1. Operating Systems

-An operating system is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all of its software and hardware.

-Most of the time, there are several different computer programs running at the same time, and they all need to access your computer's central processing unit (CPU), memory, and storage. The operating system coordinates all of this to make sure each program gets what it needs.

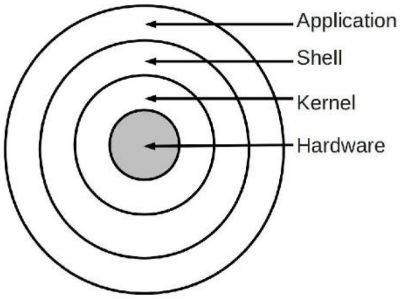
-Microsoft Windows, macOS, and Linux.

1.1. Operating System Layers

Hardware

-This layer interacts with the system hardware and coordinates with all the peripheral devices used such as printer, mouse, keyboard, scanner etc. The hardware layer is the lowest layer in the layered operating system architecture.

-The hardware consists of the memory, CPU, arithmetic-logic unit , I/O, peripheral devices (A peripheral device is an internal or external device that connects directly to a computer or other digital device but does not contribute to the computer's primary function, External Drive. USB Flash Drive. CD-ROM) and other physical devices.

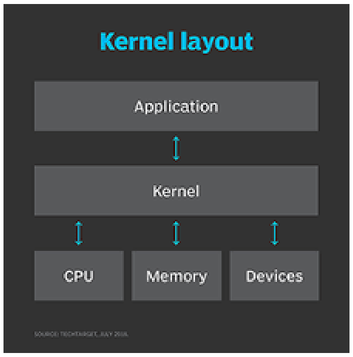


Kernel

- the kernel is the central component of most computer operating systems (an important piece of software)

- it is a bridge between applications and the actual data processing done at the hardware level.

-The kernel's responsibilities include managing the system's resources (the communication between hardware and software components).



SOURCE: TECHTARGET, JULY 2016

-Following are the **functions of a Kernel:**

-Access Computer resource: A Kernel can access various computer resources like the CPU, I/O devices and other resources.
It acts as a bridge between the user and the resources of the system.

-Resource Management: It is the duty of a Kernel to share the resources between various process in such a way that there is uniform access to the resources by every process.

-Memory Management: Every process needs some memory space. So, memory must be allocated and deallocated for its execution. All these memory management is done by a Kernel.

-Device Management: The peripheral devices connected in the system are used by the processes. So, the allocation of these devices is managed by the Kernel.

Shell

-Shell – An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's

-A shell is a piece of software that provides an interface for users to an operating system which provides access to the services of a kernel.

- The name shell originates from shells being an outer layer of interface between the user and the innards of the operating system (the kernel).

-shell → command line interpreter

functions.

-The shell manages the interaction between you and the operating system by prompting you for input, interpreting that input for the operating system, and then handling any resulting output from the operating system.

-Shells provide a way for you to communicate with the operating system. This communication is carried out either interactively (input from the keyboard is acted upon immediately) or as a shell script

-Operating system shells generally fall into one of two categories: command-line and graphical. Command-line shells provide a command-line interface (CLI) to the operating system, while graphical shells provide a graphical user interface (GUI)..

- Graphical shells provide means for manipulating programs based on graphical user interface

Application

This is the highest layer in the layered operating system. This layer deals with the many user programs and applications that run in an operating system such as word, games, browsers etc.

2. Open Source and Linux History

-In 1969, an engineer and systems expert at AT&T's Bell Laboratories found himself without work (and quite bored) as the project that he and his team had been working on, the **Multics** (Multiplexed Information and Computing Service) time-sharing operating system, had crumbled before his eyes.

-**Ken Thompson** found himself creating and playing computer games on the company's expensive GE 635 mainframe, which he had access to from working on Multics. One of the games in particular, **Space Travel**, was loved by Ken,

-Ken decided to attempt to port the game to an older computer he had found in Bell Labs, a Digital Equipment Corporation (DEC) PDP-7 that was basically left abandoned as it was already considered an obsolete machine at the time.

-In order to port the game to the more efficient and less expensive PDP-7, Ken designed his own operating system, which drew heavy inspiration from the Multics project. While still having access to the Multics environment, Ken began writing a new file and paging system on it.

-By the time late 1969 had rolled around, Thompson, along with former Multics colleague, Dennis Ritchie, built a team of Bell Labs engineers to design a simpler and more efficient version of Multics, which included a hierarchical file system, computer processes, device files, a command-line interpreter, and some smaller utility programs.

-This new operating system would form the base of what would eventually become the first version of Unix,

C became the heart of Unix and remains one of the most prolific programming languages in history, is still heavily used today

-many versions of unix was relased and The latter half of the 1980s saw Unix become popular outside of academic and technical circles. It became viable in unexpected places like large commercial installations and mainframe computers.

-By the 1980s, almost all software was proprietary, which means that it had owners who forbid and prevent cooperation by users. This made the GNU Project necessary.

-Every computer user needs an operating system; if there is no free operating system, then you can't even get started using a computer without resorting to proprietary software. So the first item on the free software agenda obviously had to be a free operating system.

-The GNU Project was conceived in 1983 as a way of bringing back the cooperative spirit that prevailed in the computing community in earlier days—to make cooperation possible once again by removing the obstacles to cooperation imposed by the owners of proprietary software

-We decided to make the operating system compatible with Unix because the overall design was already proven and portable, and because compatibility makes it easy for Unix users to switch from Unix to GNU.

-A Unix-like operating system includes a kernel, compilers, editors, text formatters, mail software, graphical interfaces, libraries, games and many other things. Thus, writing a whole operating system is a very large job.

-The GNU operating system is a complete free software system, The project to develop the GNU system is called the "GNU Project."

- The GNU Project believed in freedom to use a computer however one would like, including the software available. They began rewriting many popular Unix programs in order to bypass the UNIX copyright and distribute the system to anyone for free.

-By 1990 we had either found or written all the major components except one—the kernel.

-Then Linux, a Unix-like kernel, was developed by **Linus Torvalds** in 1991 and made free software in 1992.

-Combining Linux with the almost-complete GNU system resulted in a complete operating system: the GNU/Linux system.

3. Linux Distribution Families

1.Red Hat Family

This family concentrated on the enterprise side of things, such as servers and company workstations.

2.Debian Family

The Debian family started with the home user in mind, the community wanted to make GNU/Linux available for the average user as much as it was for enterprises at the time.

3.Other distributions built for specific use cases**

Distributions such as Arch Linux, openSUSE, SLES, Gentoo, and many others were made for specific use cases or optimisations based on what the community wanted.

4. Why Linux?

1.Open Source Nature

Linux is completely an open source project. You can have a look at the source code of a Linux OS.

2.Secure

The process of package management, the concept of repositories, and a couple more features makes it possible for Linux to be more secure than Windows.
3.Customisability
The operating system is very modular and customisable, which allows you to create your own customised system according to your needs.
4. Software Updates
Linux notice more effective and faster updates to fix the problems you might be facing due to lareg community Support.
5.Good Development Environment
Due to features like package managers, the command line, the operating system being very low on resource usage, customisability, and many more, GNU/Linux is a great development environment.
6.Privacy
Linux distributions do not collect much data (or none). Moreover, you will not be needing additional tools to protect your privacy.
7.Hardware
Linux systems are known for consuming fewer system resources (RAM, disk space, etc.)

1.1. What is the shell?

Shell :

Shell stands for the command-line interpreter. A shell is a program that processes commands and outputs the results. A shell is a layer that sits on top of the kernel:

1) It interprets and processes the commands entered by the user. Unlike users, the shell has access to the kernel. Users can only gain access to the kernel by using a shell and entering commands (i.e. running programs). System calls are used by programs to gain access to kernel functionality. The system API is made up of all system calls.
A shell is a program that acts as an interface between a user and the kernel. It allows a user to give commands to the kernel and receive responses from it. Through a shell, we can execute programs and utilities on the kernel. Hence, at its core, a shell is a program used to execute other programs on our system.

1.2. types of a shell

Different Types Of Shells

-Shells are an important part of any Linux user session. We are provided several different types of shells in Linux to accomplish tasks. Each shell has unique properties.

Hence, there are many instances where one shell is better than the other for specific requirements.
1. The Bourne Shell (sh)

-Developed at AT&T Bell Labs by Steve Bourne,
-the Bourne shell is regarded as the first UNIX shell ever.
-It is denoted as sh.
-It gained popularity due to its compact nature and high speeds of operation. Also, unlike most different types of shells in Linux, the Bourne shell cannot recall previously used commands.

It also lacks comprehensive features to offer a proper interactive use.

2. The GNU Bourne-Again Shell (bash)

-More popularly known as the Bash shell,
-the GNU Bourne-Again shell was designed to be compatible with the Bourne shell.
-It incorporates useful features from different types of shells in Linux such as Korn shell and C shell.

3. The C Shell (csh)

-The C shell was created at the University of California by Bill Joy. It is denoted as csh.
-It was developed to include useful programming features like in-built support for arithmetic operations and a syntax similar to the C programming language.

4. The Korn Shell (ksh)

The Korn shell was developed at AT&T Bell Labs by David Korn, to improve the Bourne shell. It is denoted as ksh. The Korn shell is essentially a superset of the Bourne shell.

Besides supporting everything that would be supported by the Bourne shell, it provides users with new functionalities. It allows in-built support for arithmetic operations while offereing interactive features which are similar to the C shell.

The Korn shell runs scripts made for the Bourne shell, while offering string, array and function manipulation similar to the C programming language. It also supports scripts which were written for the C shell. Further, it is faster than most different types of shells in Linux, including the C shell.

5. The Z Shell (zsh)

The Z Shell or zsh is a sh shell extension with tons of improvements for customization. If you want a modern shell that has all the features a much more, the zsh shell is what you're looking for.

echo "$SHELL" – Print the shell for the current user

2.1. What is a terminal emulator?

Terminal :

-A terminal is a text input and output environment. A terminal window, also known as a terminal emulator, is a text-only window that emulates a console in a graphical user interface (GUI).

- A terminal is a program that allows you to run a shell. Is a device with a human-readable display that accepts a stream of characters and displays them. Only chars are making their way there.

2.2. Ways to open a terminal

Ctrl+Alt+T
Ctrl+Alt+N

3.1. The command line

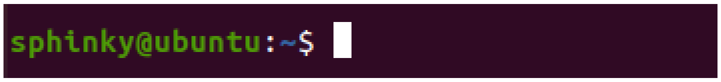
-A command line is an area to the right of the command prompt on an all-text display mode computer monitor where a user enters commands and data.

-This is the line where you type commands.

3.2. The command line prompt syntax

The username of the current logged in user.

Defines that you are connected to the machine that has the name after it



```
sphinky@ubuntu:~$
```

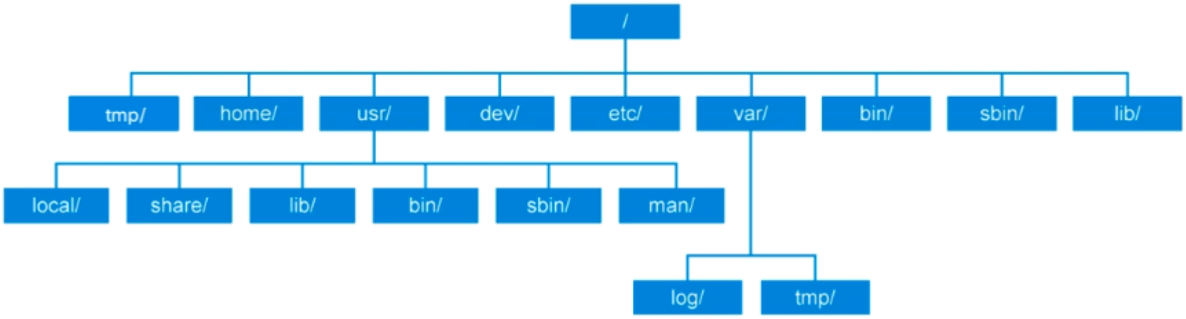
- sphinky: The username of the current logged in user.
- @: Defines that you are connected to the machine that has the name after it
- ubuntu: The name of the computer running (Name of the host)
- ~: refers to the directoy that the terminal is working in right now.
- $: States that you are logged in as a regular user.

```USERNAME@HOSTNAME:WORKING_DIRECTORY($/#)```

Just like programming languages, the Linux shell has specific syntax that you have to use. Just so that it could be understood by the shell

5. Linux Filesystem Hierarchy

- top-level Linux directories and their purposes.



- '/' – Root: The root filesystem is the top-level directory of the filesystem. It must contain all of the files required to boot the Linux system and all of the required executables and libraries required to boot the remaining filesystems (filesystem controls how data is stored and retrieved).

- /bin – User Binaries: Contains binary executables and Commands used by all the users of the system.

-/dev – Device Files: Contains device files like terminal devices, usb, or any device attached to the system (used when needed to access some device).

-/etc – Configuration File: contain core system files like startup and shutdown shell scripts.

/home – Home Directories: Home directories for all users to store their personal files.

-/lib – System Libraries: Contains library files that supports the binaries located under /bin and /sbin.

-/sbin – System Binaries: contains binary executable and linux commands.

- /tmp – Temporary Files: Directory that contains temporary files created by system and users, files under this directory are deleted when system is rebooted.

-/usr – User Programs: Contains binaries, libraries, documentation, and source-code for second level programs (user space).
/usr/bin contains binary files for user programs. /usr/sbin contains binary files for system administrators. /usr/lib contains libraries for /usr/bin and /usr/sbin. /usr/local contains users programs that you install from source.

-/var – Variable Files: Content of the files that are expected to grow can be found under this directory.

5.2. Navigating through the filesystem

The tilde (~) symbol stands for your home directory. If you are user, then the tilde (~) stands for /home/user

Cd The cd ("change directory") command is used to change the current working directory in Linux and other Unix-like operating systems

cd [OPTIONS] directory

To navigate into the root directory, use "cd /"

To navigate to your home directory, use "cd" or "cd ~"

To navigate up one directory level, use "cd .."

To navigate to the previous directory (or back), use "cd -"

Pwd

The pwd command will allow you to know in which directory you're located (pwd stands for "print working directory"). Example: "pwd" in the Desktop directory will show "~/Desktop". Note that the GNOME Terminal also displays this information in the title bar of its window. A useful gnemonic is "present working directory."

Ls

The ls command will show you ('list') the files in your current directory. Used with certain options, you can see sizes of files, when files were made, and permissions of files. Example: "ls ~" will show you the files that are in your home directory.