[https://www.tecmint.com/free-online-linux-learning-guide-for-beginners/](https://www.tecmint.com/free-online-linux-learning-guide-for-beginners/" \t "_top)

<https://bash.cyberciti.biz/guide/Main_Page>

# What is Linux?

Linux is a free open-source operating system based on Unix. **Linus Torvalds** originally created Linux with the assistance of developers from around the world. Linux is:

* Free
* Unix like
* Open Source
* Network Operating System

Strictly speaking, Linux is a kernel. A kernel provides access to the computer hardware and control access to resources such as:

* Files and data.
* Running programs.
* Loading program into memory.
* Networks.
* Security and firewall.
* Other resources etc.

The kernel decides who will use a resource, for how long and when. You can download the Linux kernel from the official web site.

## History of Linux

In 1991, **Linus Torvalds** was studying UNIX at university (**Helsinki University**), where he was using a special educational, experimental purpose operating system called MINIX (a small version of UNIX to be utilized in the academic environment). However, **MINIX** had its limitations, and Linus felt he could create something better. Therefore, he developed his version of MINIX, known as Linux. Linux was Open Source right from the start.

Linux is a kernel developed by Linus. The kernel was bundled with system utilities and libraries from the GNU project to create a usable operating system. Sometimes people refer to Linux as GNU/Linux because it has system utilities and libraries from the GNU project. **Linus Torvalds** is credited for creating the Linux Kernel, not the entire Linux operating system.

**Linux distribution** = Linux kernel + GNU system utilities and libraries + Installation scripts + Management utilities and more.

**Note**: Linux is written in C & C++ Languages

## Linux usage in Everyday life

You can use [Linux](https://bash.cyberciti.biz/guide/Linux)as a [server operating system](https://bash.cyberciti.biz/guide/Server_operating_system)or as a stand-alone [operating system](https://bash.cyberciti.biz/guide/Operating_system)on your PC. As a server operating system it provides different services/network resources to a client. A [server operating system](https://bash.cyberciti.biz/guide/Server_operating_system)must be:

* Stable
* Robust
* Secure
* High performance

## Basic components of Linux?

Just like any other typical operating system, Linux has all of these components:

* **Kernel**: It is the core component of Linux; it acts as an interface between software and hardware.
* **Shell:** It acts as an interface between the user and the Kernel.
* **GUI:** It stands for Graphic User Interface, which is another way for the user to interact with the system. But it is unlike images, buttons, text boxes for interaction.
* **System Library:** are special functions using by application program or system utilies to access the kernel’s features.
* **System Utilities:** These are the software functions that allow users to manage the computer.
* **Application Programs**: Set of functions designed to perform a set of tasks.

## Features of Linux

* **Multi-user:** Allow multiple users to login and use the resources at the same time.
* **Multi-processing:** allow to performs multiple processes at the same time.
* **Multitasking:** capable of running multiple applications and processes at the same time.
* **Portability:** we can install on all hardware architecture.
* **Reliability**: Large servers have been successfully being running without a single second of down time.
* **Security**: Inbuilt firewalls (iptables) and selinux.
* **Open-source**: Free software along with the source code and documentation.
* **Hierarchical File system** : Linux provides a standard file structure in which system files/user files are arranged.
* **Shell**: Linux provides a special interperter program which can be used to execute commands of the operating system. It can be used to do various types of operatins, call application program etc.

## Advantages of Open Source

Open-source allows you to distribute your software, including source codes freely to anyone who is interested. People would then be able to add features and even debug and correct errors that are in the source code. They can even make it run better and then redistribute these enhanced source code freely again. This eventually benefits everyone in the community.

## Dis-advantages of Open-source

* Difficulty to use.
* Compatibility issue.
* Liabilities and warranties
* Hidden cost.

## What is the Importance of the GNU project?

This so-called free software movement allows several advantages, such as the freedom to run programs for any purpose and freedom to study and modify a program to your needs. It also allows you to redistribute copies of software to other people, as well as the freedom to improve software and have it released for the public.

## What are the difference between RHEL6 and RHEL7?

<https://foxutech.com/difference-between-rhel-6-and-rhel-7/>

<https://codingcompiler.com/difference-between-rhel-6-vs-rhel-7/>

## Difference between Linux and Windows

|  |  |  |
| --- | --- | --- |
| **Sr no** | **Linux** | **Windows** |
| 1 | Open-source development model, so the programmer can redesign the OS. | It is not an open-source and hence cannot be redesigned by the programmer. |
| 2 | Linux is robust and very much secure from virus. | Windows gets affected by virus very easily. |
| 3 | The Linux servers has inbuilt security features like Iptables and Selinux. | Security is the main issue which has made windows to think to survive. |
| 4 | In Linux File System Support: ext4, XFS | In Windows File System Support: FAT & NTFS. |
| 5 | Linux use: Lilo (Linux Loader) and Grub (Grand unified bootloader) | Windows: NTLDR  (New technology Loader) |
| 6 | In Linux software file extension is: rpm | In window software file extension is: .exe |
| 7 | It costs less to design and implement Linux Network | It costs more to implement Windows Network administration, when compared to Linux. |
| 8. | There are 25000+ developers behind Linux for open-source deployment. | Compared to Linux, windows are developed by thousands of people. |

## What is a Variable?

* A variable is a location for storing a value which can be a filename, text, number or any other data. The value thus stored can be displayed, deleted, edited and re-saved.
* Variables play an important role in computer programming because they enable programmers to write flexible programs

## What are the environmental variables or Dynamic variables?

* A dynamic variable is a variable whose address is determined when the program is run. It’s exits in every operating system and their types may vary.
* This can be created, edited, saved and deleted and they also give the information about the system behaviour.
* Environment variables can change the way a software/programs behave.

## What are symbolic links?

Symbolic links act similarly to shortcuts in Windows. Its links point to programs, files or directories. It also allows you instant access to it without having to go directly to the entire pathname.

## What are hard links?

Hard links point directly to the physical file on disk, and not on the pathname. This means that if you rename or move the original file, the link will not break since the link is for the file itself, not the path where the file is located.

## Difference between hard links and soft links?

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Soft Links** | **Hard Links** |
| 1. | A symbolic or soft link is an actual link to the original file | Whereas a hard link is a mirror copy of the original file. |
| 2. | Soft links have different inodes numbers. | Hard links have the same inodes numbers. |
| 3. | If you delete the original file, the soft link has no value, because it points to a non-existent file. | But in the case of hard links, it is entirely the opposite. Even if you delete the original file, the hard link will still have the data of the original file. Because a hard link acts as a mirror copy of the original file. |
| 4. | A soft link can link to a directory. | You cannot create hard links for a directory. |
| 5. |  |  |

## What is the maximum length for a filename under Linux?

Any filename can have a maximum of 255 characters. This limit does not include the path name, so therefore the entire pathname and filename could well exceed 255 characters.

# What is kernel?

The kernel is the heart of the Linux operating system. It manages the resources of Linux such as:

* File Management.
* Multitasking.
* Memory Management.
* I/O Management.
* Process Management.
* Device Management.
* Network support including IPv4 and IPv6.

The kernel acts as an intermediary between the computer hardware and various applications. If software requests hardware, then the kernel delivers the data between software and hardware.

## How to upgrade Kernel in Linux?

We should never upgrade Linux Kernel, always install the new kernel using rpm command because upgrading a kernel can make your Linux box in an unbootable state.

## What are kernel modules and how to load it manually?

## How to add & change the Kernel parameters?

To Set the kernel parameters in Linux, first edit the file ‘**/etc/sysctl.conf**’ after making the changes save the file and run the command ‘**sysctl -p**’, this command will make the changes permanently without rebooting the machine.

## What is the command to view all the kernel parameters?

#sysctl -a

# What is SHELL?

* The shell is a user program or it is an environment provided for user interaction.
* It is a command language interpreter that executes commands read from the standard input device such as keyboard or from a file.
* The shell is not part of the system kernel, but uses the system kernel to execute programs, create files etc.

**Several shells are available for** [**Linux**](https://bash.cyberciti.biz/guide/Linux) **including:**

csh, ksh, bash, Bourne. The most commonly used and advanced shell used today is "Bash”.

**Please note that each shell does the same job, but each understands different command syntax and provides different built-in functions.**

How do I find out my current shell name?

cat /etc/shells, echo $SHELL, ps $$, ps -p $$

## Why shell scripting?

* Shell scripts can take input from a user or file and output them to the screen
* Whenever you find yourself doing the same task over and over again you should use shell scripting, i.e., repetitive task automation.

## Bash Scripts for Interview?

<https://linuxhint.com/bash_scripting_interview_questions/>

## What is Bash?

BASH is short for **Bourne Again Shell**. It was written by **Steve Bourne** as a replacement to the original Bourne Shell (represented by /bin/sh). It combines all the features from the original version of Bourne Shell, plus additional functions to make it easier and more convenient to use. It has since been adapted as the default shell for most systems running Linux.

## What is the use of a shebang line?

Shebang line at the top of each script determines the location of the engine which is to be used to execute the script.

## Explain the type of files in Linux?

<https://www.tecmint.com/explanation-of-everything-is-a-file-and-types-of-files-in-linux/>

<https://linuxconfig.org/identifying-file-types-in-linux>

## How do we create command aliases in a shell?

alias Aliasname="Command whose alias is to be created".

## File compression utilises in Linux?

[**https://www.unixmen.com/top-15-file-compression-utilities-linux/**](https://www.unixmen.com/top-15-file-compression-utilities-linux/)

## What is inode block?

This block contains the inode for every file of the file system along with all the file attributes except its name.

## What is Super block?

Super block contains all the information about the file system like the size of file system, block size used by its number of free data blocks and list of free inodes and data blocks.

## What is a boot block?

This block contains a small program called "Master Boot record"(MBR) which loads the kernel during system boot up.

## ****What are the 3 standard streams in Linux?****

0 - Standard Input 1 - Standard Output 2 - Standard Error

# Linux Boot Process?

RHEL6:

RHEL7: [BIOS(POST) => MBR=> GRUB2=>Kernel=> systemd=> User prompt]

### Step1. Power on:

* When you power on/ Restart your machine, then the power is supplied to SMPS **(Switch-mode power supply)**, which converts AC to DC. The DC power is supplied to all the devices connected to that machine such as Motherboard HDDs, CD/DVD-ROM, Mouse, keyboard etc. The most intelligent device in the computer is Processor (**CPU**), which supplied the power in sequence and power on devices.
* The first instruction is powered on BIOS (**Basic input/Output System)**. This is the first software program which is pre-installed firmware in a motherboard.
* BIOS will take care of two things. (a) Run **POST** (b) Selecting first boot device.
* POST (**Power of self-test**) verifies all the hardware and peripheral devices are working properly. Once POST completes, it flushes from the memory.
* Then BIOS searches for devices for bootable disk. Once BIOS detects the bootable device then it executes the MBR **(Master Boot Recorder)**.

### Step2: MBR (Master boot Recorder) (1’st stage of boot loader)

* Once the BIOS gives control back to CPU, it will try to load MBR of the first boot device (We will consider it as HDD). MBR is a small part of Hard Disk with just a size of 512 Bytes, This MBR stored in the first sector of HDD or end of HDD depending on manufacturer.
* MBR contain the information about boot loader location, and set the partitional table and load the second stage of boot loader into the memory.

**MBR locate on disk which have detail about.**

**Primary boot loader code (This is of 446 bytes):** This code provides boot loader information and location detail of actual boot loader code on the hard disk. This is helpful for CPU to load second stage of Boot loader.

**Partition table information (64 bytes):** MBR contains 64 bytes of data which stores partition table information such as what is the start and end of each partition, size of partitions, type of partitions etc.

**Magic number (2 bytes):** MBR ends with two bytes that are defined as the magic number (0xAA55). The magic number serves as a validation check of the MBR. If MBR gets corrupts the magic number is used to retrieve it.

Which will be equal to 512 bytes (446+64+2).

Once the MBR locate the second stage of boot loader (GRUB), it starts to load into the memory.

**&**

**Why we can create only up to 4 primary partitions?**

The reason because of a limitation of MBR. The MBR is only 512 bytes of size, it is needed to store the primary bootloader, and the partition table. Typically, the area reserved for partition table is only 64 bytes. And the partition table entry for one partition is 16 bytes. So, 16×4=64. The space is over. So, we can't create more than this.

### Step3: GRUB (Grand Unified Bootloader)

* Grub2 are mainly used boot loader in Linux. LILO is stand for **Linux loader** which is used in older Linux version. Now **Grub2** is default boot loader program in all latest version of Linux like Red Hat/Centos and also in Ubuntu.
* The GRUB 2 configuration file is located at /boot/grub2/grub.cfg (Do not edit this file directly).
* GRUB 2 menu-configuration settings are taken from /etc/default/grub when generating grub.cfg.
* If changes are made to any of these parameters, you need to run grub2-mkconfig to re-generate the /boot/grub2/grub.cfg file.

# grub2-mkconfig –o /boot/grub2/grub.cfg

* GRUB2 searches the compressed kernel image file also called as **vmlinuz** in the /**boot** directory.
* GRUB2 loads the vmlinuz kernel image file into memory and extracts the contents of the initramfs image file into a temporary, memory-based file system (tmpfs).
* The initial RAM disk (**initrd**) is an initial root file system that is mounted before the real root file system.
* GRUB boot menu (where the user can manually specify the boot parameters) to the user. GRUB loads the user-selected (or default) kernel into memory and passes control on to the kernel. If user do not select the OS, after a defined timeout GRUB will load the default kernel in the memory for starting it.

### Step4: Kernel

* Kernel loads the necessary driver modules from initrd image. Initrd is a temporary root file system that is mounted during system boot to support the two-state boot process.
* The initramfs contains kernel modules for all hardware that is required to boot, as well as the initial scripts required to proceed to the next stage of booting.

Difference between Initramfs and initrd:

(Initramfs is used by 2.6 kernels whereas initrd was used by older 2.4 (and earlier) kernels.)

* Once the real file system is mounted, the kernel starts the systemd process with a process ID 1 (PID 1) and in RHEL6 INIT is the first process.

INIT

* It is the first process that runs in the system with process id of 1.
* Also known as the parent process, responsible for starting all the other process.
* Init starts services to the corresponding run level on which our machine is booted.

Run level is a state of a machine that defines how a machine should be login, what services and scripts should run when a machine starts. Run level are defined from 0 – 6.

1. Run level 0 – shut down the system
2. Run level 1 – single mode
3. Run level 2 – multiuser mode without networking
4. Run level 3 – multiuser with text login screen
5. Run level 4- customized run level (not in use)
6. Run level 5 – run level 3 with graphical login
7. Run level 6 – Reboots the system.

* **/etc/init.d/rc** or **/etc/rc.d/rc** : It contains the total number of the script that will run.
* Init will wait until once scripts get completed and once all required services and script are executed user gets login prompt.

But from RHEL 7 init and run levels are replaced by systemd and targets respectively.

Systemd

* Same as init it is the first service started by the kernel, with process id 1.
* “**d**” in system d stands for a daemon.
* Daemons are the programs that run in the background performing various tasks.
* Configuration file: /etc/systemd.

Init had several limitations that were taken care by systemd.

Few of them are:

* In init, one script runs at a time, while in systemd multiple scripts run at a time parallel. Thus, increasing the performance.
* In init, an administrator has to take care of what scripts should run first. Init process does not know if the scripts defined in particular run level are dependent on another script, while systemd takes care of dependency.
* Systemd kills daemon properly before the machine shuts down.
* inittab is no longer used when using systemd.
* systemd uses ‘targets’ instead of run levels. By default, there are two main targets:

a) multi-user.target: analogous to runlevel 3

b) graphical.target: analogous to runlevel 5

### User Prompt

This is actually not part of booting process but thought of including it here for better understating. Once the Kernel get the control it starts multiple instances of "getty" which waits for console logins which spawn one's user shell process and gives you user prompt to login.

# Process Management

## What is a Process?

* A process is program (or command typed by user) to perform specific Job.
* In Linux when you start a process, it is given a unique number called a PID or process-id.
* PIDs start from 0 to 65535.
* PID 1 is always assigned to init process, which is the first process started at boot time. Type the following command to verify that init always has PID 1:

## Parent and Child Processes

* A parent process is a Linux process that has created one or more child processes.
* A process can fork a child i.e., create a child process.
* For example, if a user types the ls command at a shell prompt.
* The shell executes ls command.
* The Linux kernel will duplicate the shell's pages of memory and then execute the ls command.
* In UNIX, every process is created using fork and exec method. However, this model results in a waste of system resources.

Process States

Every [Linux](https://bash.cyberciti.biz/guide/Linux)process has its own life cycle such as creation, execution, termination, and removal. Every process has its own state that shows what is currently happening in the process. The status of the process which can be one of the following:

* D (uninterruptible sleep) - Process is sleeping and cannot be bring back until an event such as I/O occurred. For example, process foo is a process waiting for keyboard interrupt.
* R (running) - Process is running or executing.
* S (sleeping) - Process is not running and is waiting for an event or a [signal](https://bash.cyberciti.biz/guide/Signals).
* T (traced or stopped) - Process is stopped by [signals](https://bash.cyberciti.biz/guide/Shell_signal_values) such as [SIGINT](https://bash.cyberciti.biz/guide/SIGINT) or [SIGSTOP](https://bash.cyberciti.biz/guide/SIGSTOP).
* Z (zombie or defunct) - Processes marked <defunct> are dead processes (so-called "zombies") that remain because their parent has not destroyed them properly. These processes will be destroyed by init if the parent process exits.

## How do I view Process states?

ps -C processName -o pid=,cmd,stat

## What are daemons?

Daemons are services that provide several functions that may not be available under the base operating system. Its main task is to listen for service requests and at the same time to act on these requests. After the service is done, it is then disconnected and waits for further requests.

## What is Process id?

A PID is a process identification number on a Linux or Unix-like operating system. A PID is automatically assigned to each process when it is created. A process is nothing but a running instance of a program has a unique PID on a Unix-like system.

The easiest way to find out if the process is running is run **ps aux** command and **grep** process name.

## What is the Process status in Linux?

Five process states in Linux. They are

* **New/Ready:** A new process is created and ready to run.
* **Running:** The process is being executed.
* **Blocked/ Wait:** The process is waiting for input from the user.
* **Terminated/ Completed:** The process completed the execution or terminated by the operating system.
* **Zombie:** The process is deleted, but still the information regarding the process exists in the process table.

## Explain Process Management System Calls in Linux?

The process management system calls in Linux:

* **fork():** Used to create a new process.
* **exec():** Execute new process.
* **wait():** wait until process execution.
* **exit():** exit from the process.

System calls to get the Process id:

* **getpid():** to find the unique process id.
* **getppid():** to find the unique parent process id.

## What is the Zombie Process?

It is a process whose execution is completed but even the information exists in the process table. It occurs for the child process because the parent process needs to read the child process status. Once it is completed using the wait system call, then the zombie process is removed from the process table. This is known as the Zombie Process.

Or

These are the processes which have died but whose exit status is still not picked by the parent process. These processes even if not functional still have its process id entry in the process table.

## What is load average in Linux?

Load Average is defined as the average sum of the number of processes waiting in the run queue and number of processes currently executing over the period of 1,5 and 15 minutes. Using the ‘top’ and ‘uptime’ command we find the load average of a Linux server.

## How to check which ports are listening on my Linux Server?

Use the Command ‘**netstat --listen**’ and ‘**lsof -i**’

List the services that are enabled at a particular run level in Linux server?

With the help of command ‘**chkconfig –list | grep 5:on**’ we can list all the services that are enabled in run level5. For other run levels just replace 5 with the respective run level.

## How to enable a service at a particular run level?

We can enable a service using the Command ‘**chkconfig <Service-Name> on --level 3**’