**What Is Linux File System?**

Linux File System or any file system generally is a layer that is under the operating system that handles the positioning of your data on the storage; without it, the system cannot knows which file starts from where and ends where.

## Linux File System Types

When you try to [install Linux](https://likegeeks.com/how-to-install-linux/) you will see that Linux offers many file systems like these:

**Ext, Ext2, Ext3, Ext4, JFS, XFS, btrfs and swap**

So what are these file systems that Linux offers?

**Ext**: an old one and no longer used due to limitations.

**Ext2**: first Linux file system that allows two terabytes of data allowed.

**Ext3**: came from Ext2, but with upgrades and backward compatibility.

The only problem with it that the servers don’t use this kind of file system because this file system doesn’t support file recovery or disk snapshots.

**Ext4**: faster and allow large files with significant speed.

Ad by Valueimpression

It is a perfect option for SSD disks, and you notice when you try to install any Linux distro that this one is the default file system that Linux suggests.

**JFS**: old file system made by IBM. It works very well with small and big files, but it failed and files corrupted after long time use, reports say.

**XFS**: old file system and works slowly with small files.

**Btrfs:** made by Oracle. It is not stable as Ext in some distros, but you can say that it is a replacement for it if you have to. It has excellent performance.

You may notice From the comparison above that **Ext4** is the **best Linux File System.**

## ****Top Level Explanation****

Now you know what Linux file system is and its types. So what is inside that filesystem, I mean from the top level.

You may come from Windows, and Windows has partitions like C:\ and D:\, you can install Windows on any of these partitions, usually C:\.

What about the Linux File System Structure?

## Well, if you navigate to the root partition, which is / you’ll see the Linux File Linux File System Directories

**/bin**: Where Linux core commands reside like ls, mv.

**/boot**: Where boot loader and boot files are located.

**/dev**: Where all physical drives are mounted like USBs DVDs.

**/etc**: Contains configurations for the installed packages.

**/home**: Where every user will have a personal folder to put his folders with his name like /home/likegeeks.

**/lib**: Where the libraries of the installed packages located since libraries shared among all packages,

unlike Windows, you may find duplicates in different folders.

**/media**: Here are the external devices like DVDs and USB sticks that are mounted, and you can access their files from here.

**/mnt**: Where you mount other things Network locations and some distros, you may find your mounted USB or DVD.

**/opt**: Some optional packages are located here and managed by the package manager.

**/proc**: Because everything on Linux is a file, this folder for processes running on the system,

and you can access them and see much info about the current processes.

**/root**: The home folder for the root user.

**/sbin**: Like /bin, but binaries here are for root user only.

**/tmp**: Contains the temporary files.

**/usr**: Where the utilities and files shared between [users on Linux](https://likegeeks.com/listing-users-in-linux/).

**/var:** Contains system logs and other variable data.

Now you have a good idea about what the Linux file system is.

Choosing the right file system can lead you to the best performance, so choose the best.

**2 Run Levels in Linux**

* Last Updated : 16 Apr, 2019

Prerequisite : [Linux Booting](https://www.geeksforgeeks.org/what-happens-when-we-turn-on-computer/)  
A run level is a state of init and the whole system that defines what system services are operating. Run levels are identified by numbers. Some system administrators use run levels to define which subsystems are working, e.g., whether X is running, whether the network is operational, and so on.

* Whenever a LINUX system boots, firstly the **init** process is started which is actually responsible for running other start scripts which mainly involves initialization of you hardware, bringing up the network, starting the graphical interface.
* Now, the **init** first finds the default **runlevel** of the system so that it could run the start scripts corresponding to the default run level.
* A **runlevel** can simply be thought of as the state your system enters like if a system is in a single-user mode it will have a **runlevel 1** while if the system is in a multi-user mode it will have a **runlevel 5**.
* A **runlevel** in other words can be defined as a **preset single digit integer** for defining the operating state of your LINUX or UNIX-based operating system. Each runlevel designates a different system configuration and allows access to different combination of **processes**.

The important thing to note here is that there are differences in the runlevels according to the operating system. The standard **LINUX kernel** supports these seven different runlevels :

* 0 – System halt *i.e* the system can be safely powered off with no activity.
* 1 – Single user mode.
* 2 – Multiple user mode with no NFS(network file system).
* 3 – Multiple user mode under the command line interface and not under the graphical user interface.
* 4 – User-definable.
* 5 – Multiple user mode under GUI (graphical user interface) and this is the standard runlevel for most of the LINUX based systems.
* 6 – Reboot which is used to restart the system.

By default most of the LINUX based system boots to runlevel 3 or runlevel 5.  
In addition to the standard runlevels, users can modify the preset runlevels or even create new ones according to the requirement. Runlevels 2 and 4 are used for user defined runlevels and runlevel 0 and 6 are used for halting and rebooting the system.

Obviously the start scripts for each run level will be different performing different tasks. These start scripts corresponding to each run level can be found in special files present under **rc sub directories**.  
At **/etc/rc.d**directory there will be either a set of files named **rc.0, rc.1, rc.2, rc.3, rc.4, rc.5** and **rc.6,** or a set of directories named **rc0.d, rc1.d, rc2.d, rc3.d, rc4.d, rc5.d** and **rc6.d**.  
For example, run level 1 will have its start script either in file /etc/rc.d/rc.1 or any files in the directory /etc/rc.d/rc1.d.

**init** is the program responsible for altering the run level which can be called using **telinit** command.

For example, to change a runlevel from 3 to runlevel 5 which will actually allow the GUI to be started in multi-user mode the **telinit** command can be used as :

/\*using telinit to change

runlevel from 3 to 5\*/

**changing run level**

**telinit 5**

**NOTE :** The changing of runlevels is a task for the super user and not the normal user that’s why it is necessary to be logged in as super user for the successful execution of the above telinit command or you can use **sudo** command as :

// using sudo to execute telinit

**sudo telinit 5**

The default runlevel for a system is specified in **/etc/initab** file which will have an entry **id : 5 : initdefault** if the default runlevel is set to 5 or will have an entry **id : 3 : initdefault** if the default runlevel is set to 3.

**Need for changing the runlevel**

* There can be a situation when you may find trouble in **logging in** in case you don’t remember the password or because of the corrupted **/etc/passwd** file (have all the user names and passwords), in this case the problem can be solved by booting into a single user mode *i.e* runlevel 1.
* You can easily halt the system by changing the runlevel to 0 by using **telinit 0**.

3 Why is swap needed?

* For every operating system, there is a dedicated amount of RAM available that makes the processing of a program possible. However, the amount of this RAM is limited which is why RAM cannot hold a bulk of data in it. Therefore, there should be a backup option available which can support RAM whenever it runs out of memory.
* This concept holds for the Windows operating system as well as for Linux. In Windows OS, whenever RAM has an insufficient amount of memory to hold a process, it borrows some amount of memory from the secondary storage. This borrowed memory is known as Virtual Memory. Similarly, whenever RAM runs out of memory in Linux, it borrows some memory from the secondary storage to store its inactive content.
* In this way, the RAM finds sufficient space to hold a new process within it. Here, the borrowed space from the hard disk is called Swap Memory. In this article, we will try to learn the concept of swap memory in detail.

There are several reasons why you would need swap.

* If your system has RAM less than 1 GB, you must use swap as most applications would exhaust the RAM soon.
* If your system uses resource heavy applications like video editors, it would be a good idea to use some swap space as your RAM may be exhausted here.
* If you use hibernation, then you must add swap because the content of the RAM will be written to the swap partition. This also means that the swap size should be at least the size of RAM.
* Avoid strange events like a program going nuts and eating RAM.

### Do you need swap if you have lots of RAM?

This is a good question indeed. If you have 32GB or 64 GB of RAM, chances are that your system would perhaps never use the entire RAM and hence it would never use the swap partition.

But will you take the chance? I am guessing if your system has 32GB of RAM, it should also be having a hard disk of 100s of GB. Allocating a couple of GB of swap won’t hurt. It will provide an extra layer of ‘stability’ if a faulty program starts misusing RAM.

### Can you use Linux without swap?

Yes, you can, especially if your system has plenty of RAM. But as explained in the previous section, a little bit of swap is always advisable.

### Types of Swap Memory:

Typically there are two different types of swap memory which are mentioned below:

* **Swap Partition-**This is the default type of swap memory which is in fact, a hard drive partition that is dedicated to swapping.
* **Swap File-**This is a self-created type of swap memory. Whenever there is no sufficient amount of space left in the hard drive to create a swap partition, a swap file is manually created for swapping the inactive contents of RAM into it.

### What should be the Ideal Frequency of Swapping?

Linux allows us to set the frequency of swapping on our own i.e. how frequently the process of swapping should take place. You can set the value of swapping between 0 and 100 depending upon your requirements. A low-frequency value of swapping means that the process of swapping will take place very rarely only when it is needed whereas a high-frequency value of swapping means that the swapping process will occur quite often. However, the default and recommended value of swapping frequency is 60.

## 4 Using netstat to list open ports

Type the following netstat commandsudo netstat -tulpn | grep LISTEN

Linux Notes (Deepak)

## 1. What is Unix ?

The Unix operating system is a set of programs that act as a link between the computer and the user.

The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the **operating system** or the **kernel**.

Users communicate with the kernel through a program known as the **shell**. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

### System Shutdown

The most consistent way to shut down a Unix system properly via the command line is to use one of the following commands −

|  |  |
| --- | --- |
| Sr.No. | Command & Description |
| 1 | **halt**  Brings the system down immediately |
| 2 | **init 0**  Powers off the system using predefined scripts to synchronize and clean up the system prior to shutting down |
| 3 | **init 6**  Reboots the system by shutting it down completely and then restarting it |
| 4 | **poweroff**  Shuts down the system by powering off |
| 5 | **reboot**  Reboots the system |
| 6 | **shutdown**  Shuts down the system |

## 2. FileSystem ?

In Unix, there are three basic types of files −

* **Ordinary Files** − An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
* **Directories** − Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
* **Special Files** − Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

## Metacharacters

Metacharacters have a special meaning in Unix. For example, **\*** and **?** are metacharacters. We use **\*** to match 0 or more characters, a question mark (**?**) matches with a single character.

For Example −

$ls ch\*.doc

Displays all the files, the names of which start with **ch** and end with **.doc** −

ch01-1.doc ch010.doc ch02.doc ch03-2.doc

ch04-1.doc ch040.doc ch05.doc ch06-2.doc

ch01-2.doc ch02-1.doc c

Here, **\*** works as meta character which matches with any character. If you want to display all the files ending with just **.doc**, then you can use the following command −

$ls \*.doc

## Standard Unix Streams

Under normal circumstances, every Unix program has three streams (files) opened for it when it starts up −

* **stdin** − This is referred to as the *standard input* and the associated file descriptor is 0. This is also represented as STDIN. The Unix program will read the default input from STDIN.
* **Stdout** − This is referred to as the *standard output* and the associated file descriptor is 1. This is also represented as STDOUT. The Unix program will write the default output at STDOUT
* **stderr** − This is referred to as the *standard error* and the associated file descriptor is 2. This is also represented as STDERR. The Unix program will write all the error messages at STDERR.

## 3. File Permission ?

* **Owner permissions** − The owner's permissions determine what actions the owner of the file can perform on the file.
* **Group permissions** − The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
* **Other (world) permissions** − The permissions for others indicate what action all other users can perform on the file.

## 4. Pipes & Filters ?

You can connect two commands together so that the output from one program becomes the input of the next program. Two or more commands connected in this way form a pipe.

To make a pipe, put a vertical bar (**|**) on the command line between two commands.

When a program takes its input from another program, it performs some operation on that input, and writes the result to the standard output. It is referred to as a ***filter***.

## The grep Command

The grep command searches a file or files for lines that have a certain pattern. The syntax is −

$grep pattern file(s)

The name **"grep"** comes from the ed (a Unix line editor) command **g/re/p** which means “globally search for a regular expression and print all lines containing it”.

$ls -l | grep "Aug"

-rw-rw-rw- 1 john doc 11008 Aug 6 14:10 ch02

-rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07

-rw-rw-r-- 1 john doc 2488 Aug 15 10:51 intro

-rw-rw-r-- 1 carol doc 1605 Aug 23 07:35 macros

$

There are various options which you can use along with the **grep** command −

|  |  |
| --- | --- |
| Sr.No. | Option & Description |
| 1 | **-v**  Prints all lines that do not match pattern. |
| 2 | **-n**  Prints the matched line and its line number. |
| 3 | **-l**  Prints only the names of files with matching lines (letter "l") |
| 4 | **-c**  Prints only the count of matching lines. |
| 5 | **-i**  Matches either upper or lowercase. |

The sort Command

The **sort** command arranges lines of text alphabetically or numerically. The following example sorts the lines in the food file −

$sort food

Afghani Cuisine

Bangkok Wok

Big Apple Deli

Isle of Java

Mandalay

Sushi and Sashimi

Sweet Tooth

Tio Pepe's Peppers

$

The **sort** command arranges lines of text alphabetically by default. There are many options that control the sorting −

|  |  |
| --- | --- |
| Sr.No. | Description |
| 1 | **-n**  Sorts numerically (example: 10 will sort after 2), ignores blanks and tabs. |
| 2 | **-r**  Reverses the order of sort. |
| 3 | **-f**  Sorts upper and lowercase together. |
| 4 | **+x**  Ignores first **x** fields when sorting. |

More than two commands may be linked up into a pipe. Taking a previous pipe example using **grep**, we can further sort the files modified in August by the order of size.

The following pipe consists of the commands **ls**, **grep**, and **sort** −

$ls -l | grep "Aug" | sort +4n

-rw-rw-r-- 1 carol doc 1605 Aug 23 07:35 macros

-rw-rw-r-- 1 john doc 2488 Aug 15 10:51 intro

-rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07

-rw-rw-rw- 1 john doc 11008 Aug 6 14:10 ch02

$

This pipe sorts all files in your directory modified in August by the order of size, and prints them on the terminal screen. The sort option +4n skips four fields (fields are separated by blanks) then sorts the lines in numeric order.

## The pg and more Commands

A long output can normally be zipped by you on the screen, but if you run text through more or use the **pg** command as a filter; the display stops once the screen is full of text.

Let's assume that you have a long directory listing. To make it easier to read the sorted listing, pipe the output through **more** as follows −

$ls -l | grep "Aug" | sort +4n | more

-rw-rw-r-- 1 carol doc 1605 Aug 23 07:35 macros

-rw-rw-r-- 1 john doc 2488 Aug 15 10:51 intro

-rw-rw-rw- 1 john doc 8515 Aug 6 15:30 ch07

-rw-rw-r-- 1 john doc 14827 Aug 9 12:40 ch03

.

.

.

-rw-rw-rw- 1 john doc 16867 Aug 6 15:56 ch05

--More--(74%)

The screen will fill up once the screen is full of text consisting of lines sorted by the order of the file size. At the bottom of the screen is the **more** prompt, where you can type a command to move through the sorted text.

Once you're done with this screen, you can use any of the commands listed in the discussion of the more program.

## 5. Regular Expression ?

A regular expression is a string that can be used to describe several sequences of characters. Regular expressions are used by several different Unix commands, including **ed**, **sed**, **awk**, **grep**, and to a more limited extent, **vi**.

Here **SED** stands for **s**tream **ed**itor. This stream-oriented editor was created exclusively for executing scripts. Thus, all the input you feed into it passes through and goes to STDOUT and it does not change the input file.

* SED is a powerful text stream editor. Can do insertion, deletion, search and replace(substitution).
* SED command in unix supports regular expression which allows it perform complex pattern matching.

**Syntax ::**

sed OPTIONS... [SCRIPT] [INPUTFILE...]

1. **Replacing or substituting string :** Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file.

sed 's/unix/linux/' geekfile.txt

2. **Replacing the nth occurrence of a pattern in a line :** Use the /1, /2 etc flags to replace the first, second occurrence of a pattern in a line. The below command replaces the second occurrence of the word “unix” with “linux” in a line.

sed 's/unix/linux/2' geekfile.txt

3. **Replacing all the occurrence of the pattern in a line :** The substitute flag /g (global replacement) specifies the sed command to replace all the occurrences of the string in the line.

sed 's/unix/linux/g' geekfile.txt

4. **Replacing from nth occurrence to all occurrences in a line :** Use the combination of /1, /2 etc and /g to replace all the patterns from the nth occurrence of a pattern in a line. The following sed command replaces the third, fourth, fifth… “unix” word with “linux” word in a line.

$sed 's/unix/linux/3g' geekfile.txt

5. **Parenthesize first character of each word :** This sed example prints the first character of every word in parenthesis.

echo "Welcome To The Geek Stuff" | sed 's/\(\b[A-Z]\)/\(\1\)/g'

**Output:**

(W)elcome (T)o (T)he (G)eek (S)tuff

## 6. User Administration ?

## Managing Users and Groups

There are four main user administration files −

* **/etc/passwd** − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
* **/etc/shadow** − Holds the encrypted password of the corresponding account. Not all the systems support this file.
* **/etc/group** − This file contains the group information for each account.
* **/etc/gshadow** − This file contains secure group account information.