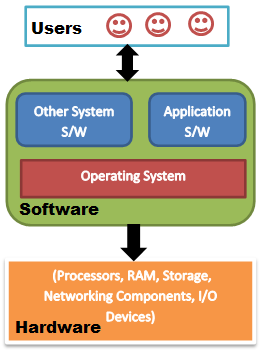
**MODULE-1**

**INTRODUCTION TO LINUX COMMANDS**

**OPERATING SYSTEM:**

* Operating system is a software that acts as an interface between the user and the computer hardware.
* It controls and manages the hardware components of the computer system based on the users and application requirements.
* It provides an environment for application software and other system software to run.
* It is an important part of the computer system and without it a computer system can’t be used.
* An operating system performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.
* Some popular Operating Systems include Linux, Windows, MacOS, VMS, OS/400, AIX, z/OS, **Apple iOS** and **Google Android** etc.
* Operating systems can be broadly categorized into two based on the number of users it allows to interact with the computer system: single user OS and multi-user OS.
* Operating systems can also be classified based on the type of user interface it provides for interaction with the computer system like CLI (Command Line Interface) and GUI (Graphical User Interface).



**Figure-1: Illustration of an OS**

**INTRODUCTION TO MULTI-USER OPERATING SYSTEM:**

* A Multi-user operating system is an operating system that allows multiple users to access the single computer system with the help of several terminals, which are connected through a network.
* It is capable of performing multiple tasks at a time coming from multiple users. For examples, one user is editing a word document while another user is executing a C program.
* It shares the resources of the computer system among different users or tasks as per their needs. For example, CPU time is sliced to execute the tasks submitted by the users. It allows concurrent access to files or data when multiple users try to access it.
* It is capable of processing of tasks in the background. For example, a printing job coming from a user is going on while another user is interacting and performing some operations on the computer.
* It can be capable of utilizing multiple processors. When a computer has multiple CPUs, it is called a multiprocessor system.
* It manages multiple users by allowing creation of user groups, roles and permissions. Multiple users with similar interests or jobs or access to specific information may form a user group. A user can be assigned different roles such as super user, administrator etc. A role is the assignment of rights or tasks to be performed by a user. Permission authorizes a user to access a particular resource of the computer system.
* A multiuser OS is relatively complex when compared to single user OS such as MS DOS that allows one user and a single task to execute at a time.
* **Example**: Linux, UNIX, Windows 2000, Mac OS Virtual Memory System (VMS) and Mainframe OS, etc.

**UNIX OS**

The UNIX operating system is a multiuser, multitasking operating system with widespread usage.

The basic structure of UNIX OS consists of a kernel layer, a shell layer and a utilities and applications layer.

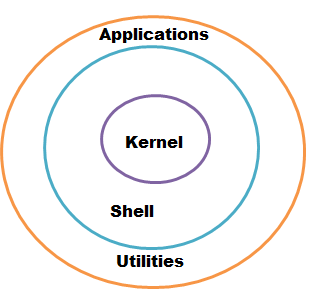


Figure-2: Basic Structure of Unix OS

## Kernel

The Kernel is the heart of the UNIX OS. It controls the hardware on behalf of the user. It handles the process, memory, file, device and network management for the operating system. The kernel is responsible for ensuring that all system and user tasks are performed concurrently.

**Shell**

The shell is the interface between the user and the kernel. It is the interpreter that translates the commands that are given by the user. It also provides programming ability to the users to write and execute shell script to perform specific tasks. There are multiple shells that are used by the UNIX OS. They include the Bourne shell (sh), the C shell (csh), the Korn shell (ksh) and the Bourne Again shell (bash). Each shell has own set of shell commands. Operating system commands are the same across all the shells.

**Utilities and Application**

The final layer of the UNIX OS is the Utilities and Applications layer. This layer includes tools and applications that offer additional functionality to the operating system.

**HISTORY OF UNIX**

* The Unix operating system found its beginnings in MULTICS (**Mult**iplexed**I**nformation and **C**omputing **S**ervice).
* General Electric, Massachusetts Institute of Technology and AT&T's Bell Laboratories jointly started a multiuser operating system project called MULTICS in mid 1960s.
* Starting in 1969, MULTICS was provided as a campus-wide information service by the MIT Information Processing Services organization, serving thousands of academic and administrative users.
* It was a mainframe timesharing operating system and an important influence on operating system development. The last MULTICS system was deactivated in 2000.
* In 1969 Bell Laboratories pulled out of the MULTICS project.
* One of Bell Laboratories people involved in the project was Kenneth Thompson. He liked the potential MULTICS had, but felt it was too complex and that the same thing could be done in simpler way.
* The first version of UNIX was created in 1969 by Kenneth Thompson at AT&T's Bell Labs. Initially it was called UNICS (Uniplexed Operating and Computing System) which was eventually shortened to UNIX.
* At Bell Labs, Ken Thompson teamed up with Dennis Ritchie, who wrote the first C compiler. In 1973 they rewrote the Unix kernel in C.
* The following year 1974, a version of Unix known as the Fifth Edition was first licensed to universities.
* In 1975, Ken Thompson spent a year of sabbatical with the University of California at Berkeley where he and two graduate students, Bill Joy and Chuck Haley, wrote the first Berkely version of Unix, which was distributed to students. The Berkeley version of Unix is known as BSD (Berkeley Software Distribution).
* In 1977 it released the first Berkeley Software Distribution, which became known as **BSD**. From BSD came the vi editor, **C shell**, virtual memory, Sendmail, and support for TCP/IP. Over time this won favour through innovations such as the C shell.
* Meanwhile the AT&T version was developing in different ways and in the year 1978, it released UNIX Seventh Edition that included the **Bourne Shell** for the first time.
* The Seventh Edition, released in 1978, served as a dividing point for two divergent lines of Unix development. These two branches are known as AT&T UNIX and BSD.
* By 1983 commercial interest was growing and AT&T developed **System V**, a more or less direct descendent of Version 7
* Compared to the other major Unix flavour BSD, System V was more conservative and focused on open standards. It was favored by commercial enterprises, which contrasted with BSD's strength in the academic arena.
* In 1987, AT&T worked with Sun Microsystems to produce System V, revision 4 (SVR4), which incorporated elements of BSD and SunOS into System V. Most commercial UNIX implementations are based on the **SVR4** model, and it has also influenced **Linux**.
* AT&T ceased development of System V when it sold its Unix holdings to Novell in 1993. Novell delivered control of the UNIX name to X/Open (now the Open Group) and after briefly marketing it as UnixWare, sold the rest of its System V interests to SCO (Santa Cruz Operation) Software Company which produced SCO Unix.

**FEATURES AND BENEFITS**

#### Multiusers:

UNIX is a multi-user system designed to support a group of users simultaneously accessing the computer system using terminals. The system allows for the sharing of processing power and peripheral resources, while at the same time providing excellent security features.

* **Multitasking**:

It is a powerful multi-tasking operating system that executes multiple tasks simultaneously. There can be a number of background processes working too. The OS handles these active and background threads efficiently and manages the system resources in a fairly shared manner.

* **Portability**:

A major contribution of the UNIX system was its portability, permitting it to move from one computer to another with a minimum of code changes.

The operating system is written in high-level language making it easier to read, understand, change and, therefore move to other machines. The code can be changed and complied on a new machine. Customers can then choose from a wide variety of hardware vendors without being locked in with a particular vendor.

* **UNIX Shell:**

UNIX has a simple user interface called the shell that acts as a command interpreter. It protects the user from having to know the intricate hardware details.  It provides a programming interface for the users to interact with the OS.

* **Pipes and Filters:**

UNIX has facilities called Pipes and Filters which permit the user to create complex programs from simple programs.

#### UNIX Tools: UNIX comes with hundreds of programs that can be divided into two classes:

**Integral utilities** that are absolutely necessary for the operation of the computer, such as the command interpreter, and

**Tools** those aren’t necessary for the operation of UNIX but provide the user with additional capabilities, such as typesetting capabilities and e-mail.

#### Applications Libraries:

Today there are hundreds of UNIX applications that can be purchased from third-party vendors, in addition to the applications that come with UNIX.

#### UNIX Communications:

UNIX e-mail permits users on the same computer to communicate with each other via their terminals.

* **Hierarchical File System:**

UNIX uses a hierarchical file structure to store information. This structure has the maximum flexibility in grouping information in a way that reflects its natural state. It allows for easy maintenance and efficient implementation.

**VERSIONS OF UNIX**

There are many different versions of UNIX. Initially, there were two main versions: the line of UNIX releases that started at AT&T (the System V versions), and another line from the University of California at Berkeley (the BSD versions).

Some other major commercial versions include SunOS, Solaris, SCO UNIX, AIX, HP/UX, and ULTRIX. The freely available versions include Linux and FreeBSD (FreeBSD is based on 4.4BSD-Lite).

## Solaris

This is the name by which the operating system of Sun Microsystems is known. It was originally called SunOS and was based on UNIX System V version 2 and BSD version 4.3. Subsequently, due to the presentation of UNIX System V version 4 a new version was developed called Solaris.

In 2010, after the **Sun** acquisition by Oracle, it was renamed Oracle **Solaris**.

## AIX

The UNIX system version for IBM machines is called AIX (Advanced Interactive eXecutive) and is based on System V version 3 and BSD 4.3. AIX runs on IBM Power Systems.

## A/UX

A/UX is Apple Inc.'s implementation of the Unix operating system for some of their Apple Macintosh computers, the latest versions running on the Mac II, Quadra and Centris series of machines.

## HP-UX

HP-UX is based on the UNIX System V Release 4 and was designed by HP for the RISC architecture of Motorola. It is a pure server operating system with high availability and flexible memory and security management.

## SCO UNIX / OpenServer

The version of Santa Cruz Operation (SCO) is based on System V / 386 version 3.2, a version of UNIX System V designed for Intel platforms.

It is now owned by Xinuos and known as **OpenServer**. Early versions of OpenServer were based on UNIX System V, while the later OpenServer 10 is based on FreeBSD.

## Linux

It was started as a simple project by Linus Torvalds, a student at the University of Helsinki in the early 90s. At present it has spread enormously all over the world and there are millions of users, both individuals and companies that use a Linux operating system.

**FEATURES OF UNIX FILE**

**FILE:**

A UNIX file is a storehouse of information – a sequence of characters. A file can be informally defined as a collection of (typically related) data, which can be logically viewed as a stream of bytes (i.e. characters). A file is the smallest unit of storage in the Unix file system.

**File attributes** are information relating to the file, but do not include the data contained within a file. File attributes for a generic operating system might include (but are not limited to):

* a file type (i.e. what kind of data is in the file)
* a file name (which may or may not include an extension)
* a physical file size
* a file owner
* file protection/privacy capability
* file time stamp (time and date created/modified)

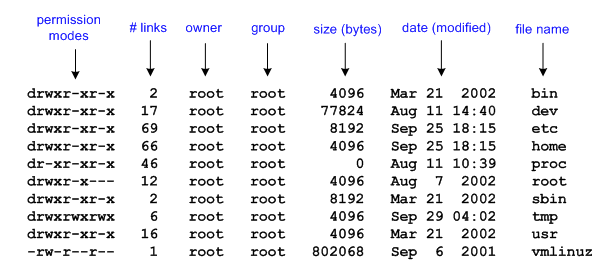


Figure-3: Sample File Attributes

**FILE TYPES:**

The most notable feature of UNIX is that it considers everything as file. Thus printer, disk, shell, kernel everything is a file. Unix divides file into three categories: ordinary files, directory files and device files. The significance of a file attribute depends on a file type.

* **Ordinary File**: This is the traditional definition of a file. It contains streams of data stored on some permanent storage device like disk. You can put any sequence of data, source program, executable program, images, video data etc.
* **Directory File**: A directory is used to organize groups of files. It contains ordinary file, device files and other sub-directories. It contains no data but keeps an account of all files and subdirectories it contains.
* **Device Files**: Unix considers any device attached to the system to be a file. Devices such as printer, tapes, hard drives, terminals all are considered as files.

**FILE SYSTEM:**

The Unix file system is a method for logically organizing and storing large quantities of data such that it becomes easy to manage. A file system consists of files, relationships to other files, as well as the attributes of each file. Additionally, file systems provide tools which allow the manipulation of files.

All of the files in the UNIX file system are organized into a multi-level hierarchy structure (an inverted tree).

At the very top of the file system is single directory called "root" which is represented by a / (slash). All other files are descendent of root. Every file other than the root must have a parent. If a file can’t be traced to its ultimate ancestor, then it is not part of the file system.

The number of levels is largely arbitrary, although most UNIX systems share some organizational similarities.

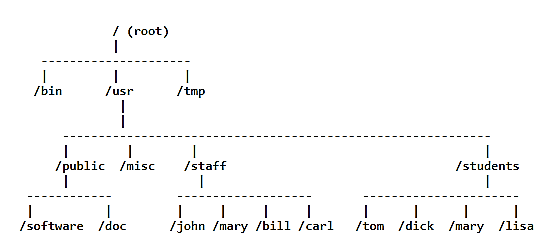


Figure-4: Example of UNIX File System

There is no single standard UNIX file structure. Most UNIX systems however, follow a general convention for files system organization at the highest level.

**/(root) *- The top level directory referred to as root.***

***Contains all files in the file system.***

**/bin *- Executable files for standard UNIX utilities***

**/dev *- Files that represent input/output devices***

**/etc *- Miscellaneous and system administrative***

***files such as the password file and system start up files.***

**/lib *- UNIX program libraries***

**/tmp *- Temporary space that can be used by programs or users.***

**/usr/bin *- More UNIX utilities. By convention /bin***

***contains standard utilities and /usr/bin***

***contains less common utilities.***

**/usr/lib *- More UNIX libraries***

**/usr/local *- Programs installed by local site***

**NAVIGATION**:

Once the relationship between various files is established in terms of its ancestor, UNIX provides a simple mechanism called **pathname** for accessing a file. UNIX allows you to move around the file system using the file pathnames. A pathname consists of a sequence of directory names, separated by /’s. A pathname can be absolute or relative.

With **absolute** file path, the pathname always begins from the root directory, complete and unambiguous. Absolute file path are sometimes referred to as fully qualified path names. Thus, absolute file specs always begin with /. For example: /etc/passwd/abc.txt

With **relative** file path, the pathname always is related to the user’s current position or location in the file system and does not start with root directory.

**GETTING STARTED**

**Logging into Unix**

Once the Unix system startup is complete, it will show a login prompt for the user to enter their username and password. If the user enters a valid username and password, then the system will log in the user and start a login session. After this, the user can open a terminal that runs a shell program. If you can’t find a launcher, or if you just want a faster way to bring up the terminal, most Linux systems use the same default keyboard shortcut to start it: **Ctrl-Alt-T**. The shell program provides a prompt where the user can proceed with running their commands.

**Logging out of Unix**

When the user wishes to end their session, they can terminate their session by logging out of the terminal or the system. To log out of a login terminal, the user can simply enter Ctrl-D or exit – both of these commands will, in turn, run the logout command that ends the login session.

**COMMONLY USED COMMANDS**

UNIX provides a number of commands for working with files and directories.

**cal**: Displays the calendar.

* **Syntax**: cal [[month] year]
* **Example**: display the calendar for April 2018
  + $ cal 4 2018

**date:** Displays the system date and time.

* **Syntax**: date [+format]
* **Example**: Display the date in dd/mm/yy format
  + $ date +%d/%m/%y

**who**: Displays the list of users currently logged in

* **Syntax**: who
* **Example**: List all currently logged in users
  + $ who

**whoami**: Displays the user id of the currently logged-in user.

* **Syntax**: whoami
* **Example**: List currently logged in user
  + $ whoami

**clear**: Clears terminal screen.

* **Syntax**: clear
* **Example**: clear the prior displayed texts
  + $ clear

**pwd**: Print the present working directory

* **Syntax**: pwd
* **Example**: Print the current working directory
  + $ pwd

**ls**: List all the files and directories in the current working directory

* **Syntax**: ls [Option]
* **Example**: List files in the current directory
  + $ ls

**Options**:

The – l option produces informative long list of files

The –x option produces a multi columnar output

**cd**: Change directory

* **Syntax**: cd directory\_name
* **Example**: Change working directory to Music
  + $ cd Music

To change to the parent directory of the current working directory use .. (two dots)

* + $ cd **..**

**mkdir**: Make directory

* **Syntax**: mkdir directory\_name
* **Example**: Create directory called progs under the current directory
  + $ mkdir progs

You can also create multiple directories by specifying multiple directory names:

* + $ mkdir dur1 dir2 dir2

**rmdir**: Remove a directory

* **Syntax**: rmdir directory\_name
* **Example**: Remove a directory progs
  + $ rmdir progs

The command can remove multiple directories also. In such case provide directory names separated by space. To delete a directory, ensure that the directory is empty. The rmdir command removes empty directory only.

**cat**: Displaying the content of a file. It can be used to create new file. It can concatenate files and print to standard output device.

* **Syntax**: cat
* **Example**: Create file1 to entered content and when done press ctrl+d to save and exit from the file.
  + $ cat > file1
* **Example**: Display contents of file1
  + $ cat file1
* **Example**: Concatenate and display contents of file1 and file2
  + $ cat file1 file2

**cp**: Copy files

* **Syntax**: cp source destination
* **Example**: Copies the abc.txt from directory progs to myprog. File in the source directory is retained
  + $ cp progs/abc.txt myprog

A group of files can be copied, but only to one destination directory

* **Example**: Copies the contents from file1 to file2 and contents of file1 is retained
  + $ cp file1 file2

**mv**: Move file or rename a file. The command can be used to rename a directory also.

* **Syntax**: mv source destination
* **Example**: Moves the abc.txt from directory progs to myprog. File in the source directory will not be available. If the destination file does not exist, it will be created; if exist, then it will be overwritten.
  + $ cp progs/abc.txt myprog

A group of files can be moved, but only to one destination directory

* **Example**: Rename the file abc.txt to abc1.txt
  + $ mv abc.txt abc1.txt

**rm**: Remove files. It can not remove a directory

* **Syntax**: rm
* **Example**: Delete file1 file2
  + $ rm file1 file2

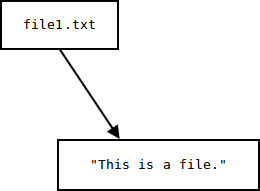
To remove all files in a directory you can use \* wild card

* $ rm \*
* $ rm progs/\*

**Linking Files**

A link is an entry in your file system which connects a file name to the actual bytes of data on the disk. It is a special type of **file** that serves as a reference to another **file** or directory. More than one file name can be linked to the same data. For example, let's create a file named **file1.txt**: $ echo "This is a file." > file1.txt

When a file is created, the operating system wrote the bytes to a location on the disk and also linked that data to a file name, **file1.txt** so that we can refer to the file in commands. If you rename the file, the contents of the file are not altered; only the information that points to it. The file name and the file's data are two separate entities. Here's an illustration of the file name and the data to help you visualize it:

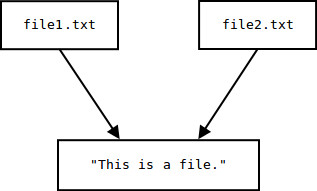


If we'll create another file name **file2.txt** for the data that already exists, we can create a link name using ln command.

**ln**: Creates a link between files.

* **Syntax**: ln file1 link\_name
* **Example**: create a link name file2.txt for file1.txt
* $ ln file1.txt file2.txt

It is important to note that we did not make a copy of this data. Both file names point to the same bytes of data on the disk. Here's an illustration to help you visualize it:



If we change the contents of the data pointed to by either one of these files, the other file's contents are changed as well.

For example, let's append a line to one of them using the **>>** operator:

$ echo "It points to data on the disk." >> file1.txt

And check the contents of the files are same. If we remove one file, the actual data is not removed, the link is removed.

**chmod**: This command is used to change the file access permissions. These permissions are read, write and execute permission for the owner, group, and others.

|  |  |  |
| --- | --- | --- |
| **Category** | **Operation** | **Permission** |
| u - user  g - group  o - others  a - all | + assign permission  - remove permission  = assign absolute permission | r - read  w - write  x - execute |

* **Syntax**: chmod [category][operation][permission] file
* **Example**: To enable all user to execute a file either of the following
  + $ chmod ugo+x file1
  + $ chmod a+x file1
  + $ chmod +x file1

You can same permission to a group of files also

* **Example**: To remove write permission from both group and others
  + $ chmod go-w file2
* **Example**: To remove write and read permission from others
  + $ chmod o-rw file3
* **Example**: If you want to assign only read permission to both group and others, and remove all other permissions, then you can use absolute permission
  + $ chmod go=r file4

**Octal Notation for chmod Command**

You can use octal notation as short-hand notation for file permission. It takes an octal representation of the permission. The read permission has value 4, write permission has a value 2 and execute permission has a value 1. When there is more than one permission associated with a particular category, the respective numbers are add. For example, if a file has read and write permission for the owner, then the octal representation of the permission will be 4 + 2 = 6. If the same permission is also repeated for the group and others, then it will be a three digit number for the category

* + $ chmod 666 file5

**Text Processing using Pipe and Filter Commands**

Using pipes and filters, one can perform powerful text processing in Linux environment.

Using a **pipe**, you can connect two commands together so that the output from one program becomes the input of the next program. Put a vertical bar (**|**) on the command line between two commands to make a pipe.

A **filter** is a program that takes its input from standard input or a file or another program, performs an operation on it and writes the results to standard output. Linux has a number of filters.

**grep:** (global regular expression printer) It is a command for pattern searching in a file and prints those lines containing that specified pattern. If the file name is not mentioned, grep searches in stdin.

* **Syntax**: grep [option] pattern [file]
* **Example**: Prints those lines with their line numbers which contains capital letters in the range of A to M
  + $ grep -n ‘[A-M]’ file1
* **Example**: Prints those lines which contains a word good
  + $ grep good file1

**sed:** (stream editor) It is a command for filtering and transforming text. The most common use of SED command in UNIX is for substitution (find and replace). By using SED you can edit files even without opening it, which is much quicker way to find and replace something in file, than first opening that file in an Editor and then changing it.

* **Syntax**: sed [options] action [file]
* **Example**: Substitute string ‘boy’ with string ‘girl’ in file1
  + $ sed ‘s/boy/girl/’ file1

**awk:** Itis a remarkable pattern scanning and processing language, it can be used to build useful filters in Linux.

* **Syntax**: awk [options] action [file]
* **Example**: Print the lines which match with the given pattern.
  + $ awk ‘/Boy/ {print}’ file1

**tr:** This tool translates or deletes characters from standard input and writes results to standard output. Options are:

-c : complements the set of characters in string.

-d : delete characters in the exp1 from the output.

-s : replaces repeated characters listed in the exp1 with single occurrence

-t : truncates exp1

* **Syntax**: tr [options] [exp1] [exp2]
* **Example**: To remove all the digits from the string.
  + $ echo "my name is Raju535" | tr -d [:digit:]
* **Example**: to remove all characters except digits
  + $ echo "my name is Raju535" | tr -cd [:digit:]

**Comparing two files**

**cmp**: This command is used to compare two files character by character.

* **Syntax**: cmp [options] file1 file2
* **Example**: Compare file1 and file2.
  + $ cmp file1 file2

**diff**: This command is used to compare two files line by line. The output indicates how the lines in each file are different, and the steps involved changing file1 to file2.

* **Syntax**: diff [options] file1 file2
* **Example**: Find difference between file1 and file2
  + $ diff file1 file2

**Disk Commands**

These commands are normally used by administrators.

**fdisk**: Fdisk is the most commonly used command to check the partitions on a disk. The fdisk command can display the partitions and details like file system type. However it does not report the size of each partitions.

* **Example**: $ sudo fdisk –l

**df**: df (Disk free) reports the amount of free space available on the disk

* **Example**: $ df

**du**: du (Disk usage) reports the amount of disk consumption

* **Example**: $ du

**INTRODUCTION TO LINUX FLAVORS**

Linux is a family of open source UNIX operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged in a Linux distribution that includes the Linux kernel and supporting system software and libraries, many of which are provided by the GNU Project. Popular Linux distributions are Ubuntu, Fedora, Debian, Red Hat Enterprise Linux, Centos, SUSE Enterprise Linux etc.

**Ubuntu**

Ubuntu is the most popular distribution today even though it was announced late in 2004. Founded by South African millionaire Mark Shuttleworth, Canonical--the company behind Ubuntu--for many years shipped Ubuntu CDs to interested users for free, thus speeding its market penetration. You can now find it in smartphones, tablets, PCs, servers, and cloud VPS (Virtual Private Servers).

Ubuntu is based on Debian and includes well-known apps such as Firefox and OpenOffice. It has a predictable, six-month release schedule, with occasional Long Term Support (LTS) versions that are supported with security updates for three to five years. Ubuntu is the base for other distributions of the Canonical family such as Kubuntu, Xubuntu, and Lubuntu. Most of these differ primarily by offering a desktop environment other than Ubuntu's standard GNOME.

**Debian**

Dating back to 1993, Debian is currently known as one of the most well-tested and bug-free distros available today. It is used both on desktop and server computers, also to run the infrastructure that runs the clouds. Being one of the two oldest and famous Linux distributions, it is the basis of numerous popular Linux distributions notably **Ubuntu** and **Kali Linux**.

Debian Linux is committed to free software (so it will always remain 100% free) but it also allows users to install and use non-free software on their machines for productivity. Although its strength is mainly visible in servers, the desktop edition has seen remarkable improvements in features and appearance. Debian also has a relatively slow release cycle, with stable ones coming out every one to three years.

**Fedora**

Fedora is the free version of Red Hat, who’s RHEL (Red Hat Enterprise Linux) has been a commercial product since 2003. Fedora continues to be one of the top used distributions for years now due to its three main available versions (Workstation (for desktops), Server edition, and Cloud image).

However, perhaps the most distinguishing characteristic of Fedora is that it’s always on the lead of integrating new package versions and technologies into the distribution. In addition, new releases of Red Hat Enterprise Linux and **CentOS** are based on Fedora. Fedora also offers a six-month release schedule, and its security features are excellent.

**Linux Mint**

Linux Mint is an Ubuntu-based distro that was just launched in 2006. The operating system adds to Ubuntu with its own, distinct desktop theme and a different set of applications; also unique to the distro are a variety of graphical tools for enhanced usability, such as mintDesktop for configuring the desktop environment, mintInstall for easier software installation and mintMenu for easier navigation.

Mint enjoys a well-deserved reputation for ease of use, so it's another good one for beginning users. It also includes some proprietary multimedia codecs that are often absent from larger distributions, thereby enhancing its hardware compatibility. Mint doesn't have a fixed release schedule.

**MX Linux**

MX Linux is another popular distribution thanks to its high stability, elegant and efficient desktop, and also an easy learning curve. It is a midweight desktop-oriented Linux operating system based on **Debian**. It comes with a simple configuration, solid performance, and a medium-sized footprint. It is built for all types of users and applications.

**SUSE Linux Enterprise Server**

SUSE Linux Enterprise Server is one of the most popular solutions for cloud-based servers. You will have to opt for a subscription in order to get priority support and assistance to manage your open source solution.

**OpenSUSE**

It provides a feature-rich and beautiful desktop experience. OpenSUSE supports a variety of desktop managers. OpenSUSE is aimed at developers and system administrators. For that reason, it’s extremely stringent on security protocols.

It does not focus on regular release rather rolling for a long-term cycle and cutting edge stable features.

**CentOS**

CentOS is more like a community edition of Red Hat Enterprise Linux. It is open source and free as well. CentOS ranks among those distributions optimized for server environments as well as desktop clients. CentOS is robust, feature-rich, and stable. You may find CentOS images on a variety of cloud platforms.

**INSTALLING LINUX INSIDE WINDOWS**

There are several ways to install Linux.

You can clean everything from your system and install Linux.

You can dual boot Linux with Windows and choose one of the operating systems at the boot time.

You can even install Linux without making any changes to your Windows system, using virtual machine (i.e. VirtualBox).

**Installing using VirtualBox**

VirtualBox is free and open source virtualization software from Oracle. It enables you to install other operating systems in virtual machines. It is recommended that your system should have at least 4GB of RAM to get decent performance from the virtual operating system.

**Step 1: Download and install VirtualBox**

Go to the website of Oracle VirtualBox and get the latest stable version from here:

<https://www.virtualbox.org/>

**Step 2: Download the Linux ISO**

Next, you need to download the ISO file of the Linux distribution. You can get this image from the official website of the Linux distribution you are trying to use.

I am using Ubuntu in this example, and you can download ISO images for Ubuntu from the link below:

<https://ubuntu.com/desktop>

**Step 3: Install Linux using VirtualBox**

You have installed VirtualBox and you have downloaded the ISO for Linux. You are now set to install Linux in VirtualBox.

Here are some links to webpages that describe steps to install:

<https://itsfoss.com/install-linux-in-virtualbox/>

<https://brb.nci.nih.gov/seqtools/installUbuntu.html>

<https://www.wikihow.com/Install-Ubuntu-on-VirtualBox>