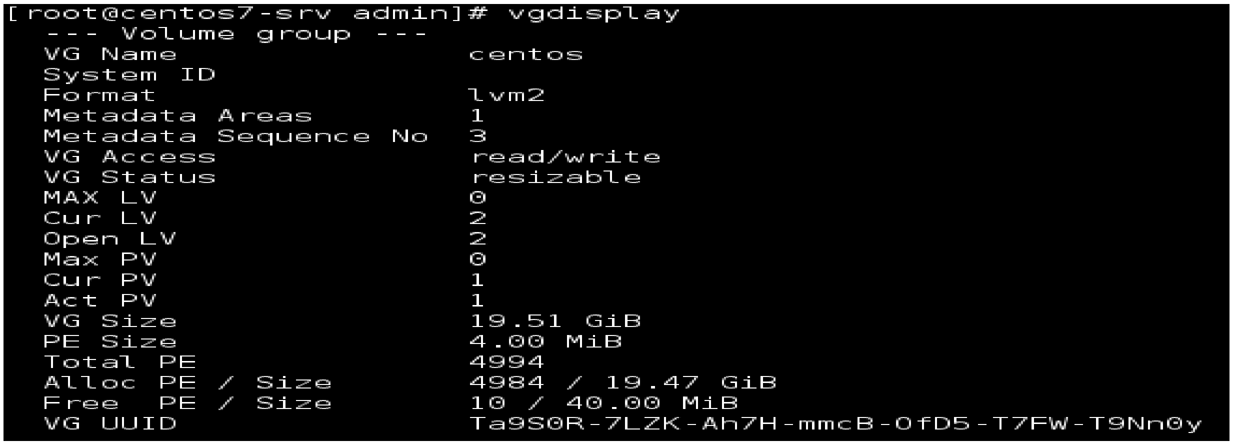
2) Create Volume Group

Physical volumes are combined into volume groups (VGs). It creates a pool of disk space out of which logical volumes can be allocated. The disk space available for allocation in Volume Group is divided into units of a fixed-size called extents. An extent is the smallest unit of storage that can be allocated. Within a physical volume, extents are referred to as physical extents.

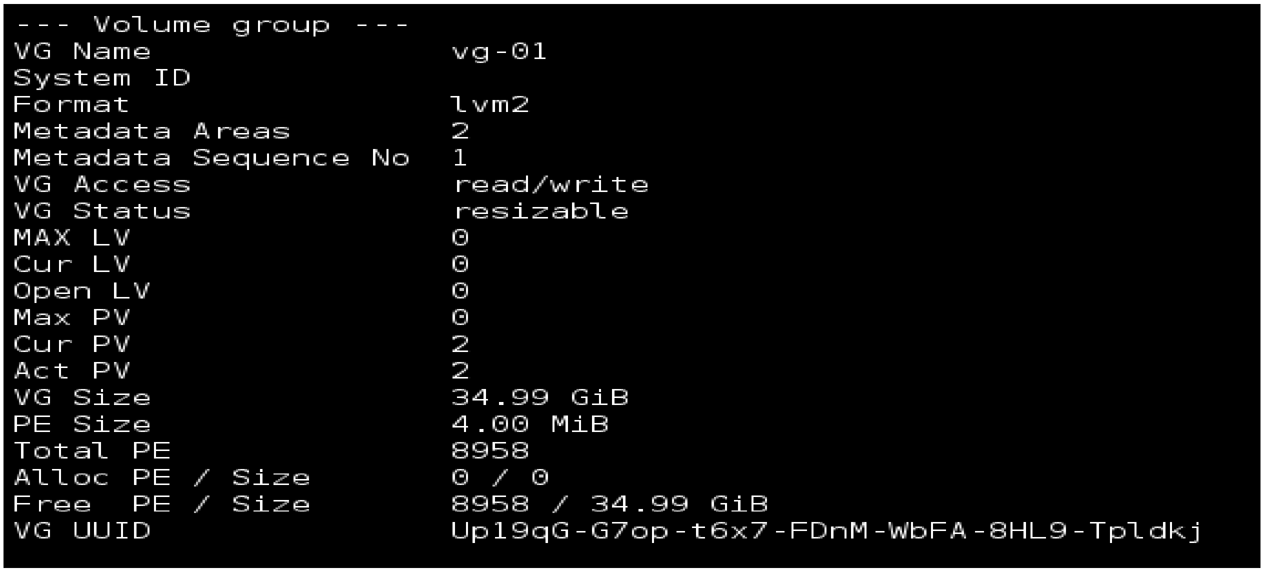
A logical volume is allocated into logical extents of the same size as the physical extents. The extent size is thus the same for all logical volumes in the volume group. The volume group maps the logical extents to physical extents.

```
# vgcreate vg-01 /dev/sdb /dev/sdc
 Volume group "vg-01" successfully created
```

Display information about VG's

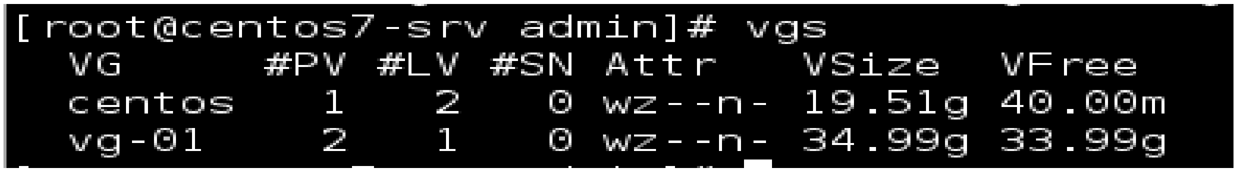


```
[root@centos7-srv admin]# vgdisplay
  --- Volume group ---
  VG Name               centos
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  3
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                2
  Open LV               2
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               19.51 GiB
  PE Size               4.00 MiB
  Total PE              4994
  Alloc PE / Size       4984 / 19.47 GiB
  Free  PE / Size       10 / 40.00 MiB
  VG UUID               Ta9S0R-7LZK-Ah7H-mmcB-OfD5-T7FW-T9Nn0y
```



```
--- Volume group ---
VG Name               vg-01
System ID
Format                lvm2
Metadata Areas        2
Metadata Sequence No  1
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                0
Open LV               0
Max PV                0
Cur PV                2
Act PV                2
VG Size               34.99 GiB
PE Size               4.00 MiB
Total PE              8958
Alloc PE / Size       0 / 0
Free  PE / Size       8958 / 34.99 GiB
VG UUID               Up19qG-G7op-t6x7-FDnM-WbFA-8HL9-Tpldkj
```

You can also use vgs command that will display the output in a configurable form.



```
[root@centos7-srv admin]# vgs
  VG      #PV #LV #SN Attr   VSize  VFree
  centos    1   2   0 wz--n- 19.51g 40.00m
  vg-01     2   1   0 wz--n- 34.99g 33.99g
```

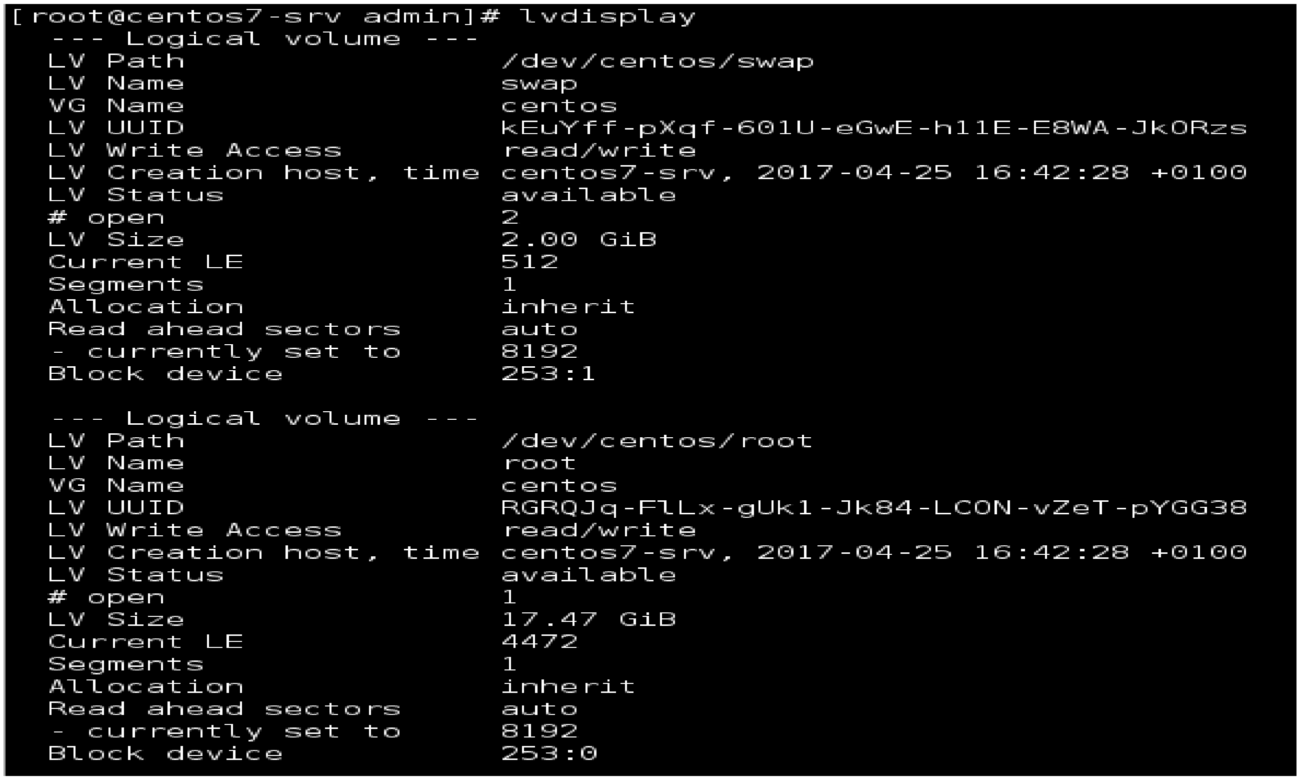
3) Create Logical Volume

A volume group is divided up into logical volumes. So if you have created vg-01 earlier then you can create logical volumes from that VG. The amount of space you want to allocate depends on your requirement. You might want to create LV of 200MB, 1GB etc.

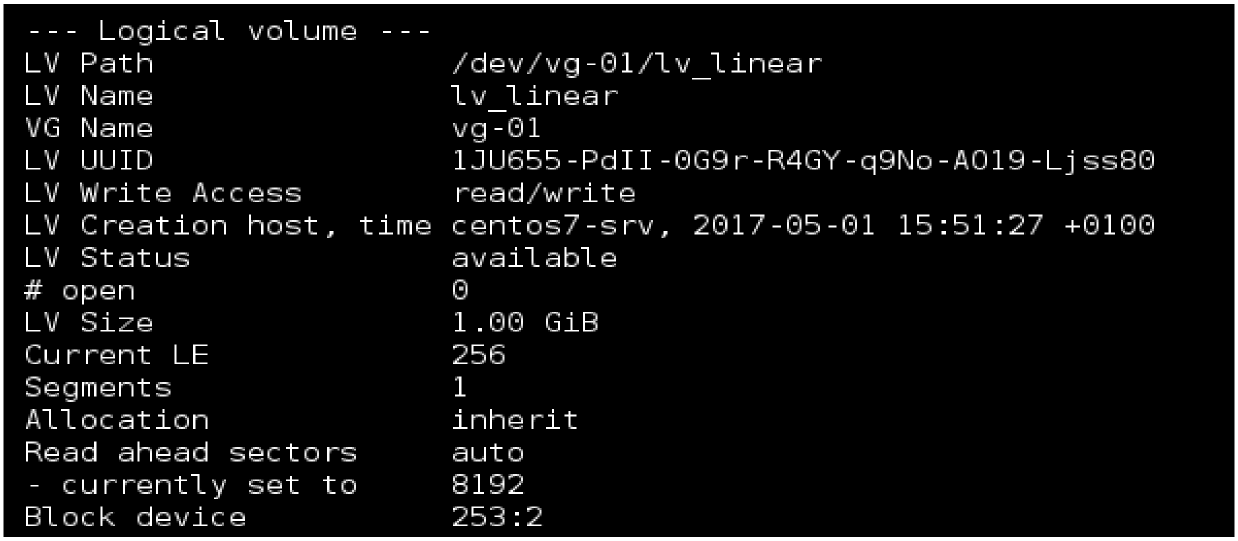
```
# lvcreate -L 1G -n lv_linear vg-01
```

```
Logical volume "lv_linear" created
```

Display Information about Logical Volumes.

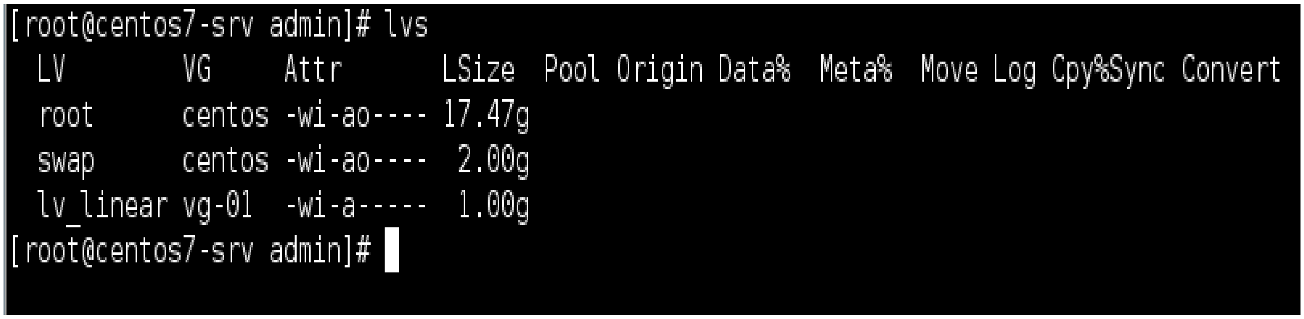


```
[root@centos7-srv admin]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/centos/swap
  LV Name                swap
  VG Name                centos
  LV UUID                kEuYff-pXqf-601U-eGwE-h11E-E8WA-JkORzs
  LV Write Access        read/write
  LV Creation host, time centos7-srv, 2017-04-25 16:42:28 +0100
  LV Status              available
  # open                 2
  LV Size                2.00 GiB
  Current LE             512
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:1

  --- Logical volume ---
  LV Path                /dev/centos/root
  LV Name                root
  VG Name                centos
  LV UUID                RGRQJq-FlLx-gUk1-Jk84-LCON-vZeT-pYGG38
  LV Write Access        read/write
  LV Creation host, time centos7-srv, 2017-04-25 16:42:28 +0100
  LV Status              available
  # open                 1
  LV Size                17.47 GiB
  Current LE             4472
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:0
```



```
 --- Logical volume ---
 LV Path                /dev/vg-01/lv_linear
 LV Name                lv_linear
 VG Name                vg-01
 LV UUID                1JU655-PdII-0G9r-R4GY-q9No-A019-Ljss80
 LV Write Access        read/write
 LV Creation host, time centos7-srv, 2017-05-01 15:51:27 +0100
 LV Status              available
 # open                 0
 LV Size                1.00 GiB
 Current LE             256
 Segments               1
 Allocation             inherit
 Read ahead sectors     auto
 - currently set to     8192
 Block device           253:2
```

You can also use lvs command that will display the output in a configurable form.

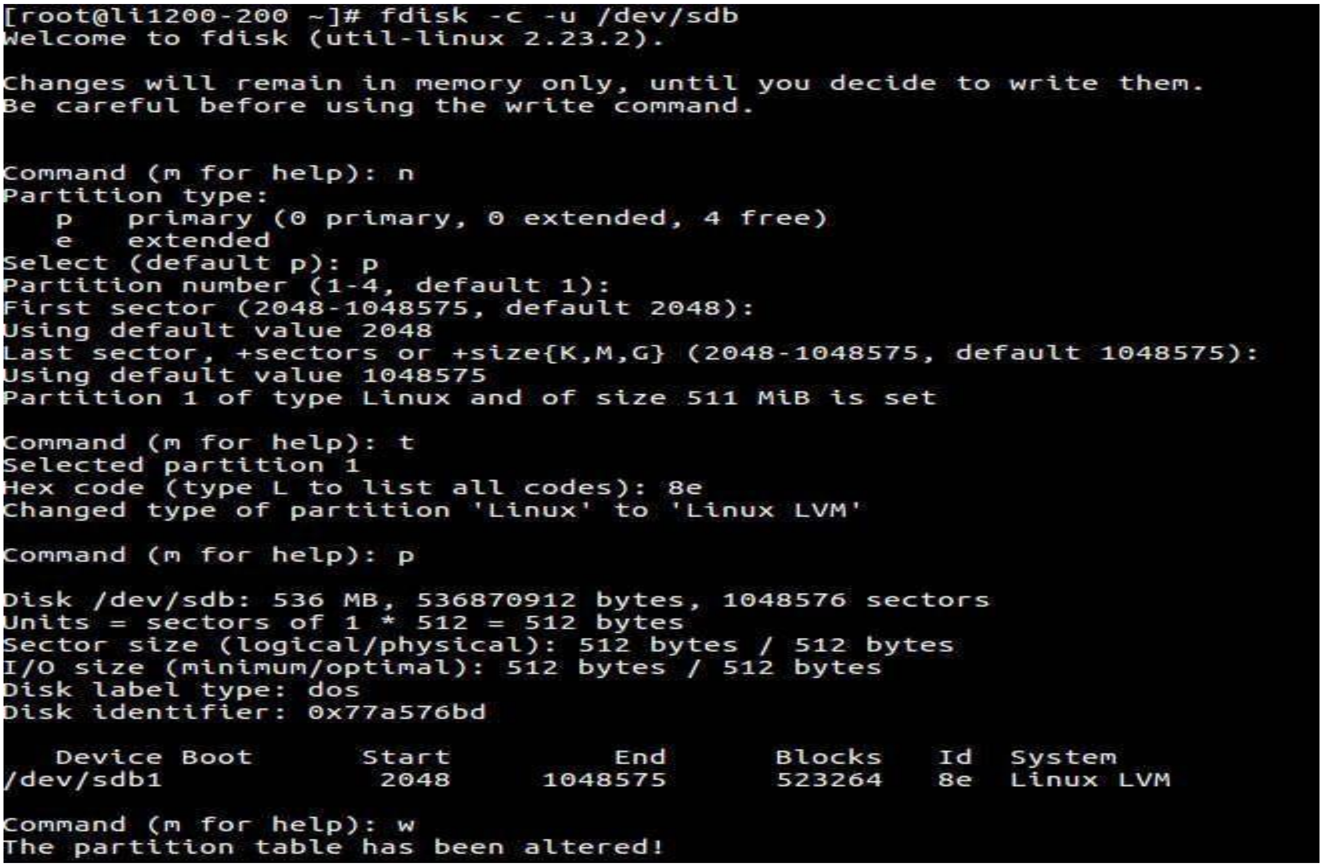


```
[root@centos7-srv admin]# lvs
  LV        VG     Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root      centos -wi-ao---- 17.47g
  swap      centos -wi-ao----  2.00g
  lv_linear vg-01  -wi-a-----  1.00g
[root@centos7-srv admin]#
```

Create LVM on a new hard disk

More physical volumes can be added to an existing volume group thus increasing its size. In general, using LVM, a partition can span more than one disk. The size of logical volumes can also be extended and reduced without any loss of data on that volume.

First using fdisk command make a partition and toggle that partition to LINUX LVM (8e) label.



```
[root@li1200-200 ~]# fdisk -c -u /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-1048575, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-1048575, default 1048575):
Using default value 1048575
Partition 1 of type Linux and of size 511 MiB is set

Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

Command (m for help): p

Disk /dev/sdb: 536 MB, 536870912 bytes, 1048576 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 label type: dos
Disk identifier: 0x77a576bd

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     1048575      523264   8e  Linux LVM

Command (m for help): w
The partition table has been altered!
```

Then create a physical volume using pvcreate command.

```
#pvcreate /dev/sdb1
```

Display the size of the physical volume

```
#pvdisplay /dev/sdb1
```

Create volume group whose name test

```
#vgcreate test /dev/sdb1
```

Display the details of volume group created

```
#vgdisplay test
```

Create logical volume of size 100 MB with name as data , /etc/test/data */

```
#lvcreate -L 100M -n data test
```

Display the information about logical volume

```
#lvdisplay /dev/test/data
```

Convert/format logical partition to ext3 filesystem

```
#mkfs –t ext4 /dev/test/data
```

Mount the volume to any directory

```
#mount /dev/test/data /mnt
```