

LINUX

What is Unix?

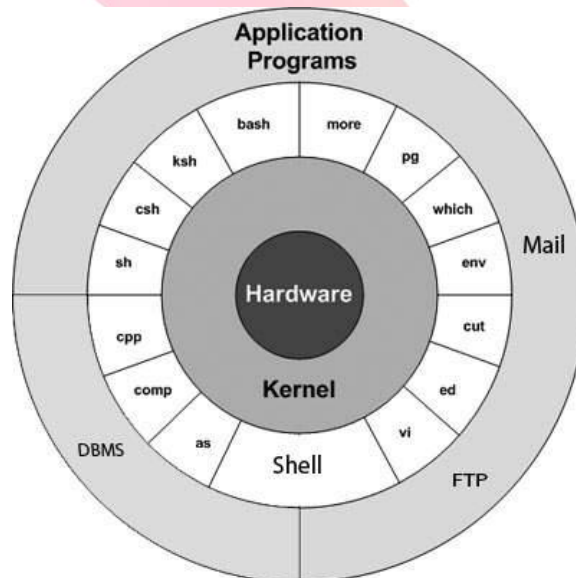
The Unix operating system is a set of programs that act as a link between the computer and the user. The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the **operating system** or the **kernel**.

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.

- Unix was originally developed in 1969 by a group of AT&T employees Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna at Bell Labs.
- There are various Unix variants available in the market. Solaris Unix, AIX, HP Unix and BSD are a few examples. Linux is also a flavor of Unix which is freely available.
- Several people can use a Unix computer at the same time; hence Unix is called a multiuser system.
- A user can also run multiple programs at the same time; hence Unix is a multitasking environment.

Unix Architecture




- Here is a basic block diagram of a Unix system –



The main concept that unites all the versions of Unix is the following four basics –

- **Kernel** – The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.

H.No: 208-1, 2nd Floor, Annapurna Block, Ameerpet, Hyderabad – 500038

 : www.srimanit.com  : <https://youtube.com/c/SrimanIT>  : https://twitter.com/sriman_it

 : <https://www.facebook.com/srimaniTech/>  : https://www.instagram.com/sriman_it/

 : +91-99850 14433  : +91-99850 24433

- **Shell** – The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.
- **Commands and Utilities** – There are various commands and utilities which you can make use of in your day to day activities. cp, mv, cat and grep, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.
- **Files and Directories** – All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the file system.

Linux Features

- **Multitasking capability:** Multiple users can access the same system resources like memory, hard disk, etc. But they have to use different terminals to operate.
- **Multitasking:** More than one function can be performed simultaneously by dividing the CPU time intelligently.
- **Portability:** Portability doesn't mean it is smaller in file size or can be carried in pen drives or memory cards. It means that it support different types of hardware.
- **Security:** It provides security in three ways namely authenticating (by assigning password and login ID), authorization (by assigning permission to read, write and execute) and encryption (converts file into an unreadable format).
- **Live CD/USB:** Almost all Linux distros provide live CD/USB so that users can run/try it without installing it.
- **Graphical User Interface (X Window system):** Linux is command line based OS but it can be converted to GUI based by installing packages.
- **Support's customized keyboard:** As it is used worldwide, hence supports different languages keyboards.
- **Application support:** It has its own software repository from where users can download and install many applications.
- **File System:** Provides hierarchical file system in which files and directories are arranged.
- **Open Source:** Linux code is freely available to all and is a community based development project.

Why use Linux

Linux is completely different from other operating systems in many ways.

- It is an open source OS which gives a great advantage to the programmers as they can design their own custom operating systems.
- It gives you a lot of option of programs having some different features so you can choose according to your need.
- A global development community look at different ways to enhance its security, hence it is highly secured and robust so you don't need an anti virus to scan it regularly. Companies like Google, Amazon and Facebook use linux in order to protect their servers as it is highly reliable and stable.
- Above all you don't have to pay for software and server licensing to install Linux, its absolutely free and you can install it on as many computers as you want.
- Its completely trouble free operating system and don't have an issue with viruses, malware and slowing down your computer.

Linux distributions for DevOps

CentOS (now CentOS Stream) is a more “stable” distribution designed for servers, with fewer packages available for it, and with less frequent releases. Many companies run CentOS because they want a slow, stable Linux distribution, but without the cost.

Red Hat Enterprise Linux (often called RHEL) closely follows CentOS. It's mostly used in large companies, such as banks. It's commercial (💰💰💰)

Amazon Linux is also in this list. If you spin up a basic EC2 machine in AWS, you'll probably be using Amazon Linux.

Fedora is a Linux distribution for home users and hobbyists. It's less likely to be used in DevOps or to run a server.

Ubuntu and Debian are two closely related distributions.

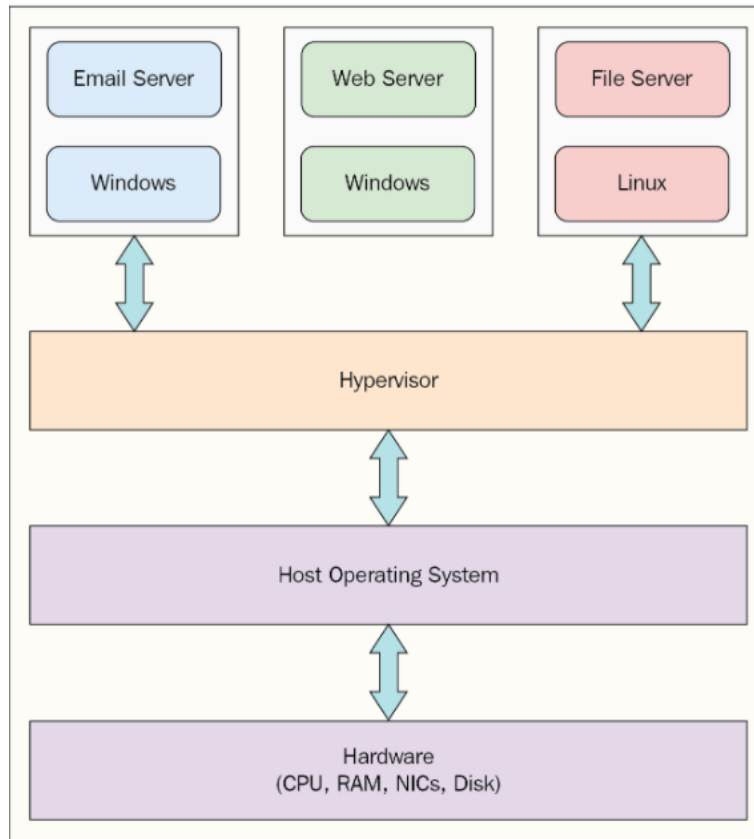
Ubuntu is massively popular, and for good reason. It's easy to install and use.



How to install Linux:

Hypervisor :

Using Hypervisor concept, we will launch linux OS on top of Windows OS



Prerequisites:

Oracle Virtual Box:

Oracle VM VirtualBox is a free and open-source hosted hypervisor for x86 virtualization, developed by Oracle Corporation.

Virtualization:

Virtualization, as name suggests, is a software that allows OS instances to run concurrently on single computer. Its type includes CPU virtualization, memory virtualization, device and I/O virtualization. It is applied in existing parts of system, so it always results in better efficiency and performance. It overall improves performance as due to technology that can balance resources.

Vagrant:

Vagrant is a tool for building and managing virtual machine environments in a single workflow. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases production parity, and makes the "works on my machine" excuse a relic of the past.

Git Bash:

Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience. Bash is an acronym for Bourne Again Shell. A shell is a terminal application used to interface with an operating system through written commands.

NotePad ++ : Text editor for windows.

Working with Vagrant Boxes

Add A Box

Vagrant uses its own versions of virtual machines, known as 'boxes' which have already been configured to allow vagrant to work with them. Hashicorp, who created Vagrant, maintain a repository of vagrant boxes. The 'vagrant box add' command is used to select and download boxes from this repository. Boxes are identified by the account name in the repository and the name of the box.

```
$ vagrant box add USER/BOX
```

For example:

```
$ vagrant box add ubuntu/trusty64
```

Which will add the 'trusty64' box from the user 'ubuntu' to your system.

List Boxes Available on Your Machine

By default, vagrant will look on your local machine for boxes. You can see which boxes are available with this command:

```
$ vagrant box list
```

Remove A Box

If you download the wrong box, or just want to remove an old one, you can do this with:

```
$ vagrant box remove
```

For example:

```
$ vagrant box remove ubuntu/trusty64
```

Initialise Vagrant

The essential part of working with vagrant is the 'Vagrantfile' which describes the type of machine to build, and how to configure it. The presence of a vagrant file in a folder gives vagrant enough to work from to build and manage virtual machines.

You can create a template vagrant file by running

```
vagrant init
```

which will create a 'Vagrantfile' in the current directory, containing a number of commented out options.

Using 'vagrant init' isn't strictly necessary for creating a vagrantfile, and you may prefer to create one yourself. An example of the contents of a simple Vagrantfile is below:

```
Vagrant.configure("2") do |config|
```

```
config.vm.box = "hashicorp/precise64"
```

```
End
```

Vagrantfiles are written using ruby syntax, but a deep knowledge of ruby is not essential. This kind of basic Depending on your requirements you may need to make some adjustments, such as making sure the provider is correct and so on.

Start the Vagrant Virtual Machine

Once you have a Vagrantfile in your project folder you can start the Vagrant virtual machine. This is as simple as running:

```
vagrant up
```

Vagrant virtual machines are typically started in the background, so there may not be obvious indications that the machine is running.

Connect to a Running Vagrant Machine

Once your vagrant machine is running then it is possible to connect to it using this command:

vagrant ssh

which will connect to your machine via ssh using vagrant's default user 'vagrant'. You can logout from this session with the 'logout' command.

Stop a Running Vagrant Machine

Once you are finished with a running machine you can stop it with this command:

vagrant halt

which is similar to manually shutting down the machine. If you want to start the machine again, you can run 'vagrant up' again.

Remove a Vagrant Machine

If you no longer need your vagrant machine at all, you can remove it with:

vagrant destroy

The Vagrantfile for your project will still exist, and you can create it again by running 'vagrant up'

VAGRANT

Introduction

Vagrant is an open-source tool that allows you to create, configure, and manage boxes of virtual machines through an easy to use command interface. Essentially, it is a layer of software installed between a virtualization tool (such as VirtualBox, Docker, Hyper-V) and a VM.

It is often used in software development to ensure all team members are building for the same configuration. Not only does it share environments, but it also shares code as well. This allows the code from one developer to work on the system of another, making collaborative and cooperative development possible.

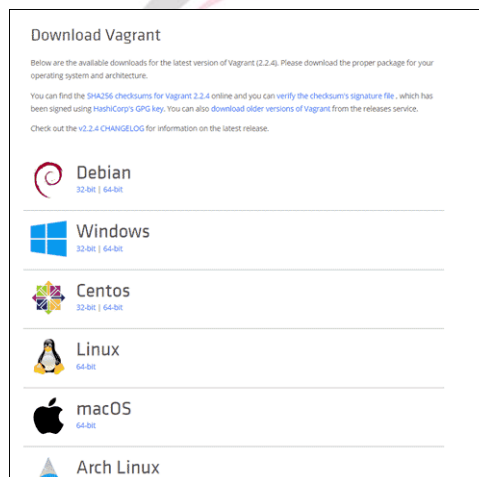
This tutorial walks you through everything you need to know about configuring and managing Vagrant.

Getting Started With Vagrant

Before you start, make sure you already have a virtualization solution on your system. Solutions that work with Vagrant include VirtualBox, VMware, Docker, Hyper-V, and custom solutions.

Installation

1. To find the latest version of Vagrant, use a web browser to navigate to its official webpage:
<https://www.vagrantup.com/downloads.html>
2. You will see a list of all the different supported operating systems, with a 32-bit and 64-bit package for each. Download the appropriate file for your operating system, then run the installer.



3. There are two ways to check if the installation was successful:

- You can either use:

```
vagrant -v
```

which should show the version number running on your computer. The latest version to date is Vagrant 2.2.6.

- Or you can type in the following command in the terminal:

```
vagrant
```

This output would show you a list of frequently used commands if the tool were installed correctly.

Vagrant Project Setup

1. Start by creating a directory to store your Vagrant file:

```
sudo mkdir vagrant-test  
cd vagrant-test
```

2. Download the Ubuntu Trusty Tahr distribution from a common library and create a basic Vagrantfile with:

```
vagrant init ubuntu/trusty64
```

If you like, you can browse to <https://app.vagrantup.com/boxes/search> and download a Vagrantbox of your choosing.

When you run the init command, Vagrant installs the box to the current directory. The Vagrantfile is placed in the same directory and can be edited or copied.

Vagrant Boxes

The basic unit in a Vagrant setup is called a “box” or a “Vagrantbox.” This is a complete, self-contained image of an operating system environment.

A Vagrant Box is a clone of a base operating system image. Using a clone speeds up the launching and provisioning process.

1. Instead of using the init command above, you can simply download and add a box with the command:

```
vagrant box add ubuntu/trusty64
```

This downloads the box and stores it locally.

2. Next, you need to configure the Vagrantfile for the virtual box it will serve. Open the Vagrantfile with the command:

```
sudo vi vagrantfile
```

3. Once the Vagrantfile is open, change the **config.vm.box** string from “base” to “ubuntu/trusty64”.

```
config.vm.box = "ubuntu/trusty64"
```

```
# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  # All Vagrant configuration is done here. The most common configuration
  # options are documented and commented below. For a complete reference,
  # please see the online documentation at vagrantup.com.

  # Every Vagrant virtual environment requires a box to build off of.
  config.vm.box = "base" → config.vm.box = "ubuntu/trusty64"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false
```

You can add another line above the end command to specify a box version:

```
config.vm.box_version = "1.0.1"
```

Or you can specify a URL to link directly to the box:

```
config.vm.box_url = "https://vagrantcloud.com/ubuntu/trusty64"
```

If you'd like to remove a box, use the following:

```
vagrant box remove ubuntu/trusty64
```

VagrantFile

Instead of building out a complete operating system image and copying it, Vagrant uses a "Vagrantfile" to specify the configuration of the box.

Providers

This tutorial shows you how to use Vagrant with VirtualBox. However, Vagrant can also work with many other backend providers.

To launch Vagrant using VMware run the command:

```
vagrant up -provider=vmware_fusion
```

Or you can launch Vagrant using Amazon Web Services with:

```
vagrant up -provider=aws
```

Once the initial command is run, subsequent commands will apply to the same provider.

Launching and Connecting

Vagrant Up

The main command to launch your new virtual environment is:

```
vagrant up
```

This will run the software and start a virtual Ubuntu environment quickly. However, even though the virtual machine is running, you will not see any output. Vagrant does not give any kind of user interface.

Vagrant SSH

You can connect to your virtual machine (and verify that it is running) by using an SSH connection:

```
vagrant ssh
```

This opens a secure shell connection to the new virtual machine. Your command prompt will change to **vagrant@trusty64** to indicate that you are logged into the virtual machine.

Once you are done exploring the virtual machine, you can exit the session with **CTRL-D**. The virtual machine will still be running in the background, but the SSH connection will be closed.

To stop the virtual machine from running, enter:

```
vagrant destroy
```

The file you downloaded will remain, but anything that was running inside the virtual machine will be gone.

Synced Folders

Vagrant automatically synchronizes content that is in your project directory with a special directory in the guest (virtual) system. The project directory is the one you created earlier, **/vagrant-test**. It's also the same one that holds the Vagrantfile.

When you log into the virtual machine, by default it starts in the **/home/vagrant/** directory. A different directory, **/vagrant/**, holds the same files that are on your host system.

You can use `vagrant up` and `vagrant ssh` to launch and log into the virtual machine, then create a test document in the `/vagrant` directory.

Use the `exit` command to close the SSH session, then use `ls` to list the contents of your `vagrant-test` directory. It should display the test file you created.

This is a handy way to manage files in the guest OS without having to use an SSH session.

Clean Up Vagrant

Once you are done working on your guest system, you have a few options how to end the session.

1. To stop the machine and save its current state run:

```
vagrant suspend
```

You can resume by running `vagrant up` again. This is much like putting the machine in **sleep mode**.

2. To shut down the virtual machine use the command:

```
vagrant halt
```

Again, `vagrant up` will reboot the same virtual machine, and you can resume where you left off. This is much like **shutting down** a regular machine.

3. To remove all traces of the virtual machine from your system type in the following:

```
vagrant destroy
```

Anything you have saved in the virtual machine will be removed. This frees up the system resources used by Vagrant.

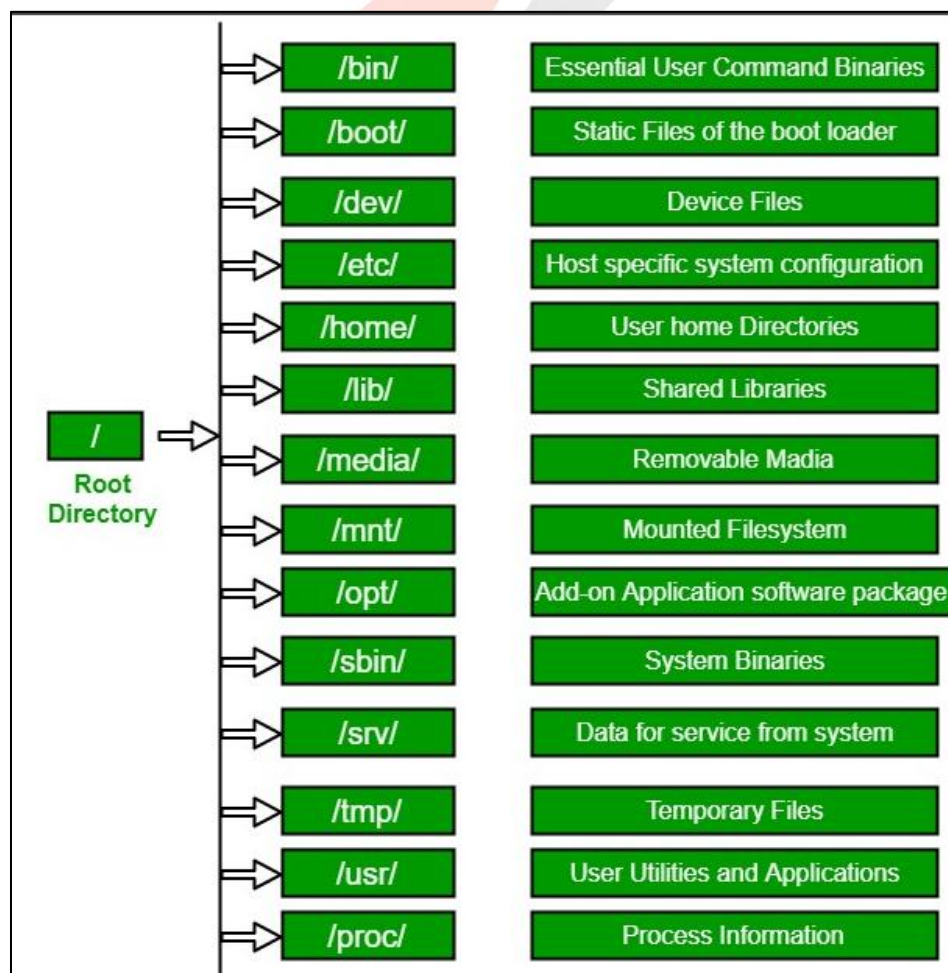
LINUX FILE HIERARCHY STRUCTURE

The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.

In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.

Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.

Most of these directories exist in all UNIX operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms other than Linux.



1. **/ (Root)**: Primary hierarchy root and root directory of the entire file system hierarchy.
 - Every single file and directory starts from the root directory
 - The only root user has the right to write under this directory
 - `/root` is the root user's home directory, which is not the same as `/`
2. **/bin** : Essential command binaries that need to be available in single-user mode; for all users, e.g., `cat`, `ls`, `cp`.

Contains binary executables

- Common linux commands you need to use in single-user modes are located under this directory.
 - Commands used by all the users of the system are located here e.g. `ps`, `ls`, `ping`, `grep`, `cp`
3. **/boot** : Boot loader files, e.g., kernels, `initrd`.
Kernel `initrd`, `vmlinuz`, `grub` files are located under `/boot`
 - Example: `initrd.img-2.6.32-24-generic`, `vmlinuz-2.6.32-24-generic`
 4. **/dev** : Essential device files, e.g., `/dev/null`.
 - These include terminal devices, `usb`, or any device attached to the system.
 - Example: `/dev/tty1`, `/dev/usbmon0`
 5. **/etc** : Host-specific system-wide configuration files.
 - Contains configuration files required by all programs.
 - This also contains startup and shutdown shell scripts used to start/stop individual programs.
 - Example: `/etc/resolv.conf`, `/etc/logrotate.conf`.
 6. **/home** : Users' home directories, containing saved files, personal settings, etc.
 - Home directories for all users to store their personal files.
 - example: `/home/kishlay`, `/home/kv`
 7. **/lib** : Libraries essential for the binaries in `/bin/` and `/sbin/`.
 - Library filenames are either `ld*` or `lib*.so.*`
 - Example: `ld-2.11.1.so`, `libncurses.so.5.7`

8. **/media** : Mount points for removable media such as CD-ROMs (appeared in FHS-2.3).
 - Temporary mount directory for removable devices.
 - Examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer
9. **/mnt** : Temporarily mounted filesystems.
 - Temporary mount directory where sysadmins can mount filesystems.
10. **/opt** : Optional application software packages.
 - Contains add-on applications from individual vendors.
 - Add-on applications should be installed under either /opt/ or /opt/ sub-directory.
11. **/sbin** : Essential system binaries, e.g., fsck, init, route.
 - Just like /bin, /sbin also contains binary executables.
 - The linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
 - Example: iptables, reboot, fdisk, ifconfig, swapon
12. **/srv** : Site-specific data served by this system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems.
 - srv stands for service.
 - Contains server specific services related data.
 - Example, /srv/cvs contains CVS related data.
13. **/tmp** : Temporary files. Often not preserved between system reboots, and may be severely size restricted.
 - Directory that contains temporary files created by system and users.
 - Files under this directory are deleted when system is rebooted.
14. **/usr** : Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications.
 - Contains binaries, libraries, documentation, and source-code for second level programs.
 - /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp

- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2
- /usr/src holds the Linux kernel sources, header-files and documentation.

15. /proc : Virtual filesystem providing process and kernel information as files. In Linux, corresponds to a procfs mount. Generally, automatically generated and populated by the system, on the fly.

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime

What are Commands

- A command is an instruction given to our computer by us to do whatever we want. In Mac OS, and Linux it is called terminal, whereas, in windows it is called command prompt. Commands are always case sensitive.
- Commands are executed by typing in at the command line followed by pressing enter key.
- This command further passes to the shell which reads the command and execute it. Shell is a method for the user to interact with the system. Default shell in Linux is called bash (Bourne-Again Shell).
- There are two types of shell commands:
- **Built-in shell commands:** They are part of a shell. Each shell has some built in commands.
- **External/Linux commands:** Each external command is a separate executable program written in C or other programming languages.

Linux Directory Commands

Directory Command	Description
pwd	The pwd command stands for (print working directory). It displays the current working location or directory of the user. It displays the whole working path starting with /. It is a built-in command.
ls	The ls command is used to show the list of a folder. It will list out all the files in the directed folder.
cd	The cd command stands for (change directory). It is used to change to the directory you want to work from the present directory.
mkdir	With mkdir command you can create your own directory.
rmdir	The <u>rmdir</u> command is used to remove a directory from your system.



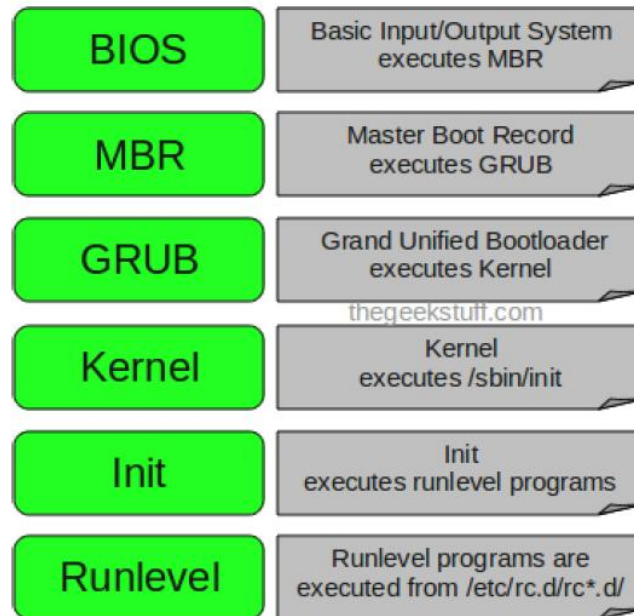
LINUX BOOT PROCESS

Boot Process (or) System Initialization:

Boot process consists the set of processes from power on the pc to login prompt comes.

Press the power button on your system, and after few moments you see the Linux login prompt.

The following 6 high level stages of a typical Linux boot process.






BIOS:

- BIOS stands for Basic Input/Output System.
- When we power on BIOS performs a POST (Power-on-self-Test) for all of the hardware components in the system to make sure everything is working properly.
- BIOS searches for MBR. So, in simple terms BIOS loads and executes the MBR boot loader.

MBR:

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda .
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.

H.No: 208-1, 2nd Floor, Annapurna Block, Ameerpet, Hyderabad – 500038

 : www.srimanit.com  : <https://youtube.com/c/SrimanIT>  : https://twitter.com/sriman_it

 : <https://www.facebook.com/srimaniTech/>  : https://www.instagram.com/sriman_it/

 : +91-99850 14433  : +91-99850 24433

- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

GRUB:

- GRUB stands for Grand Unified Boot Loader.
- In this stage, If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file.
- In older linux loader is LILO. Which is not understand file system.
- The GRUB configuration file is `"/boot/grub/grub.conf"` (`/etc/grub.conf` is a link to this).
- So, in simple terms GRUB just loads and executes Kernel and initrd images.

Kernel:

- Linux Kernel is the central core of the OS.
- Kernel executes the `/sbin/init` program
- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. To check the init process id: `#ps -ef|grep -i init`
- It Initialises devices and loads initrd module.
- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted.

INIT Process:

- The kernel, once it is loaded, finds init in `sbin(/sbin/init)` and executes it. Hence the first process which is started in Linux is init process.
- This init process reads `"/etc/inittab"` file and sets the path, starts swapping, checks the file systems, and so on.
- This `/etc/inittab` file to decide the Linux run level.

Init 0 – halt



Init 1 – Single user mode

Init 2 – Multiuser, without NFS

Init 3 – Full multiuser mode

Init 4 – unused

Init 5 – X11

Init 6 – reboot

- We can set in which runlevel we want to run our operating system by defining it on /etc/inittab file.
- Typically, you would set the default run level to either 3 or 5.
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- You can check current runlevel details of your system using below command on the terminal.

#who -r (or) #runlevel

Runlevel Programs:

- When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail OK
- Based on the selected runlevel, the init process then executes startup scripts located in subdirectories of the /etc/rc.d directory. .
- Scripts used for runlevels 0 to 6 are located in subdirectories /etc/rc.d/rc0.d through /etc/rc.d/rc6.d, respectively.

Run level 0 – /etc/rc.d/rc0.d/

Run level 1 – /etc/rc.d/rc1.d/

Run level 2 – /etc/rc.d/rc2.d/

Run level 3 – /etc/rc.d/rc3.d/

Run level 4 – /etc/rc.d/rc4.d/

Run level 5 – /etc/rc.d/rc5.d/




Run level 6 – /etc/rc.d/rc6.d/

- Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.

Note: If everything goes fine you should be able to see the Login Screen on your system.

Following are some of the important system commands in Linux.

H.No: 208-1, 2nd Floor, Annapurna Block, Ameerpet, Hyderabad – 500038

 : www.srimanit.com  : <https://youtube.com/c/SrimanIT>  : https://twitter.com/sriman_it

 : <https://www.facebook.com/srimaniTech/>  : https://www.instagram.com/sriman_it/

 : +91-99850 14433  : +91-99850 24433

uname

This will display the system information.

```
$ uname  
Linux
```

uname -o

This will display the OS information.

```
$ uname -o
```

GNU/Linux

uname -m

This will display the machine hardware information.

In the following example we are getting the machine hardware information.

```
$ uname -m  
x86_64
```

uname -v

This will display the kernel release information.

In the following example we are getting the OS kernel information.

\$ uname -v

```
#21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018
```

uname -n

This will display the network node hostname.

In the following example we are getting the network node hostname.

```
$ uname -n  
yusufshakeel-ubuntu
```

uname -a

This will display all the system related information.

In the following example we are getting all the system information.

```
$ uname -a
```

Linux yusufshakeel-ubuntu 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
x86_64 x86_64 GNU/Linux

cat /etc/os-release

This will print the version of OS installed.

I have installed Ubuntu 18.04 LTS so I am getting the related information.

\$ cat /etc/os-release

uptime

This command tells us how long the system has been running.

In the following example we can see that the system has been running for the past 55 minutes.

\$ uptime

08:37:55 up 55 min, 1 user, load average: 0.00, 0.02, 0.06

reboot

This will reboot the system.

\$ reboot

shutdown

This will shutdown the system.

\$ shutdown

whoami

This command will print the current user.

In the following example we are getting the current user.

\$ whoami

yusufshakeel

LINUX FILES

In Linux system, everything is a file and if it is not a file, it is a process. A file doesn't include only text files, images and compiled programs but also include partitions, hardware device drivers and directories. Linux consider everything as as file.

Files are always case sensitive. Let's understand it through an example.

```
sssit@JavaTpoint: ~/Downloads
sssit@JavaTpoint:~$ cd Downloads
sssit@JavaTpoint:~/Downloads$ ls
demo.txt  Demo.txt
sssit@JavaTpoint:~/Downloads$
```

In above example, we have two files named as 'Demo.txt' and 'demo.txt'. Although, they both share the same name but still they are two different files.

Linux File Commands

File : Determines file type.
Touch : Used to create a file.
Rm : To remove a file.
Cp : To copy a file.
Mv : To rename or to move a file.
Rename : To rename file.

Linux File Contents Command

There are many commands which help to look at the contents of a file. Now we'll look at some of the commands like head, tac, cat, less & more and strings.

We'll discuss about the following file contents given in the table:

Commands	Function
head	It displays the beginning of a file.
tail	It displays the last last part of a file.
cat	This command is versatile and multi worker.
more	Command line displays contents in pager form that is either in more format.
less	Command line displays contents in pager form that is either in less format.

Linux Man Command

The "man" is a short term for manual page. In unix like operating systems such as linux, man is an interface to view the system's reference manual.

A user can request to display a man page by simply typing man followed by a space and then argument. Here its argument can be a command, utility or function. A manual page associated with each of these arguments is displayed.

Syntax of man:

man [option(s)] keyword(s)

But generally [option(s)] are not used. Only keyword is written as an argument.

For example:

man ls

VI Editor with Commands in Linux/Unix Tutorial

What is the VI editor?

The VI editor is the most popular and classic text editor in the Linux family. Below, are some reasons which make it a widely used editor –

- 1) It is available in almost all Linux Distributions
- 2) It works the same across different platforms and Distributions
- 3) It is user-friendly. Hence, millions of Linux users love it and use it for their editing needs

Nowadays, there are advanced versions of the vi editor available, and the most popular one is VIM which is Vi Improved. Some of the other ones are Elvis, Nvi, Nano, and Vile. It is wise to learn vi because it is feature-rich and offers endless possibilities to edit a file.

To work on VI editor, you need to understand its operation modes. They can be divided into two main parts.

- vi Command mode
- vi Editor Insert mode
- How to use vi editor
- vi Editing commands
- Moving within a file
- Saving and Closing the file

VI Command mode:

The VI Editor

The vi editor opens in this mode, and it only understands commands

In this mode, you can, move the cursor and cut, copy, paste the text

This mode also saves the changes you have made to the file

Commands are case sensitive. You should use the right letter case.

VI Editor Insert mode:

This mode is for inserting text in the file.

You can switch to the Insert mode from the command mode by pressing 'i' on the keyboard

Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working.

To return to the command mode and save the changes you have made you need to press the Esc key

VI Editing commands

- **i – Insert at cursor (goes into insert mode)**
- a – Write after cursor (goes into insert mode)
- A – Write at the end of line (goes into insert mode)
- **ESC – Terminate insert mode**
- u – Undo last change
- U – Undo all changes to the entire line
- o – Open a new line (goes into insert mode)
- dd – Delete line
- 3dd – Delete 3 lines.
- D – Delete contents of line after the cursor
- C – Delete contents of a line after the cursor and insert new text. Press ESC key to end insertion.
- dw – Delete word
- 4dw – Delete 4 words
- cw – Change word
- x – Delete character at the cursor
- r – Replace character

- R – Overwrite characters from cursor onward
- s – Substitute one character under cursor continue to insert
- S – Substitute entire line and begin to insert at the beginning of the line
- ~ – Change case of individual character

Note: You should be in the “command mode” to execute these commands. VI editor is case-sensitive so make sure you type the commands in the right letter-case.

Moving within a file

- k – Move cursor up
- j – Move cursor down
- h – Move cursor left
- l – Move cursor right

You need to be in the command mode to move within a file. The default keys for navigation are mentioned below else; You can also use the arrow keys on the keyboard.

Saving and Closing the file

- :w – Save the file but keep it open
- :q – Quit without saving
- :wq – Save the file and quit

You should be in the command mode to exit the editor and save changes to the file.

