Operating System

Bachelors in Computer Engineering

Chapter one

Type and Structure of Operating System

OUTLINES:

1.1. Introduction and history of operating system

1.2. Operating system concepts and functionalities

- 1.2.1 Processes
- 1.2.2 Files
- 1.2.3 System Calls
- 1.2.4 The Shell

1.3. Operating System Structure

- 1.3.1. Monolithic Systems
- 1.3.2. Layered
- 1.3.3. Virtual Machines
- 1.3.4. Client-Server

1.4 Types and Evolution of operating system

Operating system

- Computer Software can roughly be divided into two types:
  a) Application Software: Which perform the actual work the user wants.
  b) System Software: Which manage the operation of the computer itself.
- The most fundamental system software is the operating system, whose job is to control all the computer's resources and provide a base upon which the application program can be written.
- Operating system acts as an intermediary between a user of a computer and the computer hardware.

- **Importance of operating systems:** Facilitates communication between hardware and software, manages resources efficiently, provides a user-friendly interface.

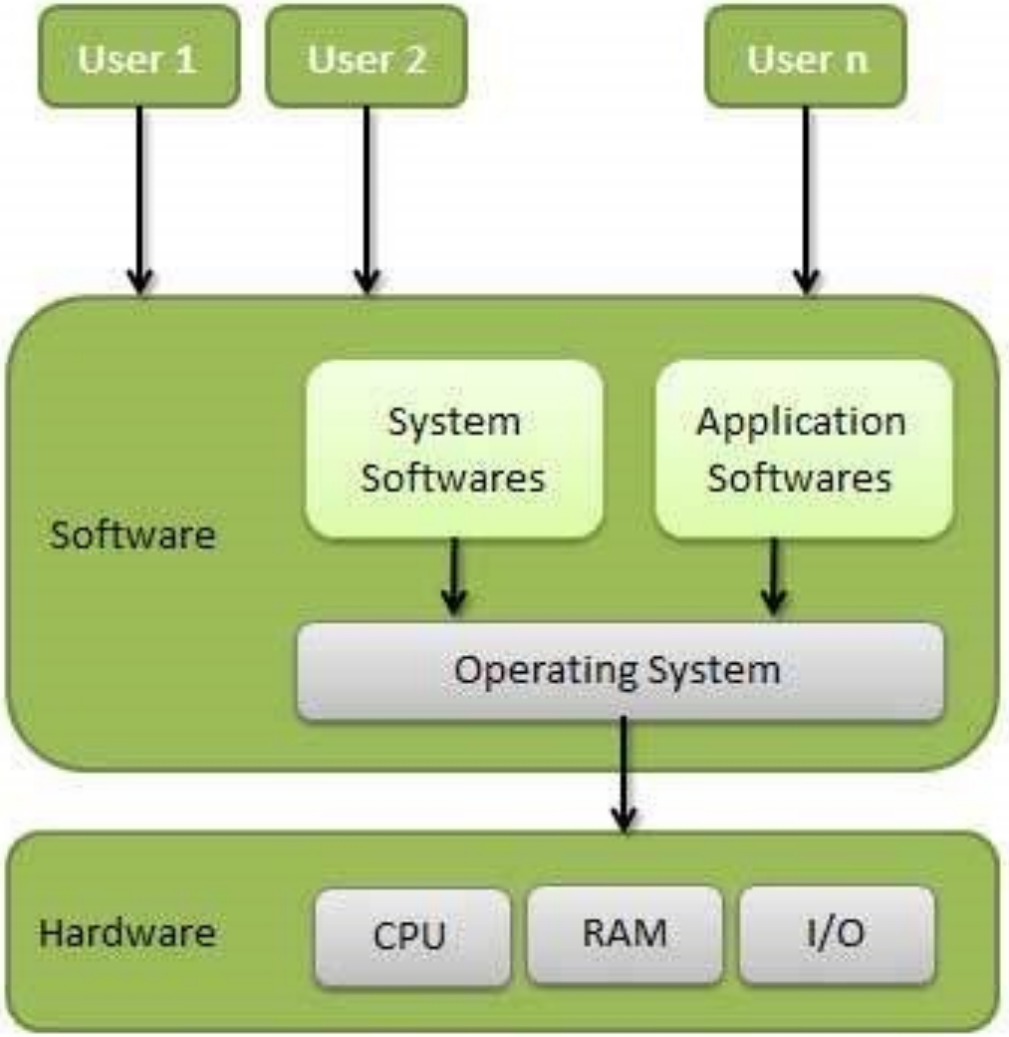


Figure: Operating system

Operating system as a Resource Manager

- A computer has many resources(Hardware and Software), which may be required to complete a task. The commonly required resources are Input/Output devices, Memory file storage space, CPU(Central Processing Unit) time and so on.
- An operating system (OS) functions as a resource manager, efficiently allocating and managing the available resources of a computer system while maintaining system stability and reliability.
- Eg: When a number of computers are connected through a network with more than one computer trying for a computer print or a common resource, then the operating system follows the same order and manages the resources in an efficient manner.
- The OS employs scheduling algorithms to allocate CPU time, manages memory by providing virtual memory and memory allocation techniques, controls storage by organizing file systems and handling disk operations, and facilitates communication between devices and software components through device drivers and input/output operations.
- By effectively managing these resources, the OS optimizes system performance, enhances user experience, and enables multiple applications to run concurrently and smoothly on a single computer system.

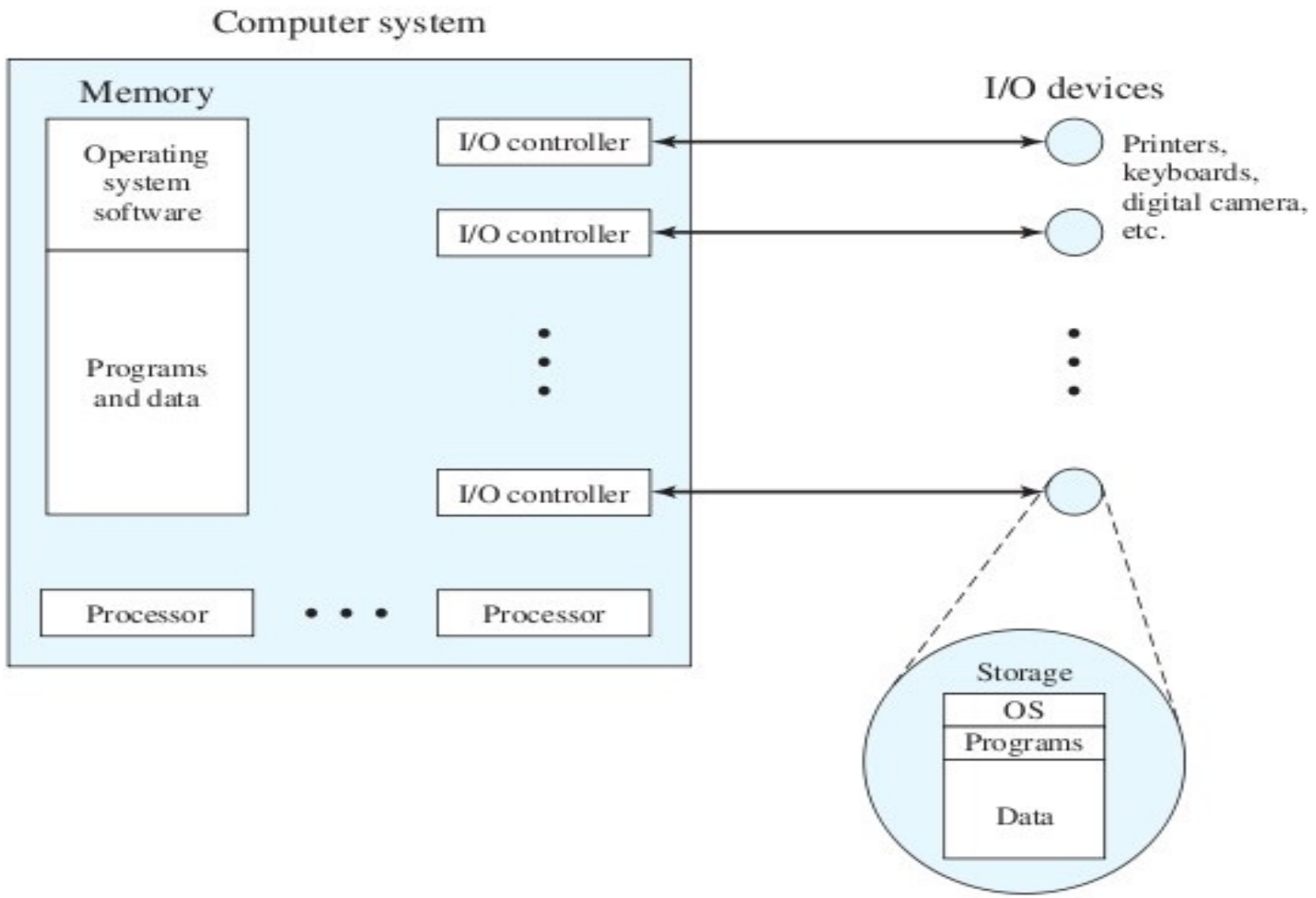


Figure: OS as a resource manager

A portion of the OS is in main memory. This includes Kernel or nucleus. The remainder of main memory contains user programs and data. The allocation of this resource (main memory) is controlled jointly by the OS and memory management hardware in the processor.

Operating system as an extented machine or virtual machine

- The operating system masks or hides the details of the Hardware form the programmers and general users and provides a convenient interface for using the system.
- The program that hides the truth about the hardware from the user and presents a nice simple view of named files that can be read and written is of course the operating system.
- In this view the function of OS is to present the user with the equivalent of an extended machine or virtual machine that is easier to program than underlying hardware.
- Just as the operating system shields the user from the disk hardware and presents a simple file-oriented interface, it also conceals a lot of unpleasant business concerning interrupts, timers, memory management and other low level features.
- A major function of OS is to hide all the complexity presented by the underlying hardware and gives the programmer a more convenient set of instructions to work with.

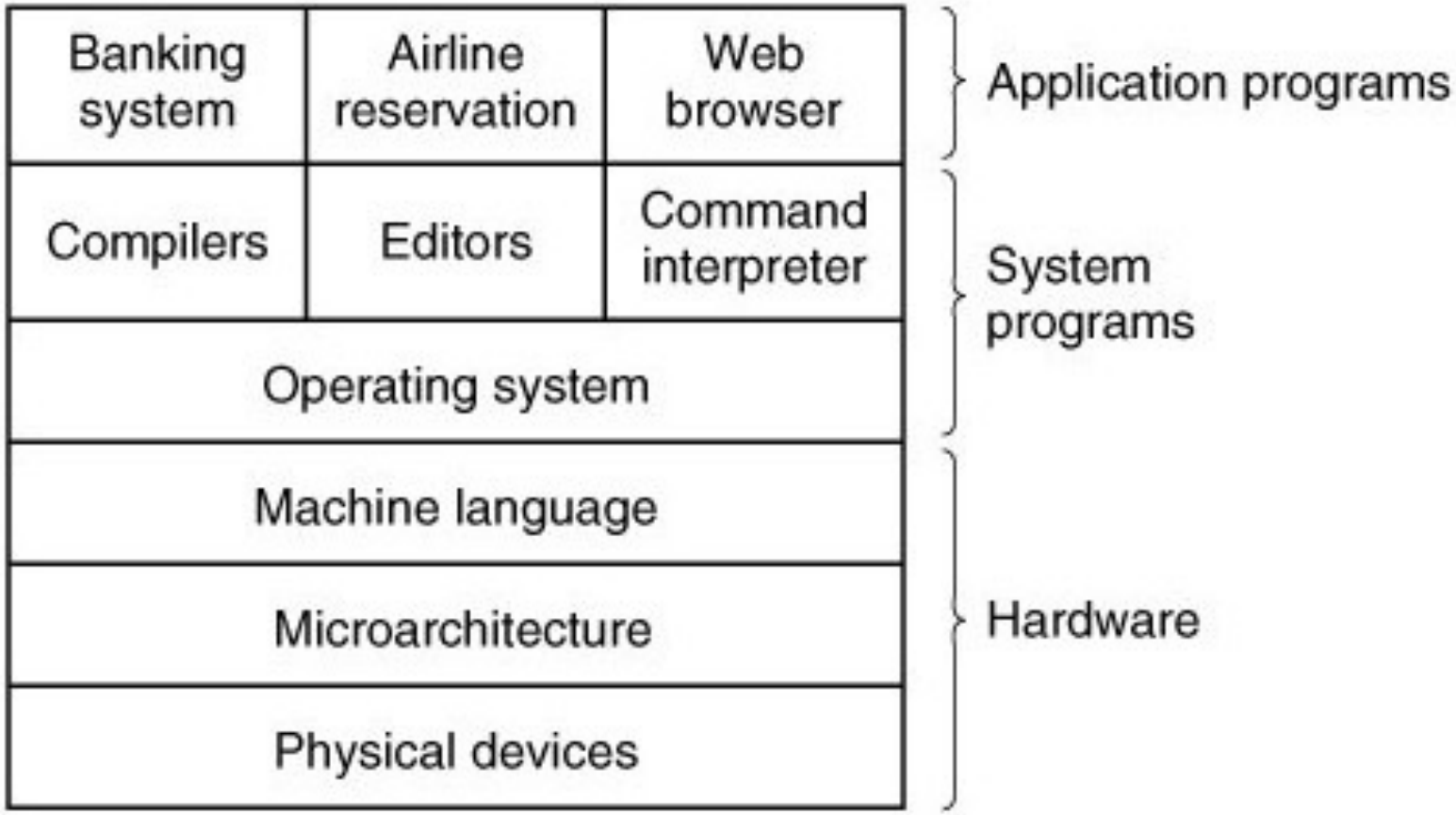
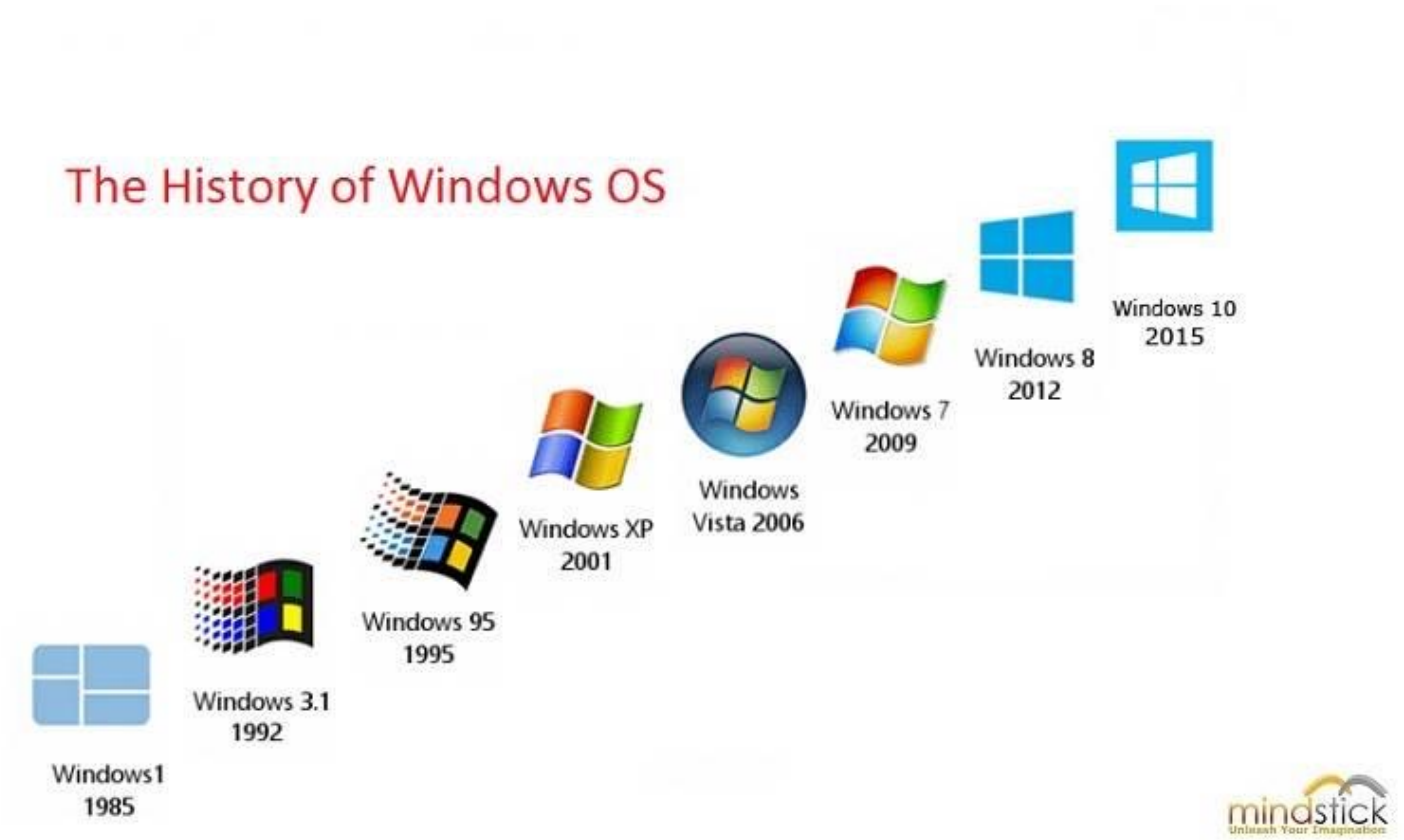


Figure: placement of OS

History of OS



Following are some of of the major milestones in the evolution of operating systems:

1. 1940s-1950s: The first electronic computers, such as the ENIAC and UNIVAC I, did not have operating systems. Users directly interacted with the machine through console switches and patch panels to load programs manually.

2. 1950s-1960s: As computers became more complex and multiprogramming emerged, the need for efficient resource management led to the development of batch processing systems. Examples include the IBM 7090's IBSYS and the Compatible Time-Sharing System (CTSS) at MIT, which allowed multiple users to share the computer's resources.

3. 1960s-1970s: The introduction of time-sharing systems marked a significant milestone. IBM's OS/360, released in 1964, was a notable example. It supported multiprogramming, allowing multiple users to run programs simultaneously. Other important systems from this era include Multics and Unix, which introduced features like hierarchical file systems, process management, and inter-process communication.

4. 1980s: Personal computers became popular, and operating systems like MS-DOS (Microsoft Disk Operating System) and Apple's Macintosh System Software gained prominence. These operating systems provided graphical user interfaces (GUIs) and simplified user interaction.

5. Late 1990s-early 2000s: The rise of the internet led to the development of network-centric operating systems. Windows 2000 and later Windows versions incorporated networking features, while Linux further expanded its support for networking and internet technologies.

6. 2010s-present: Mobile operating systems, such as Android and iOS, gained prominence with the advent of smartphones and tablets. These systems introduced touch-based interfaces, app stores, and advanced power management features.

Additionally, there have been advancements in real-time operating systems (RTOS) for embedded systems, server operating systems for data centers, and virtualization technologies that enable multiple operating systems to run concurrently on a single machine.

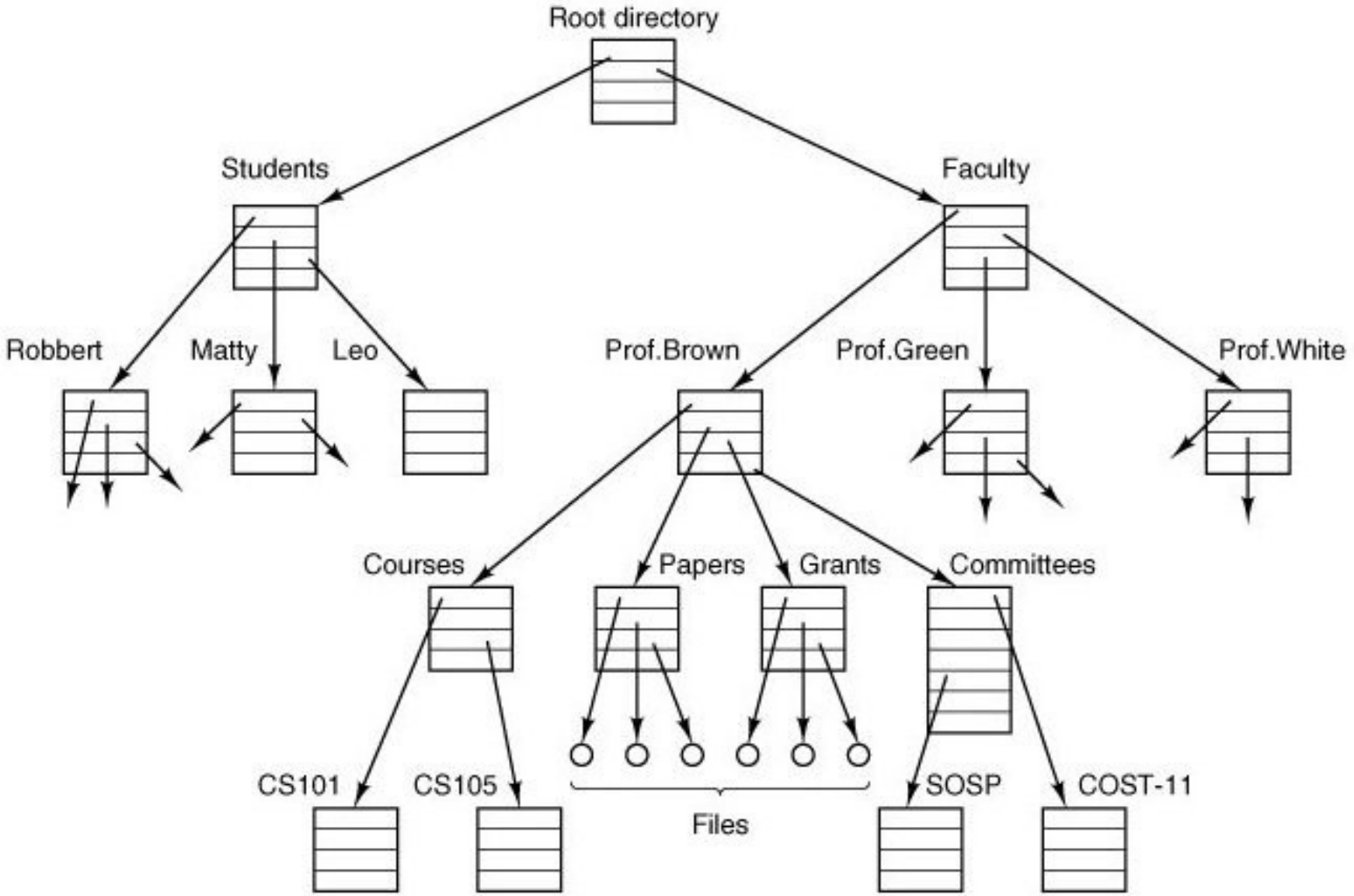
Overall, the history of operating systems showcases the continuous evolution and adaptation to new technologies and user needs, shaping the way we interact with computers and enabling the efficient utilization of hardware resources.

Function of Operating system:

1. Memory management function
2. processors management function
3. I/O Device management function
4. File management function

Files

- A major function of the operating system is to hide the pecularities of the disks and other I/O devices and present the programmer with a nice, clean abstract model of device independent files.
- System calls are obviously needed to create files, remove files, read files and write files. Before a file can be read, it must be opened, and after it has been read it should be closed, so calls are provided to do these things.
- To provide a place to keep files, most operating system has the concept of a directory as a. way of grouping files together.
- A student, for example, might have one directory for each course he is taking , another directory for his electronic mail, and still another directory for his www home page. System calls are then needed to create and remove directories.
- Calls are provided to put an existing file into a directory, and to remove a file from a directory. Directory entries may be either files or other directories. This model also give rise to a hierarchy of the file system.


Root directory
Students
Faculty
Robbert
Matty
Leo
Prof.Brown
Prof.Green
Prof.White
Courses
Papers
Grants
Committees
CS101
CS105
Files
SOSP
COST-11

Every file within the directory hierarchy can be specified by giving its path name from top of the directory hierarchy, the root directory. Such absolute path names consists of the list of directories that must be traversed from the root directory. Such absolute path names consists of the list of directories that must be traversed from the root directory to get to the file, with slashes separating the components.

For example: file path for CS101 in figure can be written as \Faculty\Prof.Brown\Courses\CS101.

System Call:

- A system call is a mechanism that allows a program to request services from the operating system's kernel.

- These services can include hardware-related operations (e.g., disk access), process management (creating and executing processes), and communication with kernel services (e.g., scheduling).

- System calls serve as the interface between a running process and the underlying operating system.

- In Unix, Unix-like, and POSIX-compatible operating systems, common system calls include open, read, write, close, wait, execve, fork, exit, and kill.

- Modern operating systems may have hundreds of system calls, with Linux having over 300 different ones.

- System calls can be roughly grouped into five major categories:

1. Process Control.
    - Load
    - execute
    - create process
    - terminate process
    - get/set process attributes
    - wait for time, wait event, signal event
    - allocate, free memory

2. File management.

- create file, delete file
- open, close
- read, write, reposition
- get/set file attributes

3. Device Management.

- request device, release device
- read, write, reposition
- get/set device attributes
- logically attach or detach devices

4. Information Maintenance.

- get/set time or date
- get/set system data
- get/set process, file, or device attributes

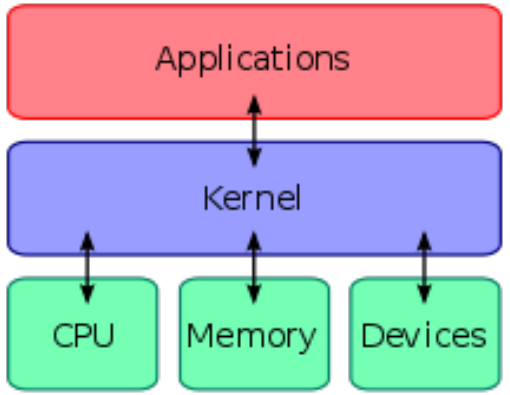
5. Communication.

- create, delete communication connection
- send, receive messages
- transfer status information
- attach or detach remote devices

Shell:

- A shell is a program that provides the traditional text only user interface for Linux and other Unix operating system.
- Its primary function is to read commands typed into a console or terminal window and then execute it.
- The term shell derives its name form the fact that it is an outer layer of the OS.
- A shell is an interface between the user and the internal part of the operating system.
- A user is in shell(i.e interacting with the shell) as soon as the user has logged into the system. A shell is the most fundamental way that user can interact with the system and the shell hides the detail of the underlying system from the user.
  Example:
- Bourne Shell
- Bash shell
- Korn Shell
- C shell

Kernel:



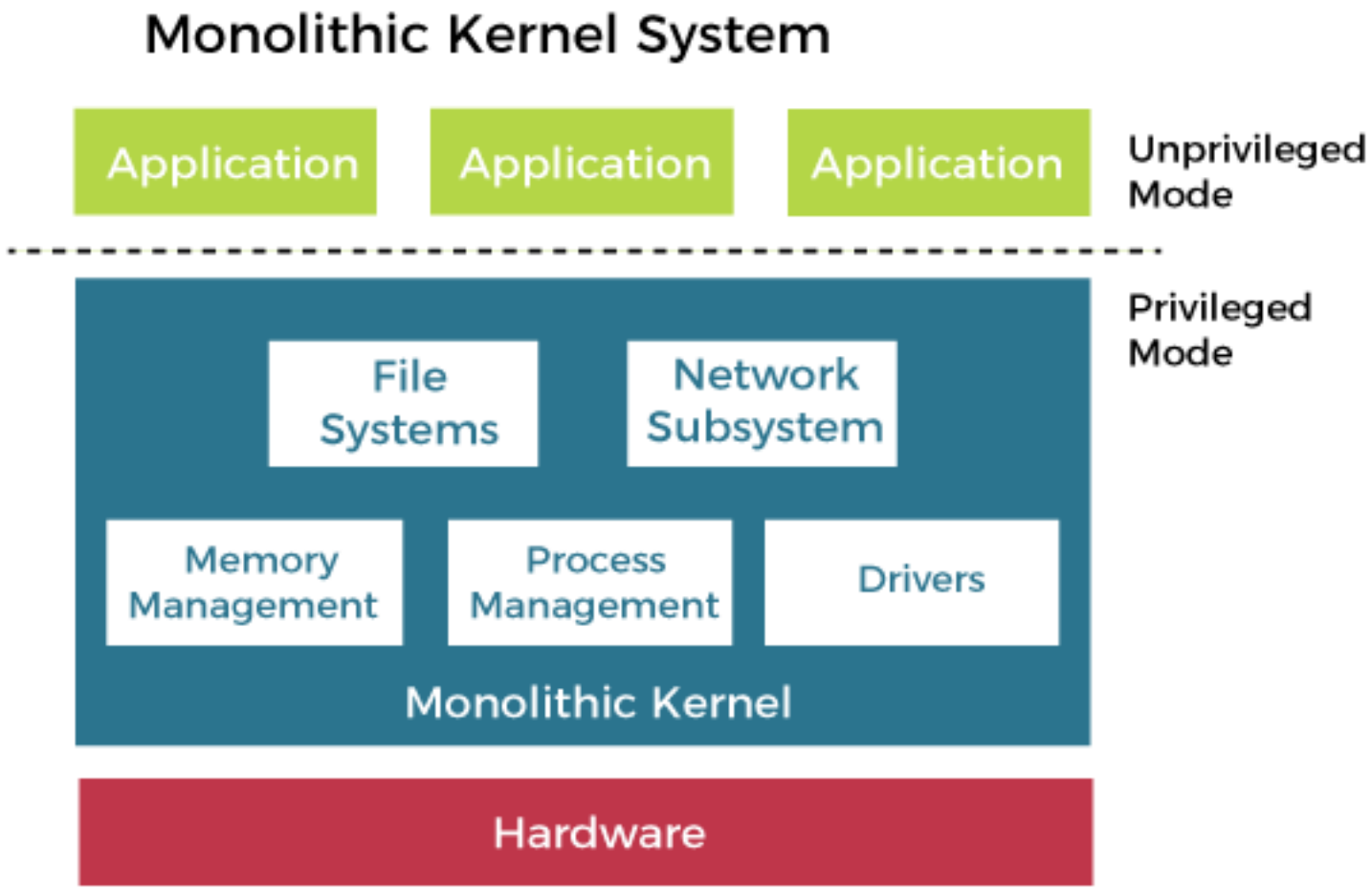
- The main component of most computer operating systems
- A bridge between applications and the actual data processing done at the hardware level.
- The kernel's responsibilities include managing the system's resources (the communication between hardware and software components).
- A kernel can provide the lowest-level abstraction layer for the resources (especially processors and I/O devices) that application software must control to perform its function.
- It typically makes these facilities available to application processes through inter-process communication mechanisms and system calls.
- Operating system tasks are done differently by different kernels, depending on their design and implementation.
  - While monolithic kernels execute all the operating system code in the same address space to increase the performance of the system,
  - microkernels run most of the operating system services in user space as servers, aiming to improve maintainability and modularity of the operating system.

Operating System Structure:

- The structure of an operating system is dictated by the model employed in building them.
- An operating system model is a broad framework that unifies the many features and services the operating system provides and tasks it performs.
- Operating systems are broadly classified into following categories, based on the their structuring mechanism as follows:
  a. Monolithic System
  b. Layered System
  c. Virtual Machine
  d. Exokernels
  e. Client-Server Model

Monolithic System

- This is a type of operating system architecture in which the entire operating system works in the kernel space.
- All the basic services of OS like process management, file management, memory management, exception handling, process communication etc. are all present inside the kernel only.
- This monolithic model differs from the other operating system architectures like micro lithic as this provides the virtual interface alone over the computer hardware which makes it more useful.
- The operating system is written as a collection of procedures that are linked together into a single large executable program. Each procedure in the system is free to call any other process. Calling any procedure makes the system very efficient
- In this structure, there is no chance of information hiding. Every procedure is visible to every other procedure.
- Traditionally been used by Unix-like operating system


Monolithic Kernel System
Application
Application
Application
Unprivileged Mode
Privileged Mode
File Systems
Network Subsystem
Memory Management
Process Management
Drivers
Monolithic Kernel
Hardware

Advantages

- The execution of this architecture is so fast.
- All the memory management, file management and process scheduling is performed under one space.
- The process runs under single address space.

Disadvantages

- If any service fails the entire system is failed.
- For adding any type of new service it must be modified by the user.

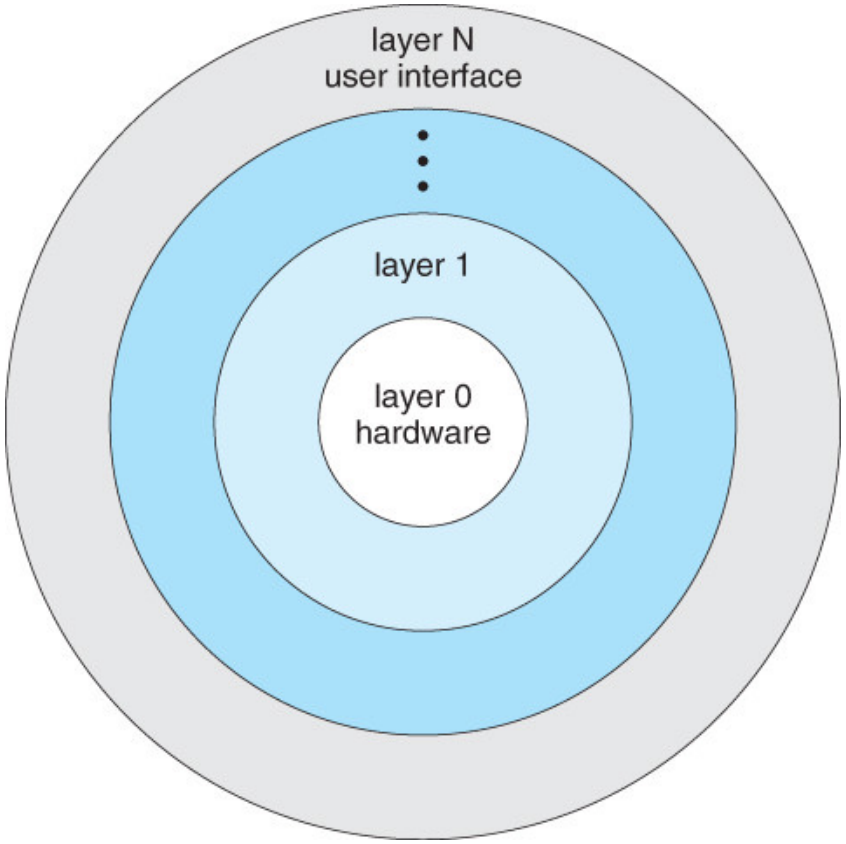
Layered Operating System

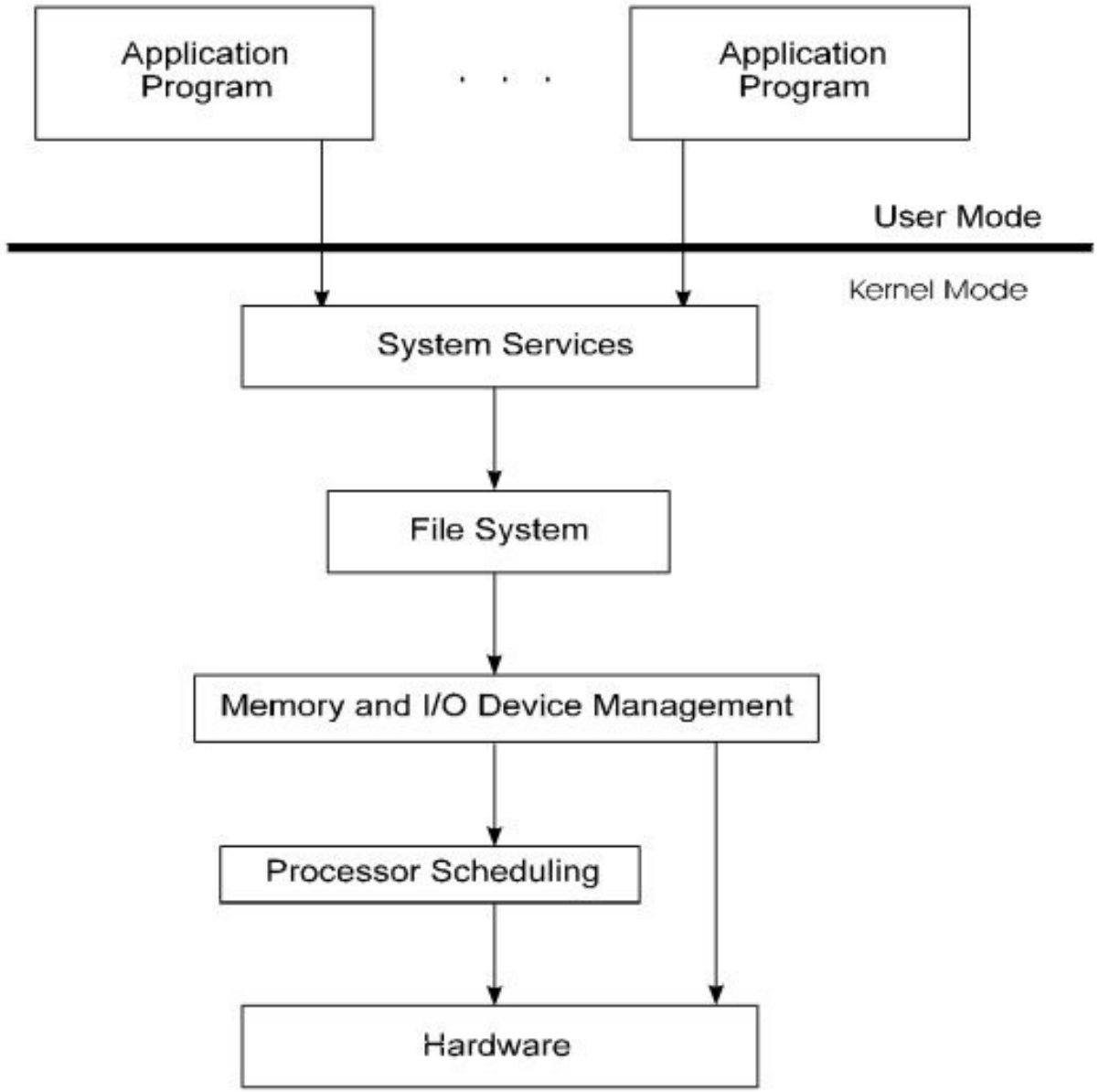
- An operating system architecture that divides software components into layers, with hardware at the bottom of each layer. Each layer of the operating system is responsible for certain functions.
- It was created to improve the pre-existing structures like the Monolithic structure (UNIX) and the Simple structure (MS-DOS).
- The layered structure approach divides the operating system into layers and gives the user much more control over the system.
- The hardware is on the bottom layer (layer 0), and the user interface is on the top layer (layer N).
- The layers sandwiched between the first and last layer are responsible for system services and management such as CPU scheduling, memory management, process management and I/O Management.
- These layers are designed in such a way that each layer only uses the functions of the lower-level layers.
- It makes debugging easier if lower-level layers are debugged and an error occurs while debugging. Because the lower-level layers have already been debugged, the error must be on that layer only.

- The layered operating system divides the operating system into numerous levels, each with its own set of capabilities.

  Layer 0 is responsible for allocating processes and switching between them when interruptions or timers occur. It also covers the basics of CPU multiprogramming.

  As a result, if the user layer wants to engage with the hardware layer, the response will pass through all layers from n-1 to 1. Each layer must be planned and implemented to only require services from the layers below it.




Application Program
Application Program
User Mode
Kernel Mode
System Services
File System
Memory and I/O Device Management
Processor Scheduling
Hardware

Advantages of Layered Structure

- **Modularity:** This architecture promotes modularity because each layer only does its assigned duties.
- **Easy debugging:** It is relatively simple to debug because the layers are discrete. If a mistake happens in the CPU scheduling layer, the developer can only debug that layer, as opposed to a Monolithic system where all services are present at the same time.
- **Easy update:** A change made to one layer will have no effect on the other layers.
- **No direct access to hardware:** The hardware layer is the design's innermost layer. So, unlike the Simple system, where the user has direct access to the hardware, a user can use hardware services but not directly modify or access it.
- **Abstraction:** Every layer is focused on its own set of tasks. As a result, the other layers' functions and implementations are abstract.

Disadvantages of Layered Structure

- **Complex and careful implementation:** Because a layer can use the services of the levels below it, the layers must be carefully arranged. The memory management layer, for example, is used by the backup storage layer. As a result, it must be placed behind the memory management layer. As a result, significant modularity leads to complicated implementation.

- **Slower in execution:** When a layer wishes to communicate with another layer, it sends a request that must travel through the layers between the two layers to be fulfilled. As a result, unlike the Monolithic system, which is faster, it increases response time. As an output, increasing the number of layers may result in an inefficient design.

Virtual Machines:

- In a virtual machine architecture, an abstraction layer called a virtual machine monitor (VMM) or hypervisor sits between the hardware and the operating system.
- Multiple operating systems, known as virtual machines, run on top of the hypervisor.
- A virtual machine (VM) is a virtual environment which functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system.
- VMs are isolated from the rest of the system, and multiple VMs can exist on a single piece of hardware, like a server. That means, it as a simulated image of application software and operating system which is executed on a host computer or a server.
- It has its own operating system and software that will facilitate the resources to virtual computers.
- CPU scheduling can be used to share the CPU and to create the appearance that users have their own processors.

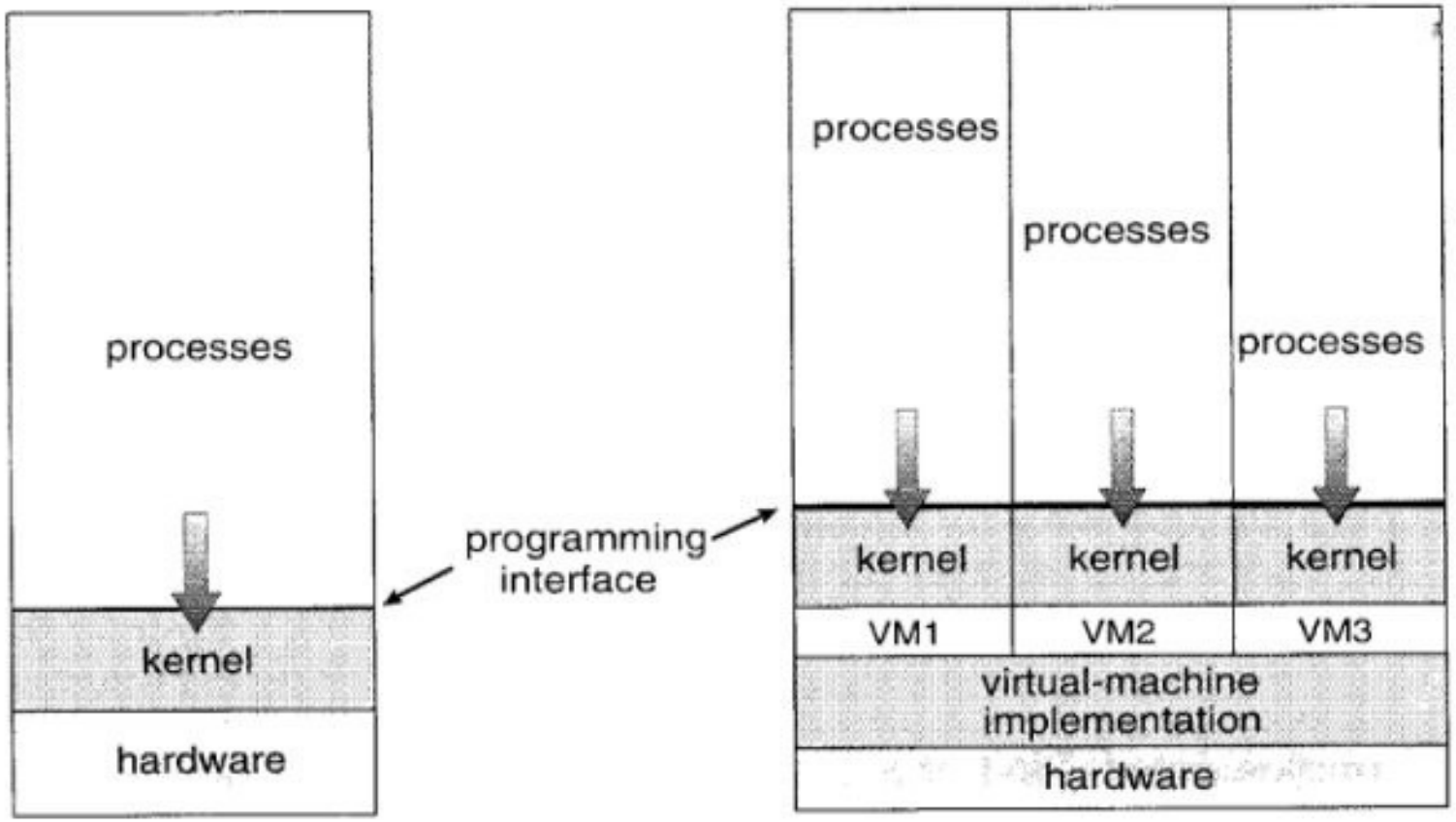


Fig: a). Non Virtual Machine b). Virtual Machine.

Although the virtual machine concept is useful, it is difficult to implement. Much effort is required to provide an exact duplicate of the underlying machine.

Advantages

The benefits of virtual machines (VMs) arise from their ability to create isolated and independent operating system environments within a single physical machine.

- The multiple Operating system environments exist simultaneously on the same machine, which is isolated from each other.
- Virtual machine offers an instruction set architecture which differs from real computer.
- Using virtual machines, there is easy maintenance, application provisioning, availability and convenient recovery.

Client-Server or Microkernel

- The advent of new concepts in operating system design, microkernel, is aimed at migrating traditional services of an operating system out of the monolithic kernel into the user-level process.
- The idea is to divide the operating system into several processes, each of which implements a single set of services - for example, I/O servers, memory server, process server, threads interface system.
- Each server runs in user mode, provides services to the requested client. The client, which can be either another operating system component or application program, requests a service by sending a message to the server.
- An OS kernel (or microkernel) running in kernel mode delivers the message to the appropriate server; the server performs the operation; and microkernel delivers the results to the client in another message.

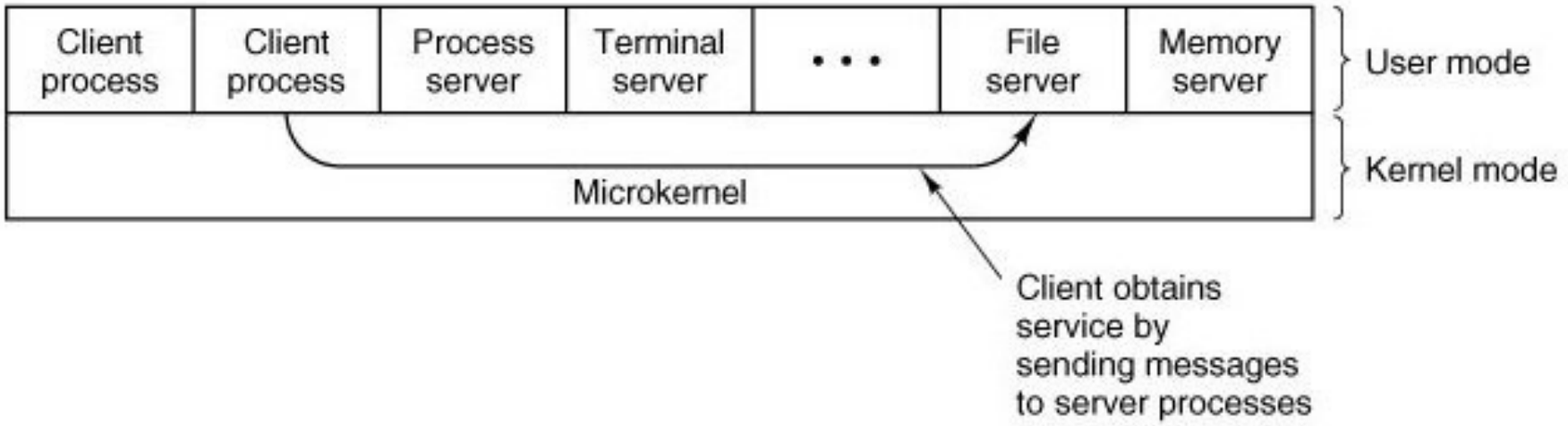


Fig: The client server model

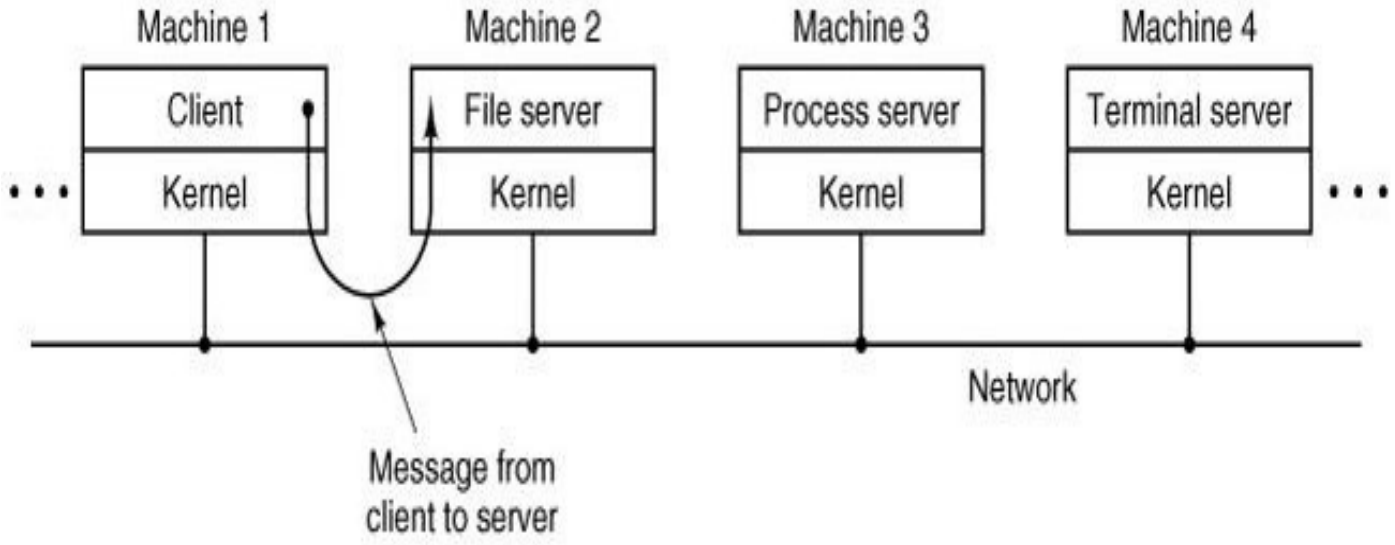


Fig: The client server model in distribiuted system

Advantages:

- **Modularity:** The microkernel design promotes modularity, making it easier to add or replace components without affecting the entire system.
- **Isolation:** Since most services run in user mode, a failure in one service doesn't necessarily affect others.
- **Flexibility:** Components can be developed independently, promoting flexibility in system design

Types and Evolution of Operating System:

Evolution of Operating Systems:

There are various stages in the evolution of operating systems, starting from early systems and progressing to more modern ones:

1. Serial processing
2. Batch processing
3. Multiprogramming
4. Multitasking or time sharing System
5. Network Operating system
6. Distributed Operating system
7. Multiprocessor Operating System
8. Real Time Operating System

1. Serial Processing:

- Early computer from late 1940 to the mid 1950.
- The programmer interacted directly with the computer hardware.
- These machine are called bare machine as they don't have OS.
- Every computer system is programmed in its machine language.
- Uses Punch Card, paper tapes and language translator

These system presented two major problems.

1. Scheduling
2. Set up time

Each of these steps involves the mounting or dismounting tapes on setting up punch cards. If an error occur user had to go the beginning of the set up sequence. Thus, a considerable amount of time is spent in setting up the program to run.

This mode of operation is turned as serial processing ,reflecting the fact that users access the computer in series.

Scheduling:

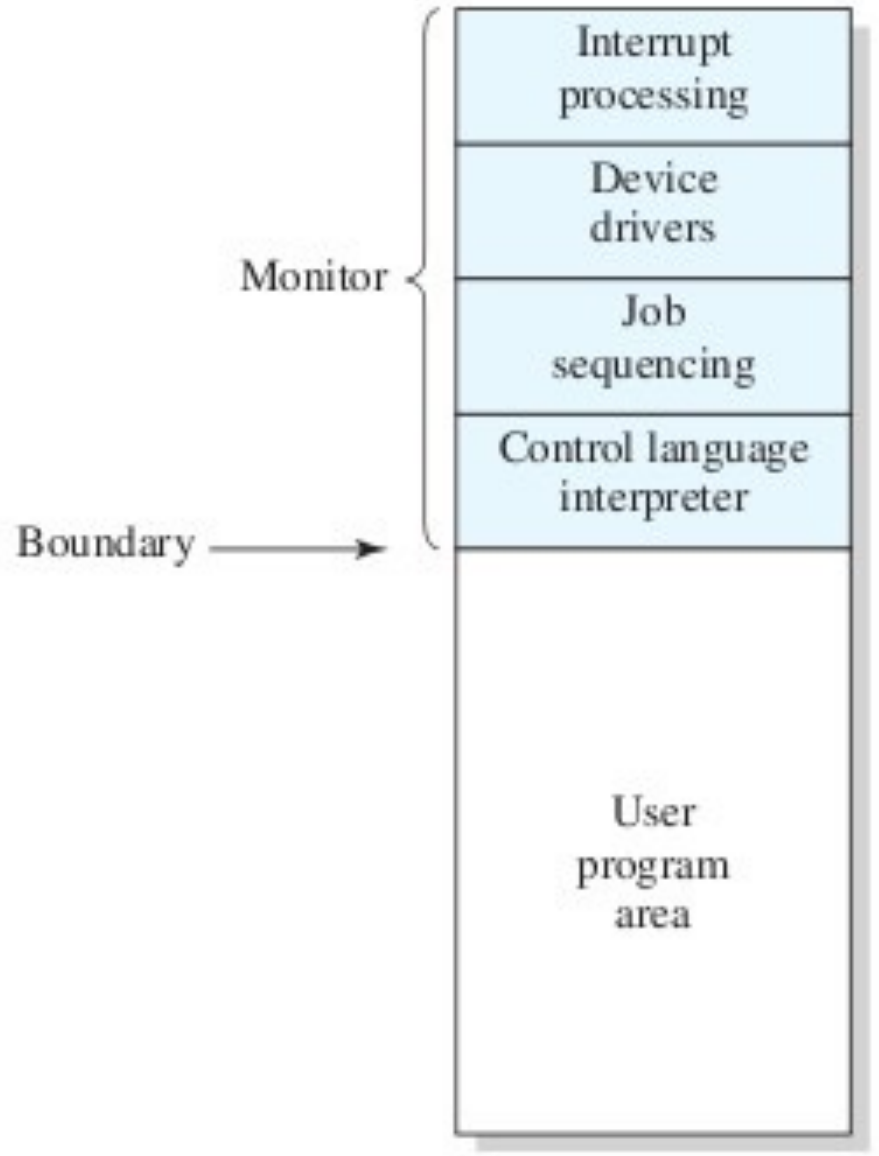
- Scheduling in early systems was a manual process.
- Programmers or operators had to meticulously plan and manage the sequence in which programs were executed.
- This manual scheduling could be complex and inefficient, as it required careful coordination to optimize the use of the computer's limited resources.

Set-Up Time:

- Set-up time referred to the time needed to prepare the computer for running a new program.
- It involved several labor-intensive tasks, such as physically loading the program and data into the computer's memory.
- Additionally, setting up input/output devices and ensuring proper configuration were essential aspects of reducing set-up time.
- The process was often time-consuming, hindering the rapid execution of different computing tasks.

2. Simple Batch Processing:

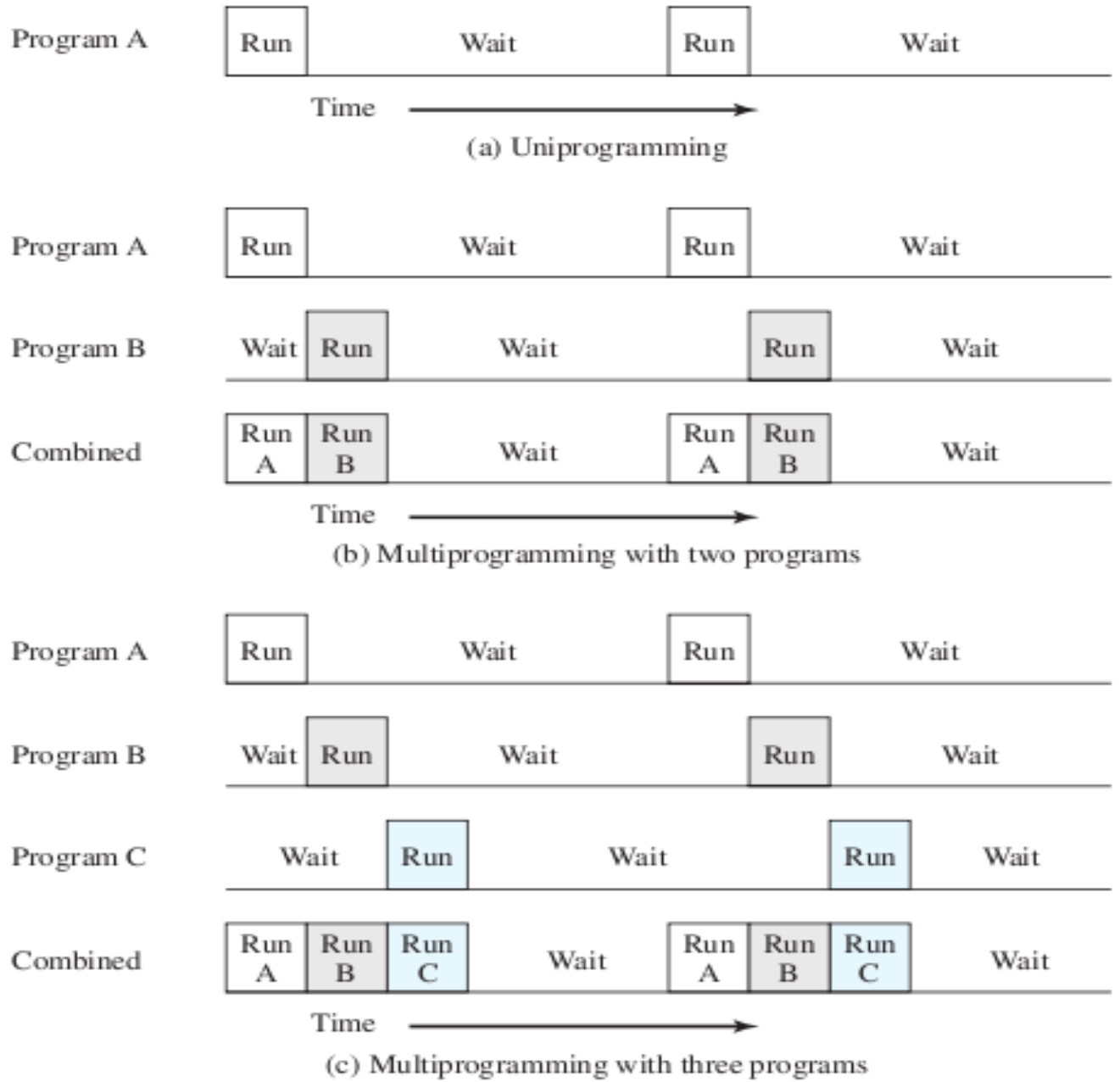
- The wasted time due to scheduling and setup time in Serial Processing was unacceptable.
- Early computers were very expensive, and therefore it was important to maximize processor utilization.
- To improve utilization, the concept of a batch operating system was developed.
- Batch is defined as a group of jobs with similar needs. The operating system allows users to form batches. Computer executes each batch sequentially, processing all jobs of a batch considering them as a single process called batch processing.
- In the simple batch-processing scheme, a piece of software called the monitor plays a central role. Users no longer have direct access to the processor. Instead, they submit their jobs on cards or tapes to a computer operator.
- The computer operator batches similar jobs together sequentially. The entire batch is placed on an input device for use by the monitor.
- Each program within the batch is designed to branch back to the monitor upon completing its processing. The monitor then automatically loads the next program in the batch, ensuring a seamless and efficient workflow.



- With a batch operating system, processor time alternates between execution of user programs and execution of the monitor.
- There have been two sacrifices: Some main memory is now given over to the monitor and some processor time is consumed by the monitor.
- Both of these are forms of overhead.

3. Multiprogramming

- In a computer system where only one program runs at a time, the CPU may not be fully utilized. There can be periods when the CPU remains idle, waiting for input or other operations to complete.
- **Multiprogramming** is a technique that allows multiple programs to be loaded into memory and executed concurrently. This technique aims to **keep the CPU busy at all times** by rapidly switching its attention from one program to another.
- **key advantages** : It enables the overlap of CPU processing and I/O operations. When one program is waiting for an I/O operation to complete (e.g., reading from disk or network), the CPU can be switched to another program that is ready to execute. This overlap reduces idle time and keeps the CPU constantly engaged.
- **Goal:** to ensure that the CPU is never left idle for extended periods. Even if it briefly becomes idle, it quickly switches to another program in the queue, minimizing wasted CPU cycles.
- Multiprogramming improves the overall efficiency of the computer system by making better use of available resources. It allows for the concurrent execution of multiple tasks, leading to faster job completion and improved system responsiveness.


Program A
Run
Wait
Run
Wait
Time
(a) Uniprogramming
Program A
Run
Wait
Run
Wait
Program B
Wait
Run
Wait
Run
Wait
Combined
Run A
Run B
Wait
Run A
Run B
Wait
Time
(b) Multiprogramming with two programs
Program A
Run
Wait
Run
Wait
Program B
Wait
Run
Wait
Run
Wait
Program C
Wait
Run
Wait
Run
Wait
Combined
Run A
Run B
Run C
Wait
Run A
Run B
Run C
Wait
Time
(c) Multiprogramming with three programs

4. Multitasking or Time Sharing System:

- Multiprogramming didn't provide the user interaction with the computer system.
- Time sharing or Multitasking is a logical extension of Multiprogramming that provides user interaction.
- There are more than one user interacting the system at the same time
- The switching of CPU between two users is so fast that it gives the impression to user that he is only working on the system but actually it is shared among different users.
- CPU bound is divided into different time slots depending upon the number of users using the system.
- A multitasking system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time shared computer. Each user has at least one separate program in memory.
- Multitasking are more complex than multiprogramming and must provide a mechanism for jobs synchronization and communication and it may ensure that system does not go in deadlock.

Although batch processing is still in use but most of the system today available uses the concept of multitasking and Multiprogramming.

5. Network Operating System:

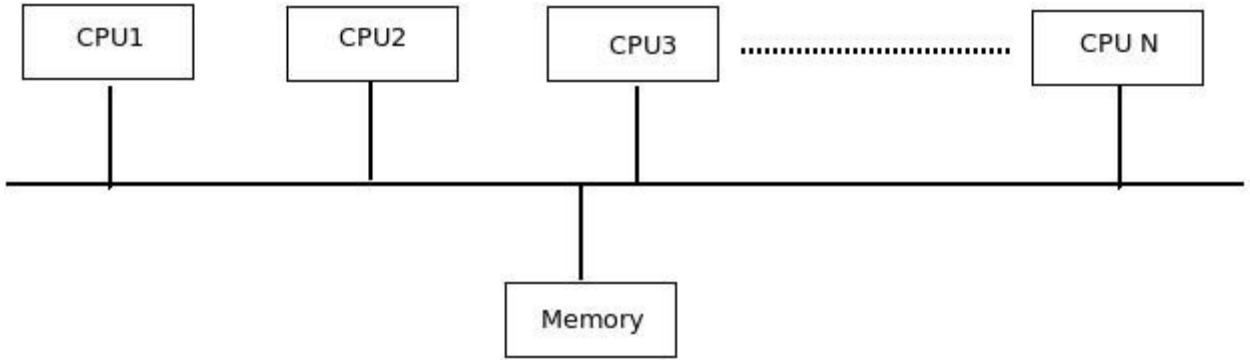
- A specialized type of operating system designed to manage and facilitate the operation of a computer network. Unlike traditional operating systems that primarily focus on managing a single computer, a NOS is tailored to oversee the resources and interactions within a network of computers and devices.
- A NOS plays a central role in coordinating these interactions between different resources in a network and ensuring that the network functions smoothly.
- A NOS manages user access to network resources. It controls who can access what on the network and enforces permissions and security policies.
- The NOS ensures that these resources are allocated efficiently and fairly among network users.
- NOS allows users to share data and files seamlessly across the network. Users can access files stored on other computers or servers, making it easier to collaborate and share information within an organization.
- NOS manages network communication by supporting various network protocols and services. It enables devices to communicate using standardized methods, ensuring compatibility and efficient data exchange.
- Network administrators use the NOS to centrally manage and monitor the network(user account management, software updates, security configurations, and troubleshooting network issues).
- Common examples of Network Operating Systems include Microsoft Windows Server, Linux distributions tailored for server environments, and Novell NetWare
- Network Operating Systems are more complex than standalone operating systems because they need to handle a wide range of network-related tasks, security considerations, and resource-sharing challenges.

6. Distributed Operating System:

- Modern computing emphasizes distributing tasks among multiple computers rather than relying on a single system.
- Distributed Operating Systems: These specialized OSs manage hardware and software resources in distributed environments.
- Distributed System: A network of autonomous computers connected via communication networks, working together.
- Users and apps don't need to know where tasks are executed or resources located.
- Distributed OSs optimize resource usage, like CPU, memory, storage, and networking.
- Nodes communicate through message passing mechanisms.
- Users interact with the system as if it's a single entity, regardless of its distributed nature.
- Benefits: Distributed computing enhances performance, fault tolerance, scalability, and resource utilization in various applications, including cloud computing and distributed databases.

7. Multiprocessor operating system:

- Multiprocessor operating system aims to support high performance through the use of multiple CPUs.
- It consists of a set of processors that share a set of physical memory blocks over an interconnected network.
- Goal: To make the number of CPUs transparent to the application. Achieving such transparency is relatively easy because the communication between different (parts of) application uses the same primitives as those in uni-processor OS.
- The idea is that all communication is done by manipulating data at the shared memory locations and that we only have to protect that data segment against simultaneous access. Protection is done through synchronization primitives like semaphores and monitors.



8. Real Time Operating System:

- **Primary Objective of RTOS**

  - The primary goal of a Real-Time Operating System (RTOS) is to provide quick and deterministic response times.

  - Meeting scheduling deadlines is paramount for RTOS, and it is designed to ensure that tasks are executed within specified time constraints.

- **Applications of Real-Time Systems**

  - RTOS is crucial for applications that involve time-critical operations, where events must be accepted and processed within strict deadlines.

  - Common applications include: Rocket launching and space exploration. , Flight control systems for aircraft and drones, Robotics and automation.

- **Real-Time Operating Systems** are critical in scenarios where time is of the essence and predictable, rapid responses are essential to ensure the safety and effectiveness of the system

- Real time systems are classified into two categories:

  a). Soft Real time System:

  If certain deadlines are missed then system continues its working with no failure but its performance degrade.

  b). Hard Real time System:

  If any deadline is missed then system will fail to work or does not work properly. This system gurantees that critical task is completed on time.

Modern Operating system??

Thank You