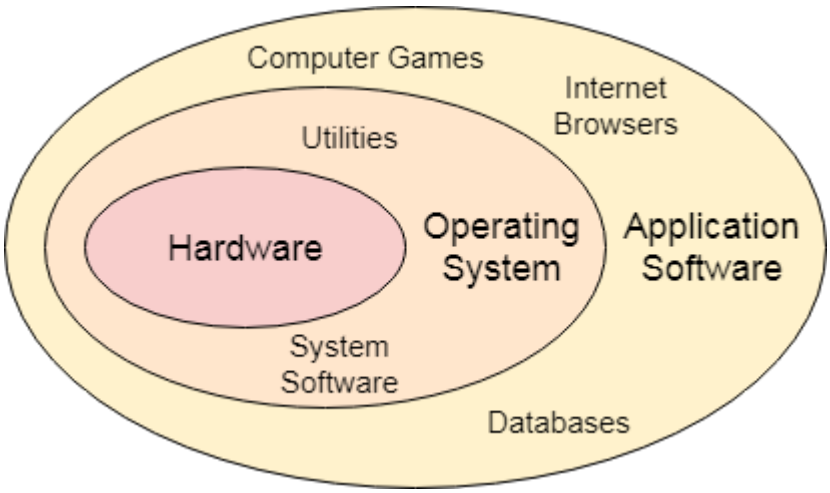
Module 1: Operating system Overview

- 1.1 Introduction, Objectives, Functions and Evolution of Operating System
- 1.2 Operating system structures: Layered, Monolithic and Microkernel
- 1.3 Linux Kernel, Shell and System Calls

What is an operating System?

In the Computer System (comprises of Hardware and software), Hardware can only understand machine code (in the form of 0 and 1) which doesn't make any sense to a naive user.

We need a system which can act as an intermediary and manage all the processes and resources present in the system.



An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks.

The purpose of an operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.

The operating system is a system program that serves as an interface between the computing system and the end-user. Operating systems create an environment where the user can run any programs or communicate with software or applications in a comfortable and well-organized way.

Furthermore, an operating is a software program that manages and controls the execution of application programs, software resources and computer hardware. It also helps manage the software/hardware resource, such as file management, memory management, input/ output and many peripheral devices like a disk drive, printers, etc.

- ***Explain different objectives of operating System?***

There are three objectives of operating system:

A) **Convenience**: An OS makes a computer more convenient to use.
B) **Efficiency**: An OS allows the computer system resources to be used in an efficient manner.
C) **Ability to evolve**: An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions at the same time without interfering with service.

- ***Explain different Functions of operating System?***

The most important functions of operating system are:

A) Process Management: The operating system is responsible for the following activities in connection with the process management

- Scheduling processes and threads on the CPU
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanism for process synchronization
- Providing mechanisms for process communication

B) Memory Management: The operating system is responsible for the following activities in connection with the memory management

- Keeping track of which parts of memory is currently being used and by whom
- Deciding which processes and data to move into and out of memory
- Allocating and de allocating memory space as needed.

C) File Management: The operating system is responsible for the following activities in connection with the file management

- Creating and deleting files
- Creating and deleting directories to organize files

- Supporting primitives for manipulating files and directories
- Mapping files onto secondary storage
- Backing up the files on non-volatile storage media

D) Device Management: The operating system is responsible for the following activities in connection with the device management

- Keeps tracks of all devices connected to system.
- designates a program responsible for every device known as the Input/Output controller
- Decides which process gets access to a certain device and for how long.
- Allocates devices in an effective and efficient way.
- De allocates devices when they are no longer required.

E) Security: The operating system uses password protection to protect user data and similar other techniques. it also prevents unauthorized access to programs and user data.

F) Booting of computer: The operating system boots the computer.

G) Control over system performance –

Monitors overall system health to help improve performance.

- Records the response time between service requests and system response to have a complete view of the system health.

- *Explain different types of operating System?*

A) Simple Batch Systems

- In this type of system, there is **no direct interaction between user and the computer**.
- The user has to submit a job (written on cards or tape) to a computer operator.
- Then computer operator places a batch of several jobs on an input device.
- Jobs are batched together by type of languages and requirement.
- Then a special program, the monitor, manages the execution of each program in the batch.
- The monitor is always in the main memory and available for execution.

B) Multiprogramming Operating System

- Multiprogramming is an extension to the batch processing where the CPU is kept always busy.
- Each process needs two types of system time: CPU time and IO time.
- In multiprogramming environment, for the time a process does its I/O, The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.

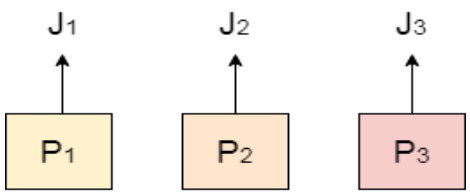
- In Non-multiprogrammed system, there are moments when CPU sits idle and does not do any work.
- In Multiprogramming system, CPU will never be idle and keeps on processing.

Time Sharing Systems are very similar to Multiprogramming batch systems. In fact time sharing systems are an extension of multiprogramming systems.

In Time sharing systems the prime focus is on **minimizing the response time**, while in multiprogramming the prime focus is to maximize the CPU usage.

C) Multiprocessing Operating System

- In Multiprocessing, Parallel computing is achieved. There are more than one processors present in the system which can execute more than one process at the same time. This will increase the throughput of the system.



Multi Processing

D) Real Time Operating System

- In Real Time systems, each job carries a certain deadline within which the Job is supposed to be completed, otherwise the huge loss will be there or even if the result is produced then it will be completely useless.
- The Application of a Real Time system exists in the case of military applications, if you want to drop a missile then the missile is supposed to be dropped with certain precision.

E) Distributed Operating System

- The motivation behind developing distributed operating systems is the availability of powerful and inexpensive microprocessors and advances in communication technology.
- These advancements in technology have made it possible to design and develop distributed systems comprising of many computers that are inter connected by communication networks. The main benefit of distributed systems is its low price/performance ratio.

F) Clustered Systems

- Like parallel systems, clustered systems gather together multiple CPUs to accomplish computational work.
- Clustered systems differ from parallel systems, however, in that they are composed of two or more individual systems coupled together.
- The definition of the term clustered is **not concrete;** the general accepted definition is that clustered computers share storage and are closely linked via LAN networking.
- Clustering is usually performed to provide **high availability**.

G) Handheld Systems

- Handheld systems include **Personal Digital Assistants(PDAs)**, such as Cellular Telephones with connectivity to a network such as the Internet. They are usually of limited size due to which most handheld devices have a small amount of memory, include slow processors, and feature small display screens.

- ***Explain services provided by operating System?***

An Operating System supplies different kinds of services to both the users and to the programs as well. It also provides application programs (that run within an Operating system) an environment to execute it freely.

Here is a list of common services offered by an almost all operating systems:

A) **User Interface**Usually Operating system comes in three forms or types. Depending on the interface their types have been further subdivided. These are:

- Command line interface
- Batch based interface
- Graphical User Interface

The command line interface (CLI) usually deals with using text commands and a technique for entering those commands. The batch interface (BI): commands and directives are used to manage those commands that are entered into files and those files get executed. Another type is the graphical user interface (GUI): which is a window system with a pointing device (like mouse or trackball) to point to the I/O, choose from menus driven interface and to make choices viewing from a number of lists and a keyboard to entry the texts.

B) **Program Execution**

The operating system must have the capability to load a program into memory and execute that program. Furthermore, the program must be able to end its execution, either normally or abnormally / forcefully.

C) **File system manipulation**

Programs need has to be read and then write them as files and directories. File handling portion of operating system also allows users to create and delete files by specific name along with extension, search for a given file and / or list file information. Some programs comprise of permissions management for allowing or denying access to files or directories based on file ownership.

D) **Input / Output Operations**

A program which is currently executing may require I/O, which may involve file or other I/O device. For efficiency and protection, users cannot directly govern the I/O devices. So, the OS provide a means to do I/O Input / Output operation which means read or write operation with any file.

E) Communication

Process needs to swap over information with other process. Processes executing on same computer system or on different computer systems can communicate using operating system support. Communication between two processes can be done using shared memory or via message passing.

F) Resource Allocation

When multiple jobs running concurrently, resources must need to be allocated to each of them. Resources can be CPU cycles, main memory storage, file storage and I/O devices. CPU scheduling routines are used here to establish how best the CPU can be used.

G) Error Detection

Errors may occur within CPU, memory hardware, I/O devices and in the user program. For each type of error, the OS takes adequate action for ensuring correct and consistent computing.

H) Accounting

This service of the operating system keeps track of which users are using how much and what kinds of computer resources have been used for accounting or simply to accumulate usage statistics.

I) Security and protection

Protection includes in ensuring all access to system resources in a controlled manner. For making a system secure, the user needs to authenticate him or her to the system before using (usually via login ID and password).

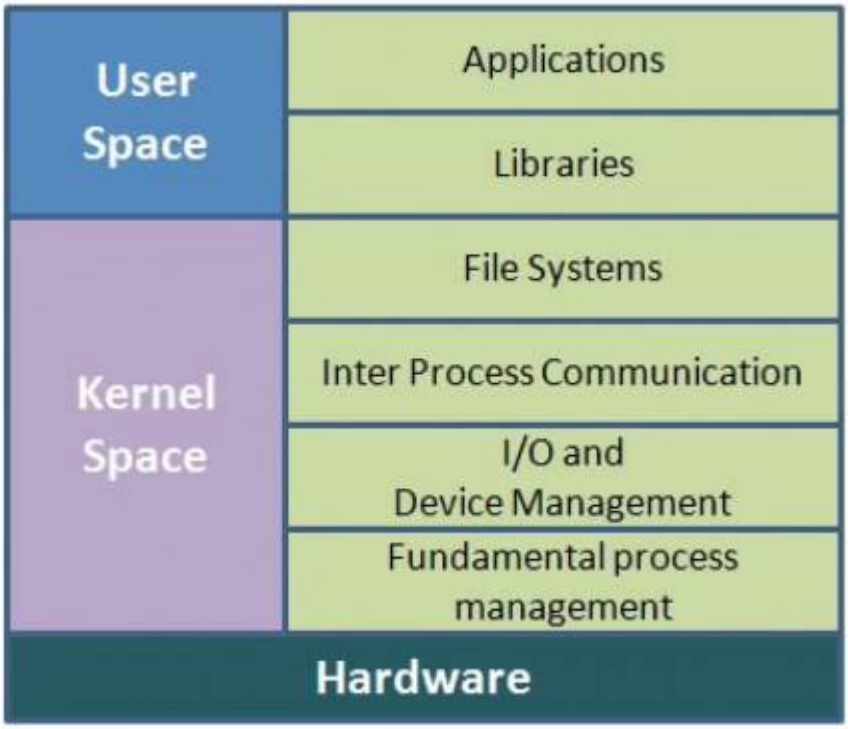
Explain the operating System Structure

The operating system structure are categorized as-

A)Monolithic Approach

The monolithic operating system is also known as the monolithic kernel. This is an old type of operating system. They were used to perform small tasks like batch processing, time sharing tasks in banks. Monolithic kernel acts as a virtual machine which controls all hardware parts. It is different than microkernel which has limited tasks.
A **microkernel** is divided into two parts i.e. kernel space and user space. Both these parts communicate with each other through IPC (Inter-process communication). Microkernel advantage is that if one server fails then other server takes control of it.
Operating systems which use monolithic architecture were first time used in the 1970's.

The monolithic operating system is a very basic operating system in which file management, memory management, device management, and process management is directly controlled within the kernel. All these components like file management, memory management etc. are located within the kernel.



Advantages of Monolithic Kernel –

- One of the major advantage of having monolithic kernel is that it provides CPU scheduling, memory management, file management and other operating system functions through system calls.
- The other one is that it is a single large process running entirely in a single address space.
- It is a single static binary file. Example of some Monolithic Kernel based OSs are: Unix, Linux, Open VMS, XTS-400, z/TPF.

Disadvantages of Monolithic Kernel –

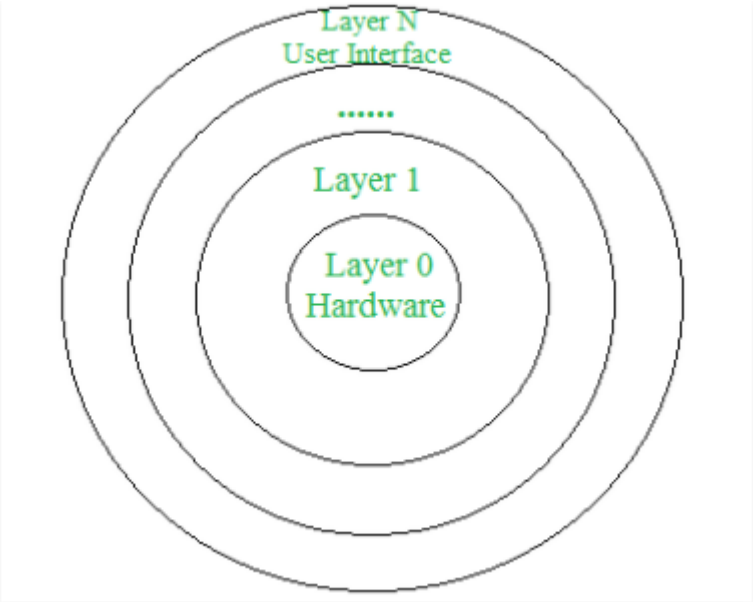
- One of the major disadvantage of monolithic kernel is that, if anyone service fails it leads to entire system failure.
- If user has to add any new service. User needs to modify entire operating system.

B) Layered Approach

An OS can be broken into pieces and retain much more control on system. In this structure the OS is broken into number of layers (levels). The bottom layer (layer 0) is the hardware and the topmost layer (layer N) is the user interface. These layers are so designed that each layer uses the functions of the lower level layers only. This simplifies

the debugging process as if lower level layers are debugged and an error occurs during debugging then the error must be on that layer only as the lower level layers have already been debugged.

The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Moreover careful planning of the layers is necessary as a layer can use only lower level layers. UNIX is an example of this structure.



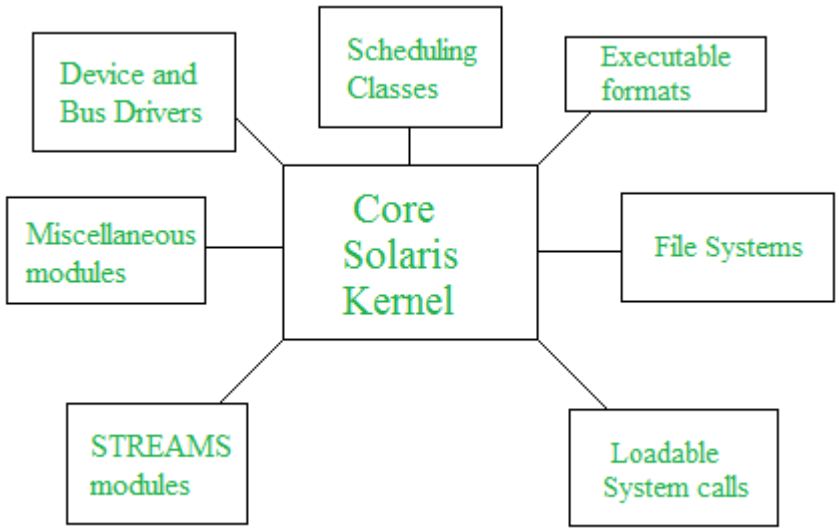
C) Microkernel approach

This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. This result in a smaller kernel called the micro-kernel.
Advantages of this structure are that all new services need to be added to user space and does not require the kernel to be modified. Thus it is more secure and reliable as if a service fails then rest of the operating system remains untouched. Mac OS is an example of this type of OS.

Modular structure or approach:
It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time. It resembles layered structure due to the fact that each kernel has defined and protected

interfaces but it is more flexible than the layered structure as a module can call any other module.

For example Solaris OS is organized as shown in the figure.



Explain the differences between monolithic and microkernel System Structure

Basis of Comparison	**Monolithic Kernel**	**MicroKernel**
Execution Style	All processes are executed under the kernel space in privileged mode.	Only the most important processes take place in the Kernel space. All other processes are executed in the user space.
Size	Kernel size is bigger when compared to Microkernel.	Kernel size is smaller with respect to the monolithic kernel.
Speed	It provides faster execution of processes.	Process execution is slower.

Stability	A single process crash will cause the entire system to crash.	A single process crash will have no impact on other processes.
Inter-Process Communication	Use signals and sockets to achieve interprocess communication.	Use messaging queues to achieve inter process communication.
Extensibility	Difficult to extend.	Easily extensible.
Maintainability	Maintenance is more time and resource consuming.	Easily maintainable
Debug	Harder to debug	Easier to debug
Security	Less Secure.	More Secure
Example	Linux	Mac OS

System Calls

To understand system calls, first one needs to understand the difference between **kernel mode** and **user mode** of a CPU. Every modern operating system supports these two modes.

Modes supported by the operating system

Kernel Mode

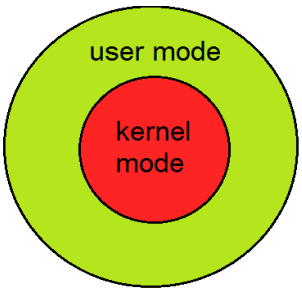
- When CPU is in **kernel mode**, the code being executed can access any memory address and any hardware resource.
- Hence kernel mode is a very privileged and powerful mode.

- If a program crashes in kernel mode, the entire system will be halted.

User Mode

- When CPU is in **user mode**, the programs don't have direct access to memory and hardware resources.
- In user mode, if any program crashes, only that particular program is halted.
- That means the system will be in a safe state even if a program in user mode crashes.
- Hence, most programs in an OS run in user mode.

When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource. This is done via something called a **system call**.



When a program makes a system call, the mode is switched from user mode to kernel mode. This is called a **context switch**.

Then the kernel provides the resource which the program requested. After that, another context switch happens which results in change of mode from kernel mode back to user mode.

A **system call** is a mechanism that provides the interface between a process and the operating system. It is a programmatic method in which a computer program requests a service from the kernel of the OS.

System call offers the services of the operating system to the user programs via API (Application Programming Interface). System calls are the only entry points for the kernel system.

Why do you need System Calls in OS?

Following are situations which need system calls in OS:

- Reading and writing from files demand system calls.
- If a file system wants to create or delete files, system calls are required.
- System calls are used for the creation and management of new processes.
- Network connections need system calls for sending and receiving packets.
- Access to hardware devices like scanner, printer, need a system call.

Types of System calls

Here are the five types of system calls used in OS:

- Process Control
- File Management
- Device Management
- Information Maintenance
- Communications

Process Control
This system calls perform the task of process creation, process termination, etc.

Functions:

- End and Abort
- Load and Execute
- Create Process and Terminate Process
- Wait and Signed Event
- Allocate and free memory

File Management
File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc.

Functions:

- Create a file
- Delete file
- Open and close file
- Read, write, and reposition
- Get and set file attributes

Device Management

Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc.

Functions

- Request and release device
- Logically attach/ detach devices
- Get and Set device attributes

Information Maintenance
It handles information and its transfer between the OS and the user program.

Functions:

- Get or set time and date
- Get process and device attributes

Communication:
These types of system calls are specially used for interprocess communications.

Functions:

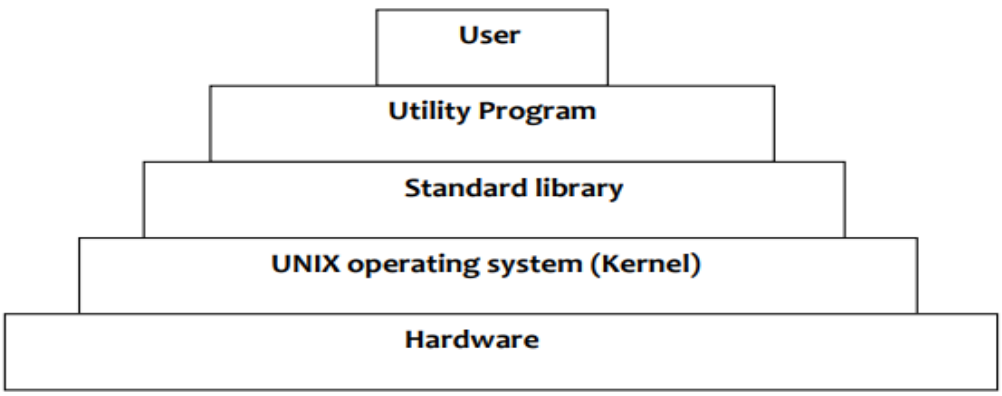
- Create, delete communications connections
- Send, receive message
- Help OS to transfer status information
- Attach or detach remote devices

Examples of Windows and Unix System Calls –

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Linux Operating system

LINUX is an operating system or a kernel distributed under an open-source license. Its functionality list is quite like UNIX. The kernel is a program at the heart of the Linux operating system that takes care of fundamental stuff, like letting hardware communicate with software.



Linux architecture

Linux Kernel

The main purpose of a computer is to run a *predefined sequence of instructions*, known as a **program**. A program under execution is often referred to as a **process**. Now, most special purpose computers are meant to run a single process, but in a sophisticated system such a general purpose computer, are intended to run many processes simultaneously. Any kind of process requires hardware resources such are Memory, Processor time, Storage space, etc.

In a General Purpose Computer running many processes simultaneously, we need a middle layer to manage the distribution of the hardware resources of the computer efficiently and fairly among all the various processes running on the computer. This middle layer is referred to as the **kernel**. Basically the kernel virtualizes the common hardware resources of the computer to provide each process with its own virtual resources. This makes the process seem as it is the sole process running on the machine. The kernel is also responsible for preventing and mitigating conflicts between different processes.
This schematically represented below:

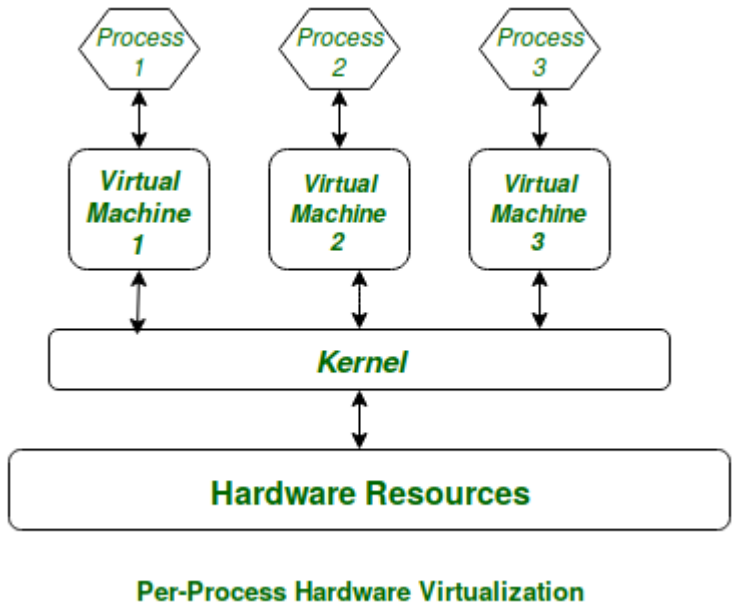


Figure: Virtual Resources for each Process
The **Core Subsystems** of the **Linux Kernel** are as follows:

1. The Process Scheduler
2. The Memory Management Unit (MMU)
3. The Virtual File System (VFS)
4. The Networking Unit
5. Inter-Process Communication Unit

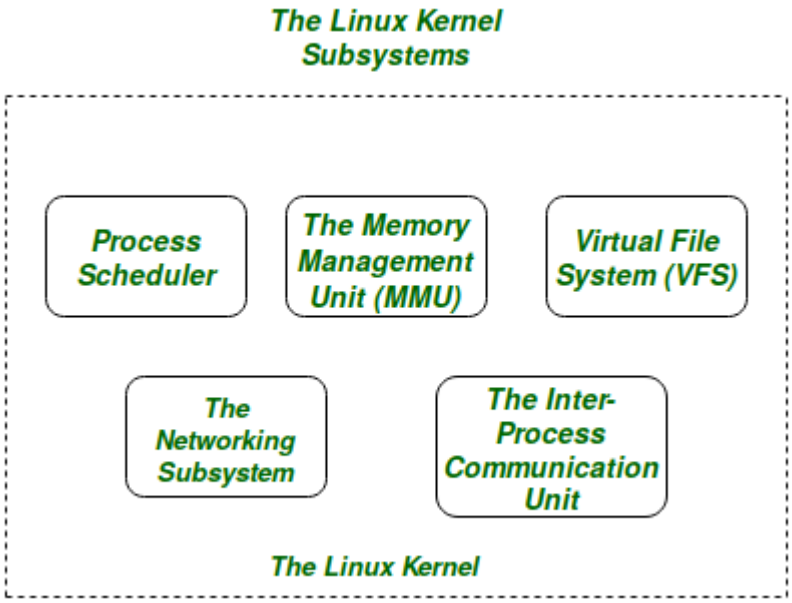


Figure: The Linux Kernel

For the purpose of this article we will only be focussing on the 1st three important subsystems of the Linux Kernel.

The basic functioning of each of the 1st three subsystems is elaborated below:

- **The Process Scheduler:**
  This kernel subsystem is responsible for fairly distributing the CPU time among all the processes running on the system simulteneously.
- **The Memory Management Unit:**
  This kernel sub-unit is responsible for proper distribution of the memory resources among the various processes running on the system. The MMU does more than just simply provide separate virtual address spaces for each of the processes.
- **The Virtual File System:**
  This subsystem is responsible for providing a unified interface to access stored data across different filesystems and physical storage media.

Write a short note on shell

SHELL is a program which provides the interface between the user and an operating system. When the user logs in OS starts a shell for user. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

Shell Prompt

The prompt, **$**, which is called the **command prompt**, is issued by the shell. While the prompt is displayed, you can type a command.

Shell reads your input after you press **Enter**. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.

Types of Shell:

- **The C Shell –**
  Denoted as **csh**
  Bill Joy created it at the University of California at Berkeley. It incorporated features such as aliases and command history. It includes helpful programming features like built-in arithmetic and C-like expression syntax.

  In C shell:

  Command full-path name is /bin/csh,

  Non-root user default prompt is hostname %,

  Root user default prompt is hostname #.

- **The Bourne Shell –**
  Denoted as **sh**
  It was written by Steve Bourne at AT&T Bell Labs. It is the original UNIX shell. It is faster and more preferred. It lacks features for interactive use like the ability to recall previous commands. It also lacks built-in arithmetic and logical expression handling. It is default shell for Solaris OS.

  For the Bourne shell the:

  Command full-path name is /bin/sh and /sbin/sh,

Non-root user default prompt is $,

Root user default prompt is #.

- **The Korn Shell**
  It is denoted as **ksh**
  It Was written by David Korn at AT&T Bell LabsIt is a superset of the Bourne shell.So it supports everything in the Bourne shell.It has interactive features. It includes features like built-in arithmetic and C-like arrays, functions, and string-manipulation facilities.It is faster than C shell. It is compatible with script written for C shell.

  For the Korn shell the:

  Command full-path name is /bin/ksh,

  Non-root user default prompt is $,

  Root user default prompt is #.

Shell Scripts

The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by # sign, describing the steps.

There are conditional tests, such as value A is greater than value B, loops allowing us to go through massive amounts of data, files to read and store data, and variables to read and store data, and the script may include functions.

Example Script

Assume we create a **test.sh** script. Note all the scripts would have the **.sh** extension. Before you add anything else to your script, you need to alert the system that a shell script is being started. This is done using the **shebang** construct. For example –

#!/bin/sh

This tells the system that the commands that follow are to be executed by the Bourne shell. *It's called a shebang because the # symbol is called a hash, and the ! symbol is called a bang*.

To create a script containing these commands, you put the shebang line first and then add the commands –

#!/bin/bash

```
pwd
ls
```

The shell script is now ready to be executed −

```
$./test.sh
```

Upon execution, you will receive the following result −

```
/home/amrood
index.htm  unix-basic_utilities.htm  unix-directories.htm
test.sh    unix-communication.htm    unix-environment.htm
```

Note − To execute a program available in the current directory, use **./program_name**