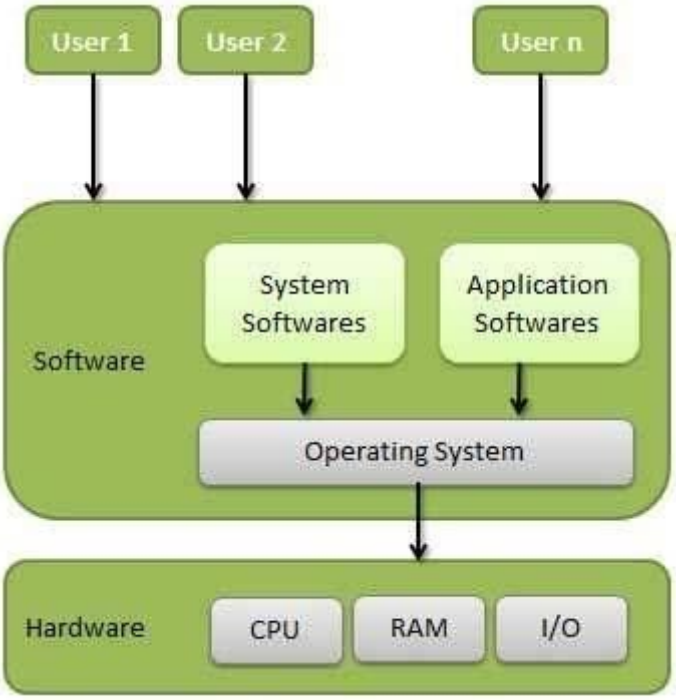
Unit-1

Introduction

Concept of Operating Systems (OS)

An operating system is essentially software that manages and co-ordinates various processes and system resources. It performs all the essential tasks like file management, memory management, process management, handling input and output and controlling peripheral devices such as disk drives and printers.



System Software is the type of software which is the interface between application software and system. Ex. Compiler

Application Software is the type of software which runs as per user request. Ex. MSWord

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Generations of OS

Following are the various generations of OS:

First Gen (1945-55): Serial Processing

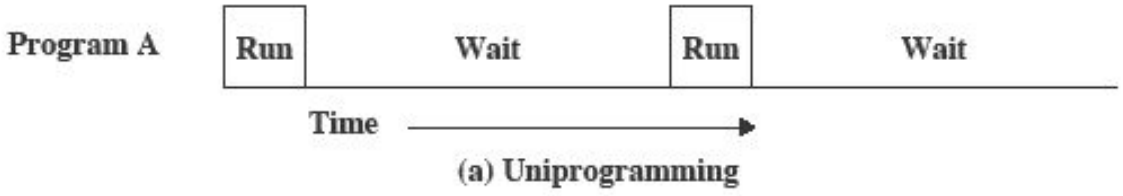
- No operating system
- Machines run from a console with display lights, toggle switches, input device, and printer
- Schedule time
- Setup included loading the compiler, source program, saving compiled program and loading and linking

Second Gen (1955-65): Batch OS

- Monitor is the central idea
- Software that controls the sequence of events
- User does not interact with the CPU directly
- Batch jobs together
- Program returns control to monitor when finished
- Contained Special type of programming language called JCL (Job Control Language) that can provide instruction to the monitor
- Included memory protection, timer, Privileged Instructions, Interrupts

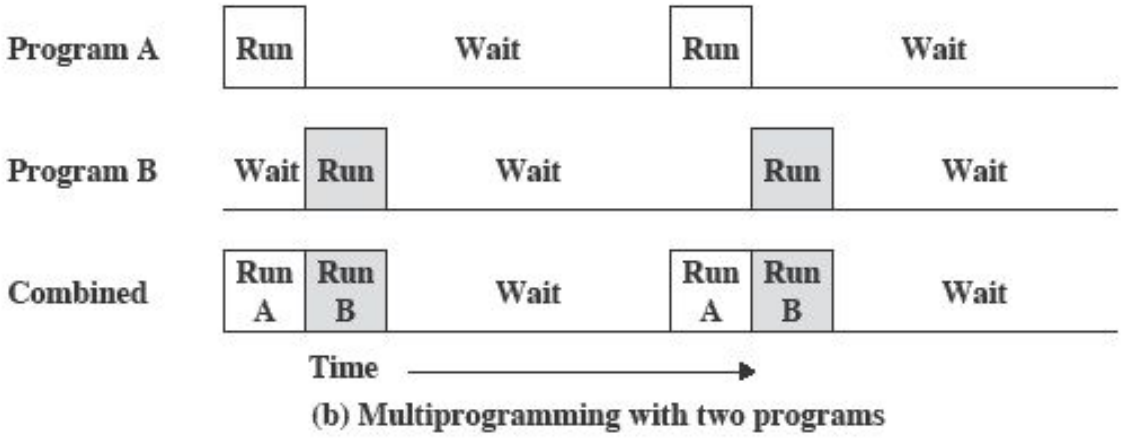
Third Gen (1965-80): Multiprogramming

- In case of Uniprogramming, Processor must wait for I/O instruction to complete before proceeding



(a) Uniprogramming

- In case of Multiprogramming, when one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

Fourth Gen (1980-current): Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is slotted and shared among multiple users
- Multiple users simultaneously access the system through terminals

Types of OS

- Batch OS
- Multiprogramming OS
- Multiprocessing Systems
- Multitasking/time sharing OS
- Distributed OS
- Clustered OS
- Real time OS

OS Services

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

1. **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

2. **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.

3. **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.

4. **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.

5. **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.

6. **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.

7. **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.

8. **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.

9. **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).

Interrupt handling and System Calls

- 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.
- **How System Call Works?**
  - **Step 1)** The processes executed in the **user mode** till the time a **system call interrupt**s it.

- **Step 2)** After that, the system call is executed in the **kernel-mode** on a priority basis.
- **Step 3)** Once system call execution is over, control returns to the user mode.
- **Step 4)** The execution of user processes resumed in Kernel mode.

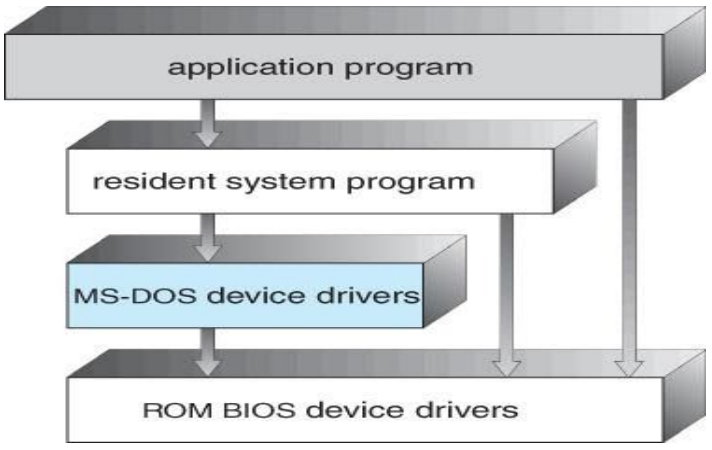
Basic architectural concepts of an OS

There are four types of OS architectures:

- Simple
- Monolithic
- Layered
- Microkernel

Simple:

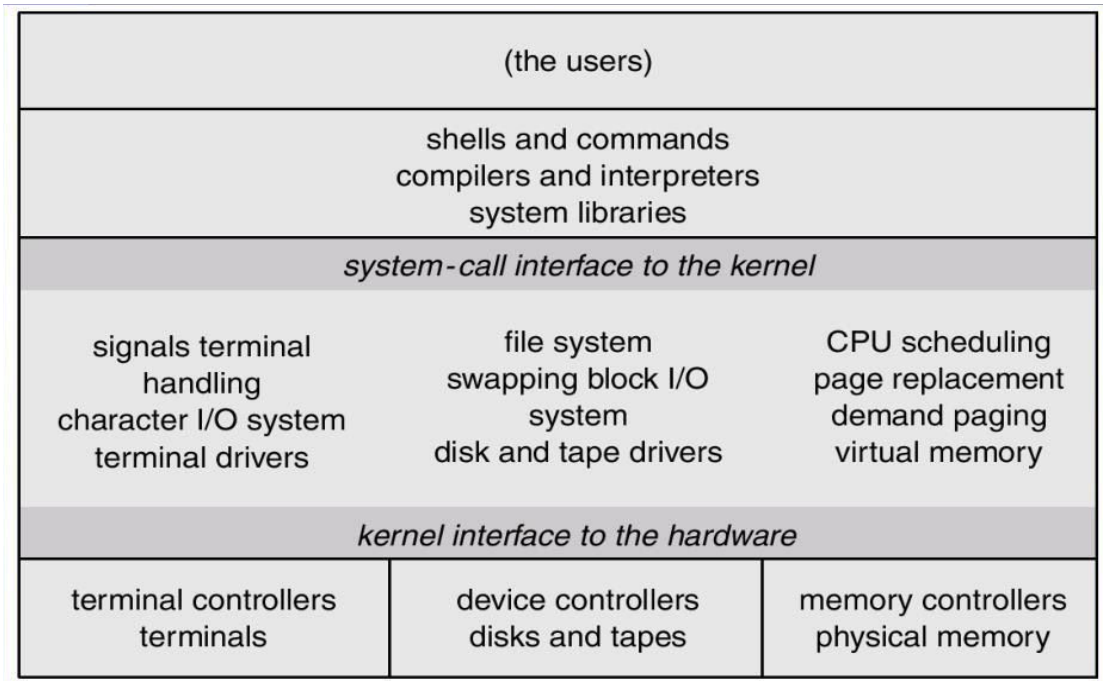
- All aspects of the OS linked together in one binary
- In MS-DOS, applications may bypass the operating system.
- Operating systems such as MS-DOS and the original UNIX did not have well-defined structures and APIs.
- Interfaces and levels of functionality not well separated
- There was no CPU Execution mode (user and kernel), and so errors in applications could cause the whole system to crash.



Monolithic:

- Monolithic OS is an OS architecture where the entire OS is working as a **one large program** in **kernel space** and is alone in **supervisory mode**.
- Every component of OS is contained in the kernel and can directly communicate with any other component (using a simple function call)

- Most of the OS functionalities like scheduling, file system, networking, device drivers, memory management etc. are taken care of by the kernel itself.
- A monolithic kernel is that of a Linux and other Unix systems.
- It has poor portability as the architecture dependent code spreads throughout the OS
- Testing and debugging is very difficult.

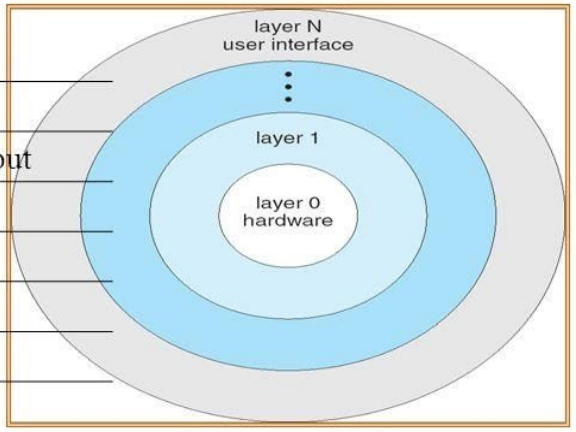


Layered:

- This approach breaks up the operating system into different layers.
- Higher layer uses the services of only the lower layers to operate.
- This allows implementers to change the inner workings of a layer without affection the other layers of the system, and increases modularity.
- With the layered approach, the bottom layer is the hardware, while the highest layer is the user interface.
  - The main *advantage* is simplicity of construction and debugging.
  - The main *difficulty* is defining the functionality of various layers and separating them.
  - The main *disadvantage* is that the OS tends to be less efficient than other implementations as each layer adds some overhead.
- Example: The Microsoft Windows NT Operating System.

Its six layers are as follows:

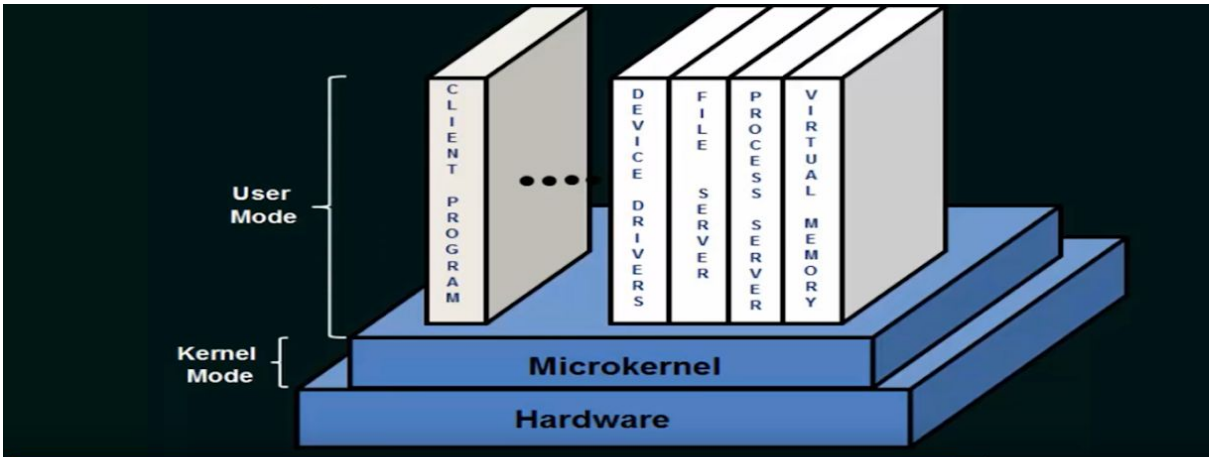
layer 5:	user programs
layer 4:	buffering for input and output
layer 3:	Process management
layer 2:	memory management
layer 1:	CPU scheduling
layer 0:	hardware



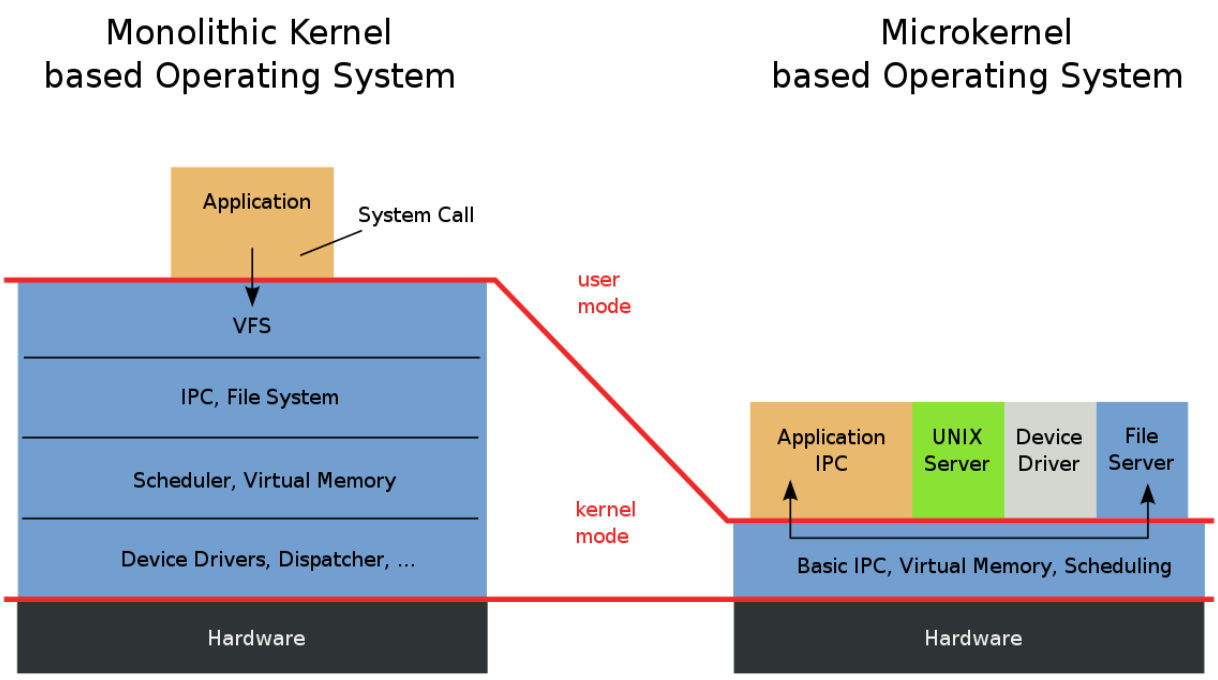
NOTE: In the fig. above, six layers have been shown but there can be any number of layers. Placement of layers and their number are strictly OS dependent.

Microkernel:

- Assigns **only a few essential functions to the kernel** like basic scheduling, Inter process communication, address space utilization.
- This structure removes all nonessential portions of the kernel and implementing them as system and user level programs, essentially called **servers**.
- Servers may be customized
- Communication between components of the OS is provided by message passing.
- The *benefits* of the microkernel are as follows:
  - Extending the operating system becomes much easier.
  - Any changes to the kernel tend to be fewer, since the kernel is smaller.
  - The microkernel also provides more security and reliability.
- Main *disadvantage* is poor performance due to increased system overhead from message passing.
- A well known microkernel system is **Mach**, which was developed at Carnegie Mellon University in the mid-1980. Mach was used as the low-level part of Apple OS X.



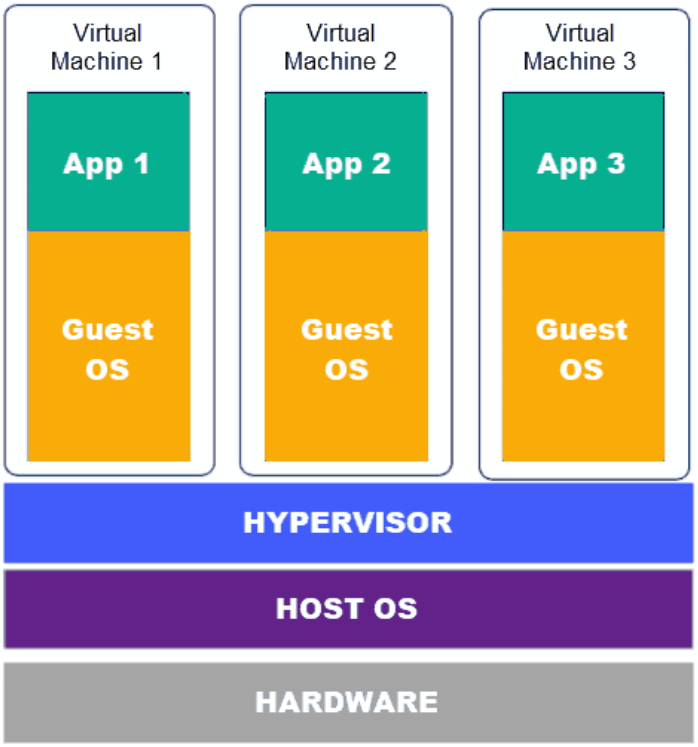
Monolithic vs. Microkernel:



Concept of Virtual Machine

- A virtual machine (VM) is a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface and storage, created on a physical hardware system (located off- or on-premises).
- **Hypervisor** also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs).
- It separates the machine's resources from the hardware and provisions them appropriately so they can be used by the VM.
- The physical machines, equipped with a hypervisor such as Kernel-based Virtual Machine (KVM), is called the host machine, host computer, host operating system, or simply *host*.
- The many VMs that use its resources are guest machines, guest computers, guest operating systems, or simply *guests*.
- The hypervisor treats compute resources—like CPU, memory, and storage—as a pool of resources that can easily be relocated between existing guests or to new virtual machines.
- VMs are isolated from the rest of the system and multiple VMs can exist on a single piece of hardware, like a server. They can be moved between host servers depending on demand or to use resources more efficiently.

- VMs allow multiple different operating systems to run simultaneously on a single computer.
- Each operating system runs in the same way an operating system or application normally would on the host hardware, so the end user experience emulated within the VM is nearly identical to a real-time operating system experience running on a physical machine.



Views of an OS

Any OS can be viewed from three different perspectives:

- Resource Manager View
- Process View; and
- Hierarchical View

Resource Manager View: The OS can be viewed as a Resource Manager

- Manages all resources (hardware, CPU time, memory space, File storage space, I/O devices etc.) fairly between various jobs.
- Decides between conflicting requests for efficient and fair resource use

Process View: The OS can be viewed as a Control Program

- Controls execution of programs to prevent errors and improper use of the computer

Hierarchical View: Modern day OS can be viewed as a Hierarchical structure with the layered approach, the bottom layer is the hardware, while the highest layer is the user interface.

- The main *advantage* is simplicity of construction and debugging.
- The main *difficulty* is defining the various layers.
- The main *disadvantage* is that the OS tends to be less efficient than other implementations.