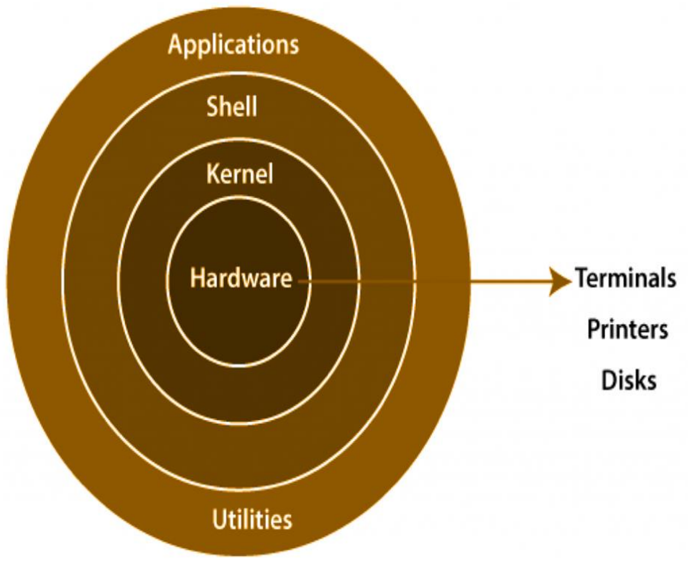
Linux Architecture

Linux is similar to other operating systems you may have used before, such as Windows, macOS (formerly OS X), or iOS. Like other operating systems, Linux has a graphical interface, and the same types of software you are accustomed to, such as word processors, photo editors, video editors, and so on. Linux was created in 1991 by Linus Torvalds, a then-student at the University of Helsinki. Torvalds built Linux as a free and open source alternative to Minix, another Unix clone that was predominantly used in academic settings. He originally intended to name it "Freax," but later renamed it as "Linux" after a combination of Torvalds' first name and the word Unix.

The diagram illustrates the structure of the Linux system, according to the layers concept.



The Linux architecture is largely composed of elements such as the Applications, Shell , Kernel, Hardware layer, System Utilities and Libraries.

Applications

Applications are the programs that the user runs on top of the architecture. The applications are the user space element that includes database applications, media players, web browsers, and presentations.

Shell

Shell is the interface that interacts with humans and processes the commands that are given for the execution. One can call it an interpreter because it takes the command from the keyboard and makes it understandable to the kernel.

Shells are categorized into two sections:

Command Line Shell

Graphical User Shell

The command line shell is the user interface where the user types commands in a text form. When the user provides the command in the terminal, the shell interprets the commands for the kernel. The shell also has some built-in commands that help the user to navigate, manage, and change the file system

The graphical user shell is the user interface using the system's peripheral components like a mouse, and keyboard. It is beneficial for users who are not familiar with commands. These shells are used to make the desktop environment easier.

Kernel

In Linux operating systems, the kernel is the core component that acts as the bridge between computer hardware and applications. It is responsible for managing the system's resources and allowing software and hardware to communicate with each other.

System Utilities and Libraries

The system utilities and libraries provide a wide range of functions to manage the system. Low-level hardware complexity to high-level user support is served by the system utilities and libraries.

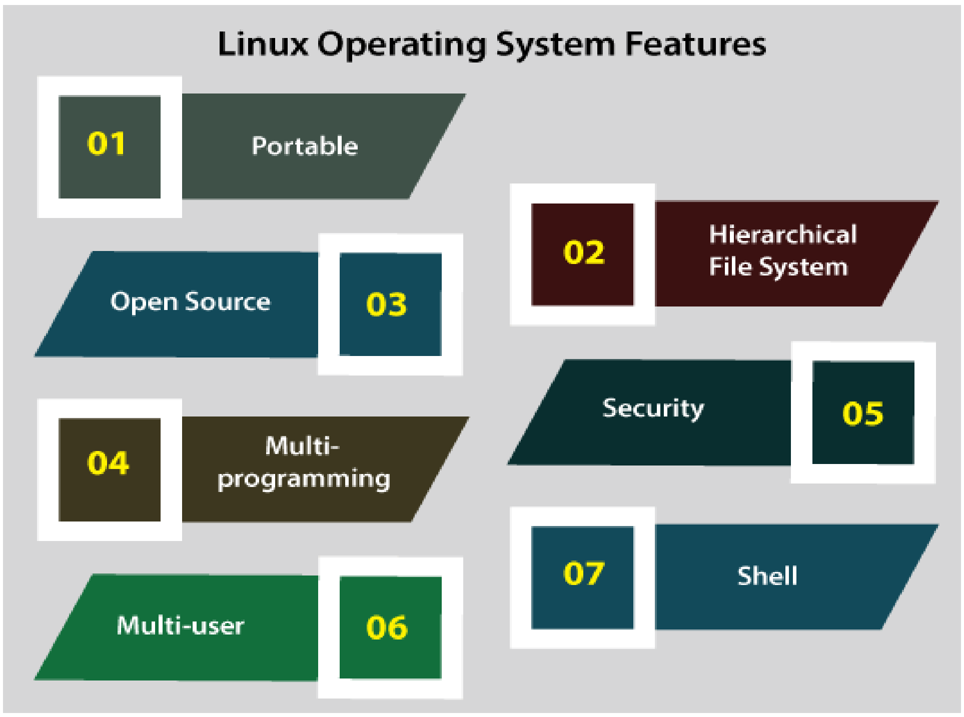
- **System Utilities:** It is the program that performs the task which is given by the users and manages the system.(Antivirus,Winrar,Winzip etc)
- **System Libraries:** Functions through which the system interacts with the kernel and handles all the functionalities without the kernel privileges

Hardware

Hardware layer of Linux is the lowest level of operating system track. It is plays a vital role in managing all the hardware components. It includes device drivers, kernel functions, memory management, CPU control, and I/O operations. This layer generalizes hard complexity, by providing an interface for software by assuring proper functionality of all the components.

Some Linux distributions are: MX Linux, Manjaro, Linux Mint, elementary, Ubuntu, Debian, Solus, Fedora, openSUSE, Deepin

Linux Operating System Features



1. **Portable:** Linux OS can perform different types of hardware and the kernel of Linux supports the installation of any type of hardware environment.
2. **Hierarchical file system:** Linux OS affords a typical file structure where user files or system files are arranged.
3. **Open source:** Linux operating system source code is available freely and for enhancing the capability of the Linux OS, several teams are performing in collaboration.
4. **Multiprogramming:** Linux OS can be defined as a multiprogramming system. It means more than one application can be executed at the same time.
5. **Security:** Linux OS facilitates user security systems with the help of various features of authentication such as controlled access to specific files, password protection, or data encryption.
6. **Multi-user:** Linux OS can also be defined as a multi-user system. It means more than one user can use the resources of the system such as **application programs, memory,** or **RAM** at the same time.
7. **Shell:** Linux operating system facilitates a unique interpreter program. This type of program can be applied for executing commands of the operating system. It can be applied to perform various types of tasks such as call application programs and others.

Linux Commands

echo command

The **echo** command in Linux and Unix-like operating systems is a built-in command used primarily to display messages or output text to the terminal or into a file.

Syntax

echo [option] [string]

1.How to input a text to get the output using the echo command?

Input : echo “Linux Commands”

Output:- Linux Commands

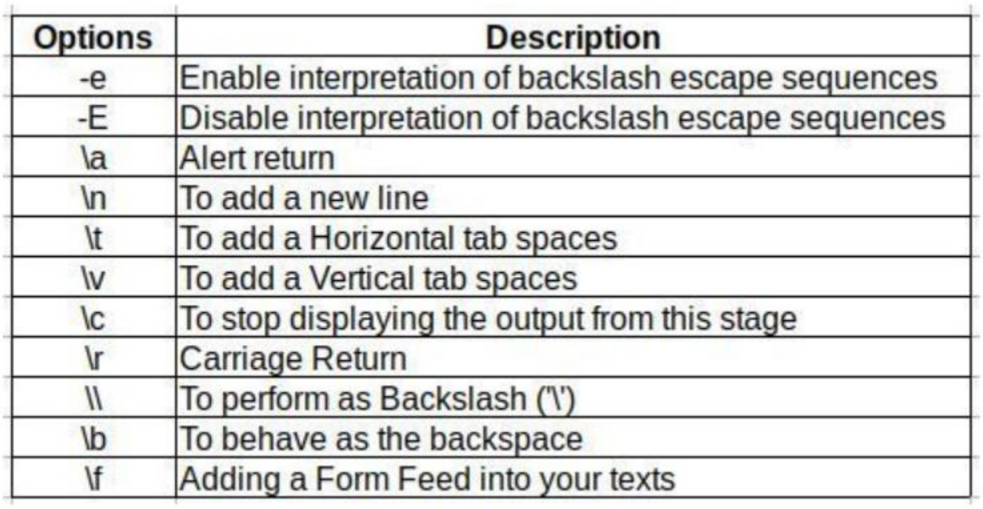
2. How to print a variable?

x=100

Input : echo “The value of x =$x”

Output : The value of x = 100

The following tabular gives you the possible options in the echo command:



Options	Description
-e	Enable interpretation of backslash escape sequences
-E	Disable interpretation of backslash escape sequences
\a	Alert return
\n	To add a new line
\t	To add a Horizontal tab spaces
\v	To add a Vertical tab spaces
\c	To stop displaying the output from this stage
\r	Carriage Return
\\	To perform as Backslash ('\')
\b	To behave as the backspace
\f	Adding a Form Feed into your texts

3. How to add a new line?

Input: echo -e "Linux \n commands"

```
Output:Linux
        Commands
```

Note: The Linux "echo" command provides the "-e" option, which instructs the command to interpret backslash escape sequences in the specified text. By utilizing the '\n' sequence, we can add a new line and print the subsequent text on a new line below the previous one. This enables us to format the output and make it more readable by separating it into distinct lines.

5. How to add a horizontal tab to your content?

```
echo -e "Linux \t commands"
```

```
Output: Linux    commands
```

PATH

PATH is an Environment variable in Linux .Environment variables are dynamic values that affect the processes or programs on a computer. It essentially tells the shell which directories to search through to find the executable files (programs or scripts) that match the command names entered by the user. This search process occurs in the order the directories are listed within the **PATH** variable.

To view the path (echo command)

$echo $PATH

Output

```
usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games
```

The **PATH** variable contains a list of directories separated by colons (**:**). When a command is entered, the shell searches for an executable file with the command's name starting from the first directory listed in the **PATH** variable and proceeds to the next if it's not found. This continues until either the executable is found or the list is exhausted. If the executable is not found in any of the directories listed in **PATH**, the shell typically returns a "command not found" error.

Adding a Directory to the PATH Environment Variable(export Command)

A directory can be added to PATH in two ways: at the start or the end of a path.

Adding a directory (/the/file/path for example) to the **start of PATH** will mean it is checked first:

export PATH=/the/file/path:$PATH

echo $PATH

Output :

```
the/file/path:usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games
```

Adding a directory to the **end of PATH** means it will be checked after all other directories:

export PATH=$PATH:/the/file/path

echo $PATH

Output :
usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/ the/file/path

Multiple directories can be added to PATH at once by adding a colon: between the directories:

export PATH=$PATH:/the/file/path:/the/file/path2

echo $PATH

Output :
usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/

the/file/path:/the/file/path2

man

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.

The basic man command syntax is:

man [option] [section number] [command name]

option – the search result output.

section number – the section in which to look for the man page.

command name – the name of the command which man page you want to see.

But generally [option(s) and section number] are not used. Only command is written as an argument.

some options include

- **-k KEYWORD** (Search for the **KEYWORD** in the whole **manual** page and shows all the **matches**)
- **-f KEYWORD** (Look for a short description of any **KEYWORD** or Command)
- **-d, –default** (Resets the **man** command behavior to **default**)
- **-i, –ignore-case** (**Ignore case sensitivity** of the command)
- **-I, –match-case** (Looking inside the man page with **case sensitivity**)

- **-a, –all** (Shows **all manual** pages that **match** the specific **keyword** or command)

By default, **man** looks in all the available sections of the manual and shows the first match (even if the page exists in several sections). Providing a section number instructs the **man** command to look in a specific section.

There are nine sections of the manual:

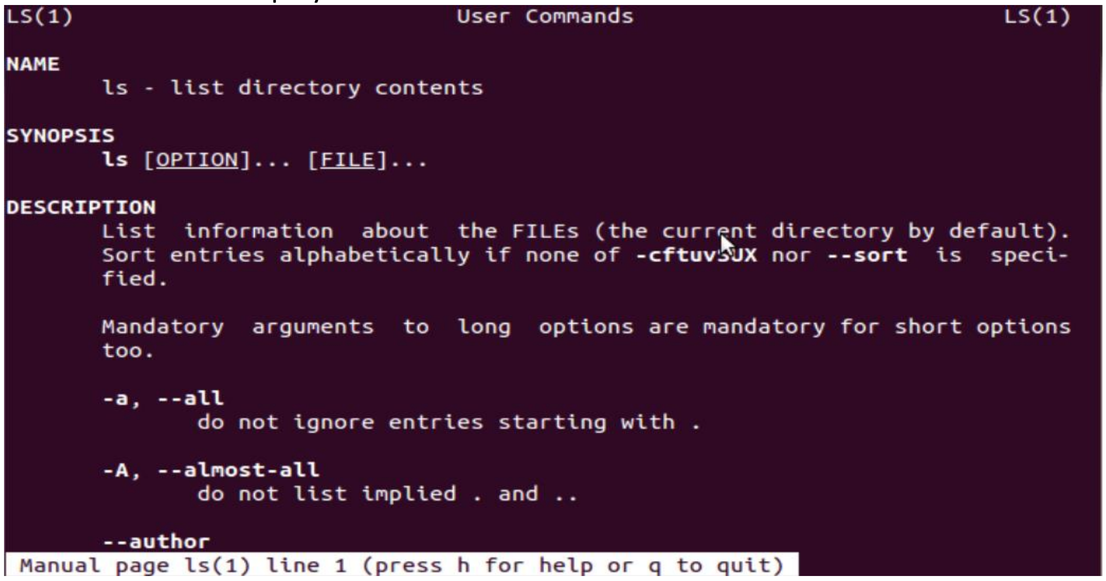
1. **General commands:** Commands used in the terminal.
2. **System calls:** Functions the kernel provides.
3. **Library functions:** Functions in program libraries.
4. **Special files:** Usually devices found in **/dev** and related drivers.
5. **File formats and conventions:** File formats like **etc/passwd**.
6. **Games:** Descriptions of commands that display database quotes.
7. **Miscellaneous:** Various descriptions, including macro packages and conventions, boot parameters, and others.
8. **System administration commands:** Commands mostly reserved for root.
9. **Kernel Routines:** Information about internal kernel operations.

Example

Syntax of command without option and section

man ls

This command will display all the information about **'ls'** command as shown in the screen shot.



```
LS(1)                         User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
 Manual page ls(1) line 1 (press h for help or q to quit)
```

Syntax for a particular section:

man section_number command

Example : man 2 passwd

The **man 2 passwd** command in Linux is intended to show the manual page for the **passwd** system call, which is found in section 2 of the manual. Section 2 of the manual pages typically documents system calls, which are interfaces provided by the Linux kernel.

printf

printf command is used to output a given string, number or any other format specifier. The command operates the same way as *printf* in C, C++, and Java programming languages.

Example

```
$ printf "%s" "Hello, Welcome to KMIT”
Output: Hello, Welcome to KMIT
```

script

script command in Linux is used to make typescript or record all the terminal activities. After executing the *script* command it starts recording everything printed on the screen including the inputs and outputs until exit. By default, all the terminal information is saved in the file *typescript* , if no argument is given.

```
script
```

We can specify a filename as an argument to save the output to a different file:

```
script my_session.log
```

To end the recording session, type **exit** or press **Ctrl-D**. This will return you to your normal terminal session and stop recording.

Example:

```
script my_session.log
echo "Hello, World!"
ls
exit
```

passwd

The **passwd** command in Linux and Unix-like operating systems is used to change the password of a user account. It can be used by both the system administrators to change other users' passwords and by individual users to update their own passwords.

Basic Usage

For a user to change their own password, they would simply type:

passwd

Upon execution, the system prompts the user to enter their current password (for verification), followed by the new password, and then to retype the new password for confirmation.

uname

The **uname** command in Linux and Unix-like operating systems is used to display system information. It provides details about the kernel name, version, and other system information. By default, without any options, **uname** will print the kernel name.

Options

The **uname** command supports several options that can be used to display specific system information:

- **-a**, **--all**: Print all available system information (kernel name, nodename, kernel release, kernel version, machine, processor, hardware platform, and operating system).
- **-s**, **--kernel-name**: Print the kernel name.
- **-n**, **--nodename**: Print the network node hostname.
- **-r**, **--kernel-release**: Print the kernel release.
- **-v**, **--kernel-version**: Print the kernel version.
- **-m**, **--machine**: Print the machine hardware name (e.g., x86_64).
- **-p**, **--processor**: Print the processor type or "unknown".
- **-i**, **--hardware-platform**: Print the hardware platform or "unknown".
- **-o**, **--operating-system**: Print the operating system.

Example

Displaying the Kernel Name

Simply typing uname without any options will display the kernel name

Example output might be Linux *for a Linux system.*

who

The **who** command in Linux and Unix-like operating systems is used to display information about users who are currently logged into the system. It provides a list of users, the terminals they are logged in from, the login time, and sometimes the host from which they are accessing the system.

Basic Usage

To run the command, simply type:

```
who
```

This will display a list that typically includes the username, terminal name , the date and time of login, and the remote host name or IP address from which the user is accessing the system.

Options

The **who** command supports several options that can alter its output or provide additional information:

- **-a**, **--all**: Show all information, combining the effects of many other options.
- **-m**: Same as **who am i**, showing information about the current terminal. It's equivalent to running **who** with the **$USER** variable.
- **-q**, **--count**: Display only the names and number of users currently logged on.
- **-H**, **--heading**: Include column headers in the output.
- **-r**, **--runlevel**: Show the current runlevel. This is useful in multi-user environments and for system administrators.
- **-u**, **--users**: Show the current login name, terminal line, login time, idle time and the exit status of a process. The idle time is particularly useful for finding out how long a terminal has been inactive.

Date

The **date** command in Linux and Unix-like operating systems is used to display or set the system's date and time. By default, running the **date** command without any options will display the current date and time according to the system's settings.

Basic Usage

To display the current date and time, simply type:

```
date
```

Output : **Tue Mar 9 12:34:56 PST 2021**

Options

The **date** command supports several options to format the output, set the system's date and time, and more. Some commonly used options include:

- **+%FORMAT**: Allows you to specify the output format of the date/time. For example, **date +"%Y-%m-%d %H:%M:%S"** would output the date and time in the format **YYYY-MM-DD HH:MM:SS**.
- **-u**, **--utc**, **--universal**: Displays or sets the Coordinated Universal Time (UTC).

we can pass the strings like "yesterday", "monday", "last monday" "next monday", "next month", "next year," and many more.

Consider the below commands:

1. date -d now
2. date -d yesterday
3. date -d tomorrow
4. date -d "next monday"
5. date -d "last monday"

The above commands will display the dates accordingly.

Set or Change Date in Linux

To change the system clock manually, use the set command. For example, to set the date and time to *5:30 PM, May 13, 2010*, type:

```
date --set="20100513 05:30"
```