BCA 3rd Semester

Operating system

Course Code: BCAC0022

Presented by:
Jayati Krishna Goswami
Assistant Professor, Dept. Of CEA
GLA University, Mathura

BCAC 0022: OPERATING SYSTEM

Objective:

Module No.	Content	Teaching Hours
I	**Introduction:** Operating system and functions, Classification of Operating systems: Batch, Interactive, Time sharing, Real Time System, Multiprocessor Systems, Multiuser Systems, Multithreaded Systems, Operating System Structure, System Components, Operating System Services, Kernels, Monolithic and Microkernel Systems. **Process Management:** Process Concept, Process States, Process Synchronization, Critical Section, Mutual Exclusion, Classical Synchronization Problems, Process Scheduling, Process States, Process Transitions, Scheduling Algorithms Inter-process Communication, Threads and their management, Security Issues. **CPU Scheduling:** Scheduling Concepts, Techniques of Scheduling, Preemptive and Non-Preemptive Scheduling: First-Come-First-Serve, Shortest Request Next, Highest Response Ration Next, Round Robin, Least Complete Next, Shortest Time to Go, Long, Medium, Short Scheduling, Priority Scheduling. Deadlock: System model, Deadlock characterization, Prevention, Avoidance and detection, Recovery from deadlock. **Memory Management:** Memory allocation, Relocation, Protection, Sharing, Paging, Segmentation, Virtual Memory, Demand Paging, Page Replacement Algorithms, Thrashing.	30
II	**I/O Management and Disk Scheduling:** I/O devices, and I/O subsystems, I/O buffering, Disk storage and disk scheduling, RAID. **File System:** File concept, File organization and access mechanism, File directories, and File sharing, File system implementation issues, File system protection and security. **Shell introduction and Shell Scripting:** What is shell and various type of shell, Various editors present in linux, Different modes of operation in vi editor, What is shell script, Writing and executing the shell script, Shell variable (user defined and system variables) System calls, Using system calls, Pipes and Filters, Decision making in Shell Scripts (If else, switch), Loops in shell, Functions, Utility programs (cut, paste, join, tr , uniq utilities), Pattern matching utility (grep).	30

Text Book:

- Abraham Silberschatz, Greg Gagne, and Peter B. Galvin, "Operating System Concepts," Tenth Edition, Wiley, 2018.

Reference Books:

- Andrew S. Tanenbaum and Herbert Bos,"Modern Operating Systems," Fourth Edition, Pearson, 2014.
- William Stallings, "Operating Systems: Internals and Design Principles," Seventh Edition, Prentice Hall, 2011.
- Dhanjay Dhamdhere, "Operating Systems," First Edition, McGraw-Hill, 2008.
- Milan Milankovic "Operating systems, Concepts and Design" McGraw Hill.

Outcome: *A student who successfully completes the course will have the ability to:*

- CO1: Understand role, responsibilities, features, and design of operating system.
- CO2: Analyze memory management schemes and process scheduling algorithms.
- CO3: Apply process synchronization techniques to formulate solution for critical section problems.
- CO2: Illustrate concept of disk scheduling.
- CO5: Evaluate process deadlock handling techniques.

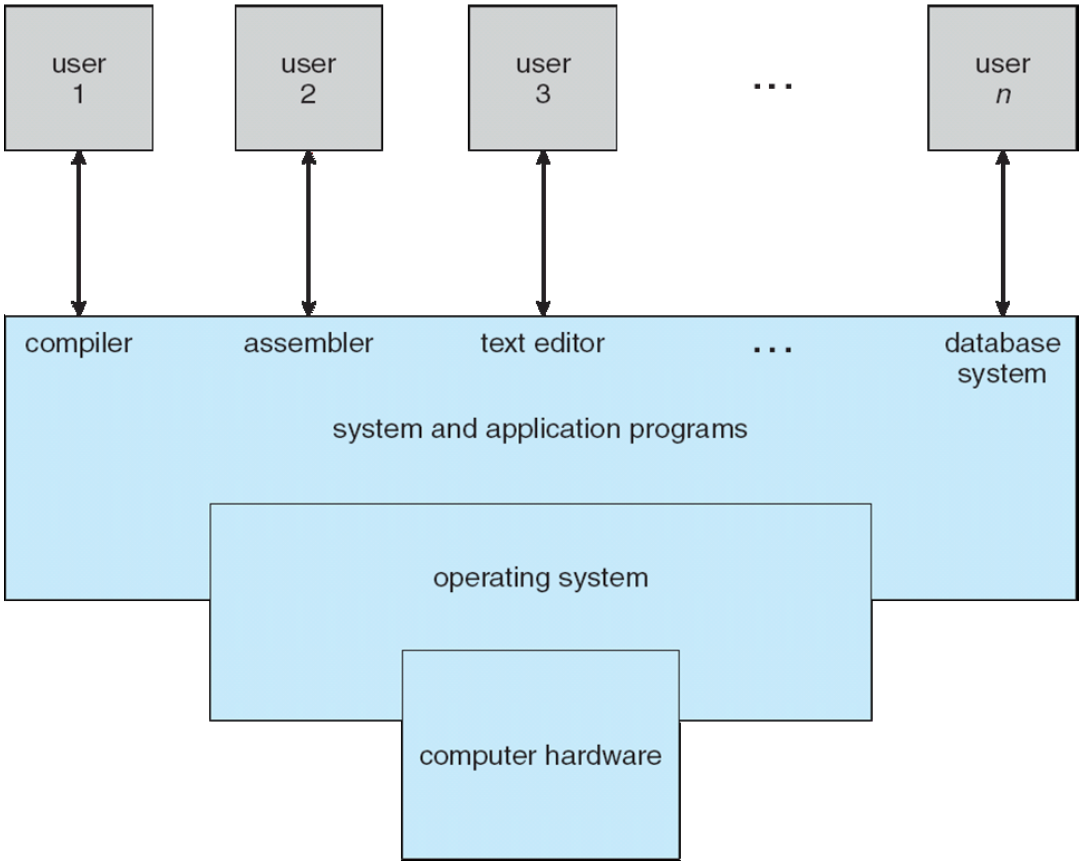
An Operating System

- An **Operating System** (**OS**) is a collection of program which provides an interface between a computer user and computer hardware.
- An **operating system** is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.
- A Collection program that acts as an intermediary between a user of a computer and the computer hardware.
- It is responsible for the management and coordination of activities and the sharing of the resources of the computer.

Computer System Components

- **Hardware** Provides basic computing resources (CPU, memory, I/O devices).
- **Operating System** Controls and coordinates the use of hardware among application programs.
- **Application Programs** Solve computing problems of users (compilers, database systems, video games, business programs such as banking software).
- **Users** People, machines, other computers

Four Components of a Computer System



Operating System Views

- Resource allocator :to allocate resources (software and hardware) of the computer system and manage them efficiently.
- Control program : Controls execution of user programs and operation of I/O devices.
- Kernel :The program that executes forever (everything else is an application with respect to the kernel).

Goals of an Operating System

- Simplify the execution of user programs and make solving user problems easier.
- Use computer hardware efficiently.
- Allow sharing of hardware and software resources.
- Make application software portable and versatile.
- Provide isolation, security and protection among user programs.
- Improve overall system reliability

Functions/Components of Operating System:

Process management

- A program in its execution state is known as **process.**
- A process needs certain resources including CPU time, memory, files and I/O devices to accomplish its task.
- These resources are either given to the process when it is created or allocated to it while it is running.
- A program is a passive *entity* such as contents of a file stored on the disk whereas a process is an active *entity*

The operating system is responsible for the following activities in process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling.

2. Memory management

Main memory is a collection of quickly accessible data shared by the CPU and I/O devices.

The central processor reads instructions from main memory (during instruction-fetch cycle) and both reads and writes data from main memory (during data-fetch cycle).

The operating system is responsible for the following activities in memory management:

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

3. File-System Management

The operating system is responsible for the following activities with file management:

- Creating and deleting files
- Creating and deleting directories to organize files
- Supporting primitives for manipulating files and directories
- Backing up files on stable (nonvolatile) storage media.

4. Secondary storage Management The operating system is responsible for the following activities with disk management:

- Free-space management
- Storage allocation
- Disk scheduling

5. I/O system Management

The operating system is responsible for the following activities with I/O subsystem:

- A memory-management component that includes buffering, caching, and spooling
- A general device-driver interface
- Drivers for specific hardware devices

6. Protection and Security

- A mechanisms ensure that files, memory segments, CPU, and other resources can be operated on by only those processes that have been allowed proper authorization from the operating system.
- For example, memory-addressing hardware ensures that a process can execute only within its own address space.
- Protection is a mechanism for controlling the access of processes or users to the resources defined by a computer system.

7. Networking

- A distributed system is a collection of physically separated computer systems that are networked to provide the users with access to the various resources that the system maintains.
- Access to a shared resource increases computation speed, functionality, data availability, and reliability.
- Network Operating System (NOS) provides remote access to the users.
- It also provides the sharing of h/w and s/w resources from remote machine to own systems.

8.Command Interpreter

- To interface with the operating System we use command-line interface or **command interpreter** that allows users to directly enter commands that are to be performed by the operating system.
- The main function of the command interpreter is to get and execute the user-specified commands. Many of the commands given at this level manipulate files: create, delete, list, print, copy, execute, and so on. Eg: MS-DOS and UNIX shells.

Types of Operating System

- **Single user Operating system** This OS provides the environment for single user i.e. only one user can interact with the system at a time. Eg: MS-DOS, MS WINDOWS-XP, ME, 2000 etc.

- **Multi user Operating System** This OS provides the environment for multiple users i.e. many user can interact with the system at a time. These users are remotely connected to a system taking benefits of shared resources of master system through networking. Eg: UNIX, LINUX.

Serial Processing Operating System

- 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

- In a typical sequence first the editor is been called to create a source code of user program then translator is been called to convert source code into its object code, finally the loader is been called to load its executable program into main memory for execution.

- If syntax errors are detected than the whole program must be restarted from the beginning.

Batch processing Operating System

- Early computers were not interactive device, there user use to prepare a job which consist three parts
    1. Program
    2. Control information
    3. Input data
- Only one job is given input at a time as there was no memory, computer will take the input then process it and then generate output.
- Common input/output device were punch card or tape drives. So these devices were very slow, and processor remain ideal most of the time.

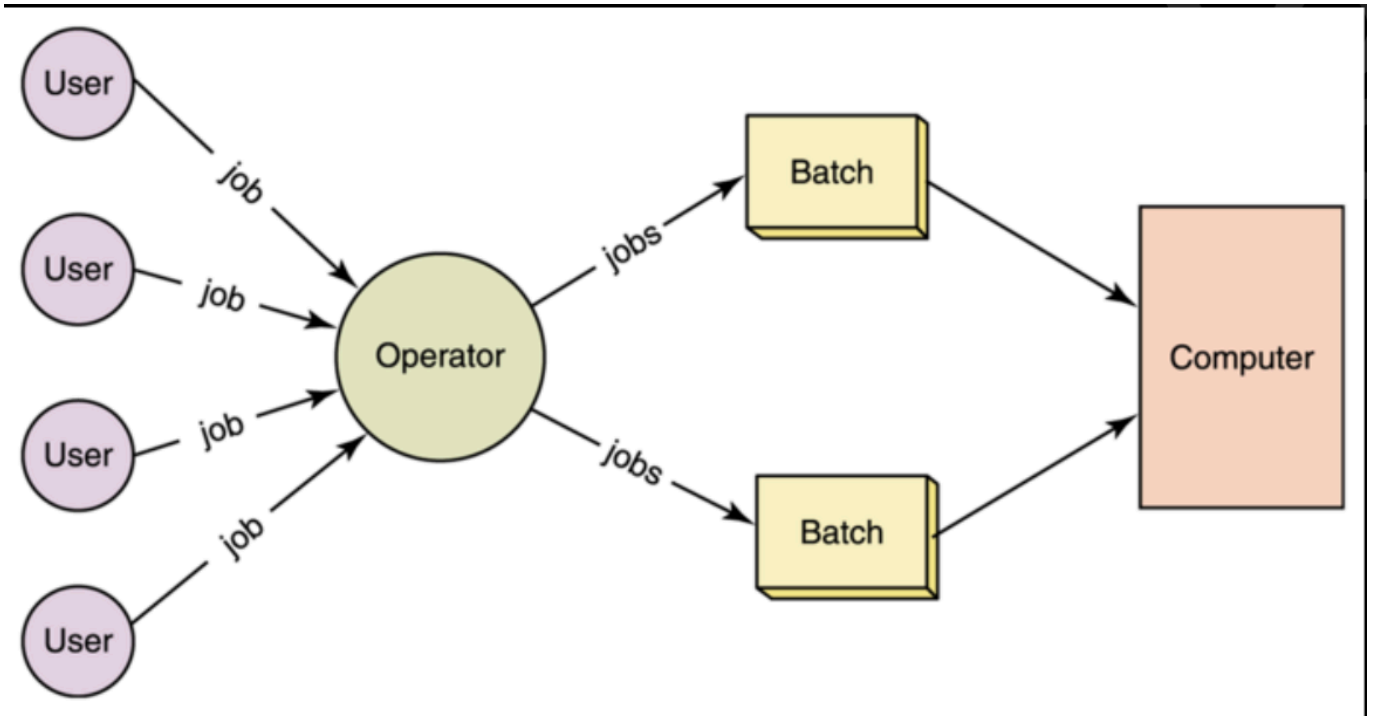
Batch processing Operating System

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Batch processing Operating System

- To speed up the processing job with similar types (for e.g. FORTRAN jobs, COBOL jobs etc.) were batched together and were run through the processor as a group (batch).
- In some system grouping is done by the operator while in some systems it is performed by the 'Batch Monitor' resided in the low end of main memory)
- Then jobs (as a deck of punched cards) are bundled into batches with similar requirement.

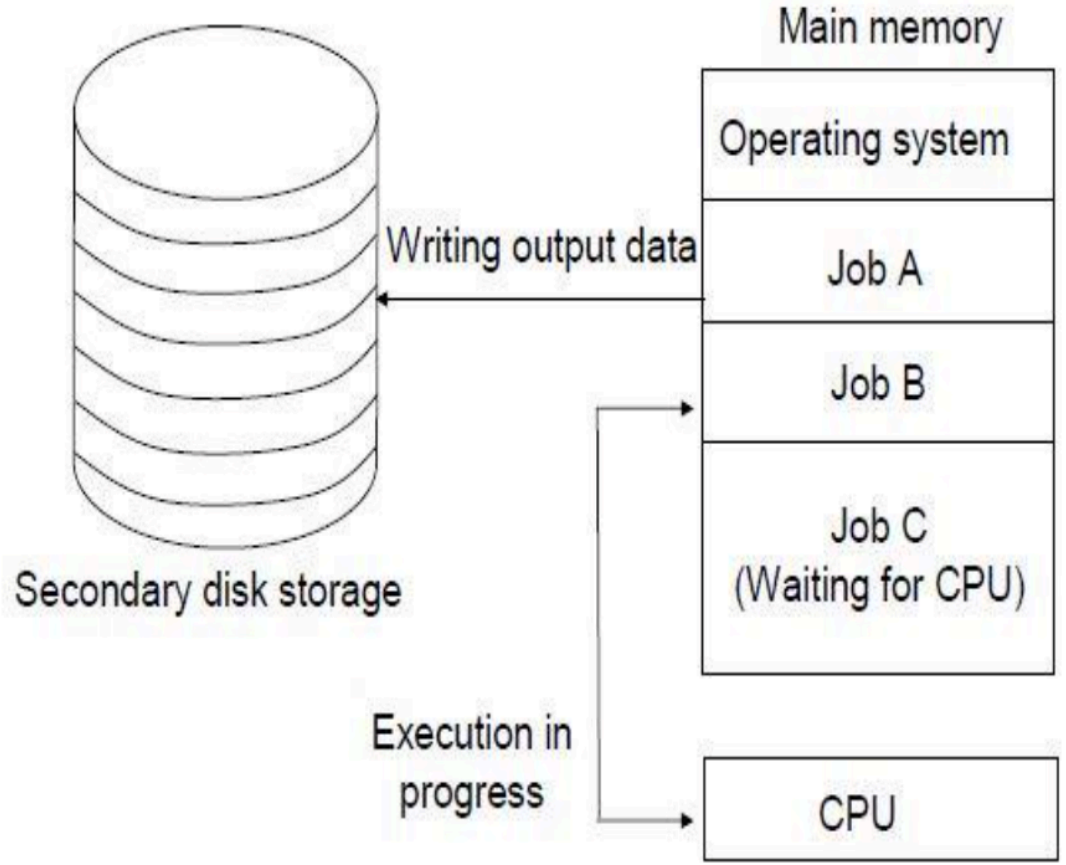
Batch processing Operating System



Multiprogramming

- Multiprogramming is a technique to execute number of programs simultaneously by a single processor.
- In Multiprogramming, number of processes reside in main memory at a time.
- The OS picks and begins to executes one of the jobs in the main memory.
- If any I/O wait happened in a process, then CPU switches from that job to another job.
- Hence CPU in not idle at any time.

Multiprogramming

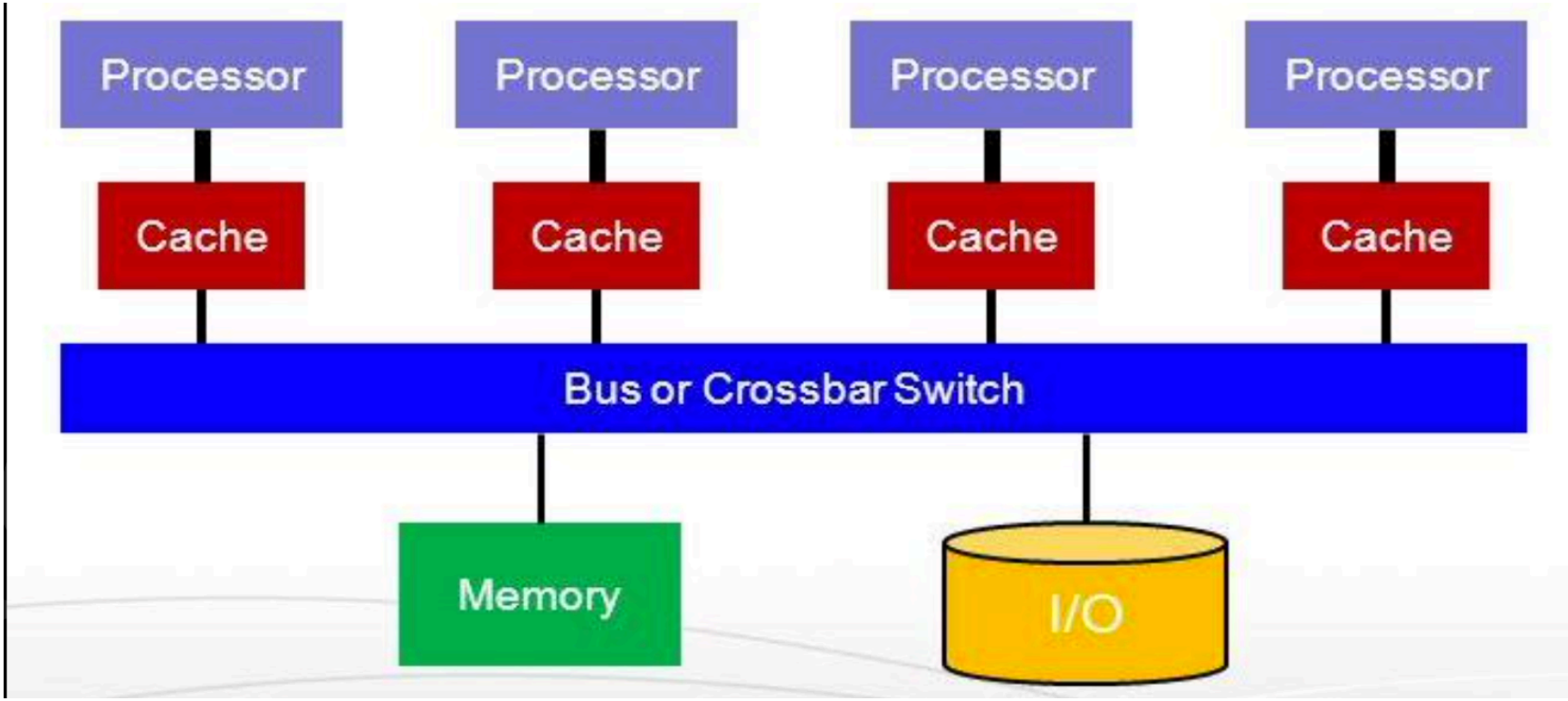


- Figure dipicts the layout of multiprogramming system.
- The main memory consists of 3jobs at a time, the CPU executes one by one.
- **Advantages & Disadvantages:**
- Efficient memory utilization
- Throughput increases
- CPU is never idle, so performance increases.
- Complex Scheduling: Difficult to program.
- Complex Memory Management: Intricate handling of memory is required.

Multiprocessor system :-This system has more than one processor which share common bus, clock, peripheral devices and sometimes memory.

These systems have following advantages:

- **Increased throughput** By increasing the number of processor, we get more work done in less time.
- **Economy of scale** Multiprocessor system are cost effective as several processors share same resources of the system(memory, peripherals etc.)
- **Increased reliability** each processor is been fairly allotted with different job, failure of one processor will not halt the system, only it will slower down the performance. for example if we have ten processor and one fails, then each of the remaining nine processor share the work of failed processor. Thus the system will be 10% slower rather than failing altogether



This system can be categorized into:

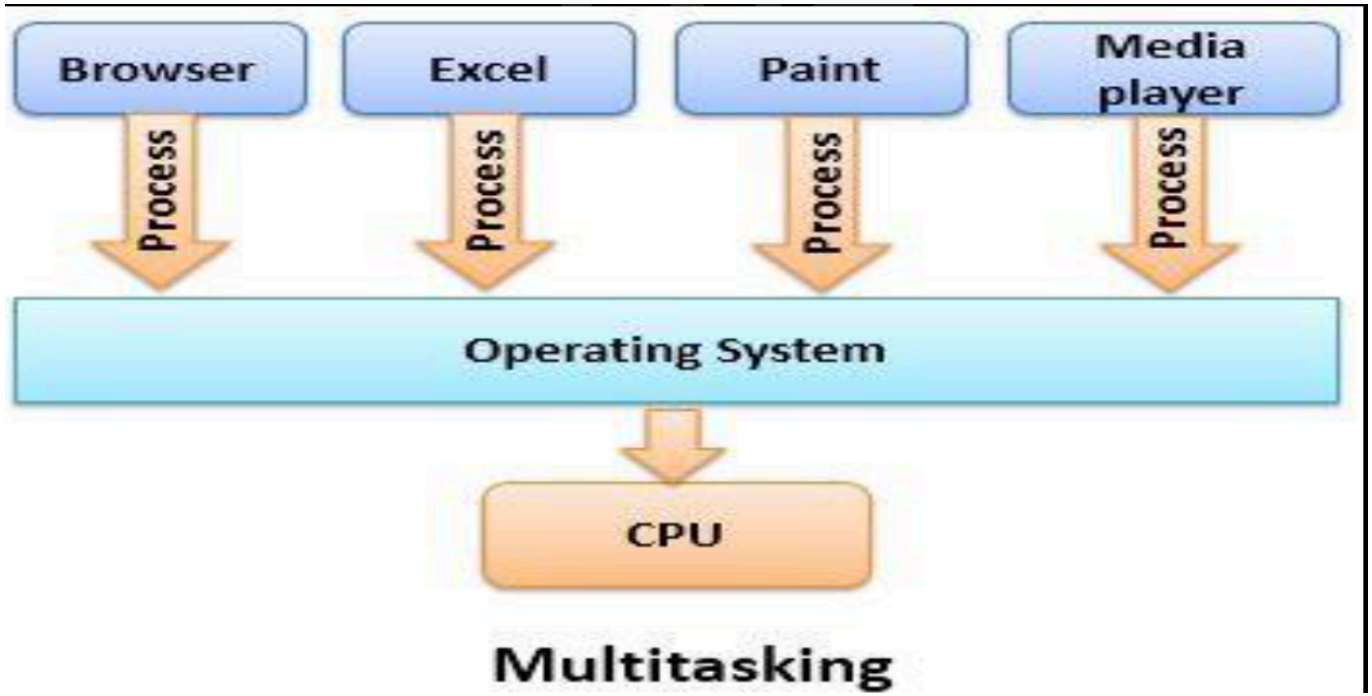
- **i) SMP (Symmetric multiprocessing)**In this system each processor runs an identical copy of the task and however these processor communicate with each other whenever needed. Eg Windows NT, Solaris, Unix/Linux.

- **ii) ASMP(Asymmetric multiprocessing)** In asymmetric multiprocessing each processor runs a specific task. As there is a master processor which controls the whole system,and other processor looks to the master for instructions. Eg Sun OS ver. 4

Time sharing System(Multitasking)

- Time sharing, or multitasking, is a logical extension of multiprogramming.
- Multiple jobs are executed by switching the CPU between them.
- In this, the CPU time is shared by different processes, so it is called as "Time sharing Systems".
- A time-shared operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer.
- Time slice is defined by the OS, for sharing CPU time between processes.
- Examples: Multics, Unix, etc.,

Time sharing System(Multitasking)

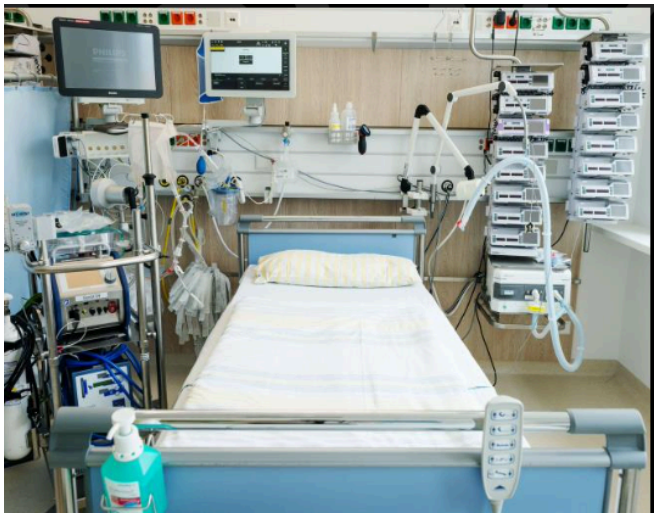


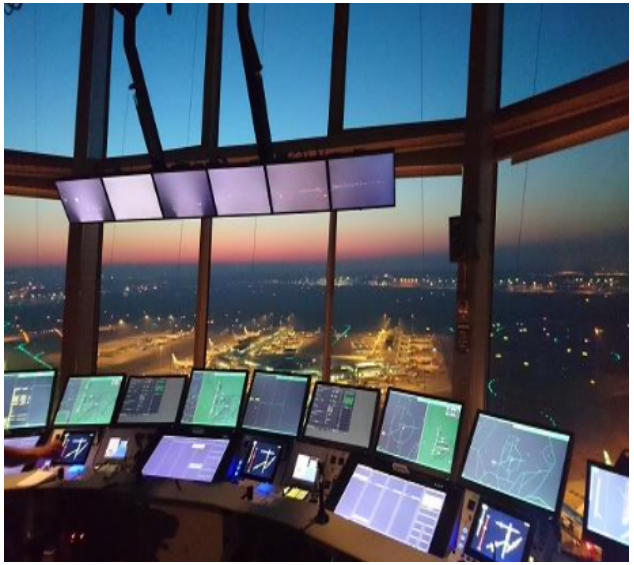
Time sharing System(Multitasking)

1. It allows many users to share the computer simultaneously. the CPU executes multiple jobs (May belong to different user) by switching among them, but the switches occur so frequently that, each user is given the impression that the entire computer system is dedicated to his/her use, even though it is being shared among many users.
2. In the modern operating systems, we are able to play MP3 music, edit documents in Microsoft Word, surf the Google Chrome all running at the same time. (by context switching, the illusion of parallelism is achieved)
3. For multitasking to take place, firstly there should be multiprogramming i.e. presence of multiple programs ready for execution. And secondly the concept of time sharing.

Real time Operating System

- A real time system has well defined fixed time constraints, processing must be done within defined constraints or system will get failed.
- System that controls scientific system, experimenting medical system, industrial control system and certain display systems are real time system.
- They are also applicable to automobile engine fuel system, home appliance controller and weapon systems.
- There are two types of real system:





- **i) <u>Hard real time system</u>** This system guarantees that critical tasks be completed on time. For this all the delays in the system should be bounded, from the retrieval of stored data to the time it takes operating system to finish any request made to it.

- **ii<u>) Soft real time system</u>** This is less restrictive type of system defined as **not hard real-time**, simply providing that a critical real-time task will receive priority over other tasks and that it will retain the priority until it completes.

Network Operating System :-An Os that includes special functions for connecting computers and devices into a LAN. Some OS such as UNIX and the Mac OS, having networking functions built in.

Some popular NOS's for DOS and Windows systems include Novell Netware, Microsoft LAN Manager and Windows NT.

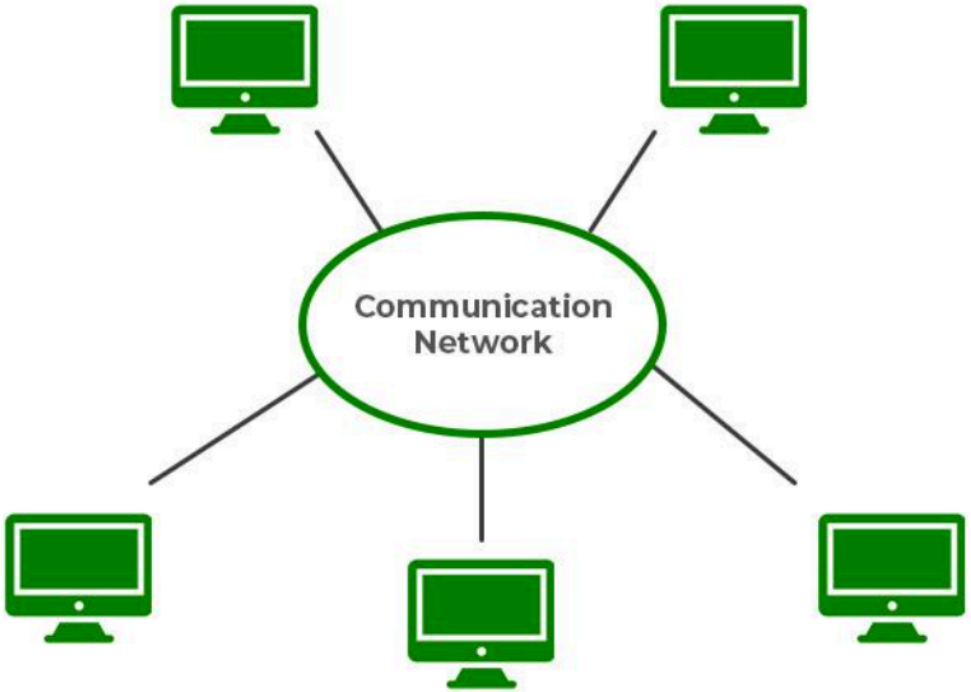
- **Some characteristics**
- Each computer has its own private OS, instead of running part of a global system wide operating system.
- Each user normally works on his/her own system.

Distributed Operating System :-It hides the existence of multiple computers from the user i.e. the user doesn't know that many computers are being used to process the data.

These computers may be located at many places around the globe. This OS provides provide single-system image to its users. All these computers work in close coordination with each other.

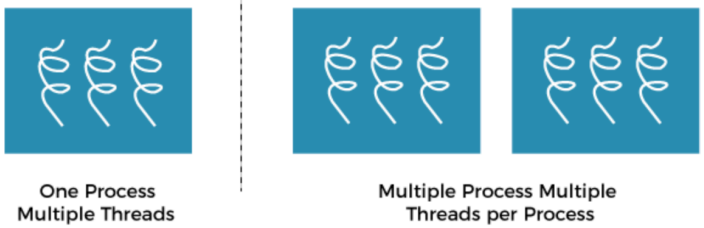
- In this OS, each processor has its own memory and clock. The processor communicates with each other through various communication lines such as high speed buses and telephone lines.
- There are four major reasons for building distributed systems: resource sharing, computation speedup, reliability, and communication.

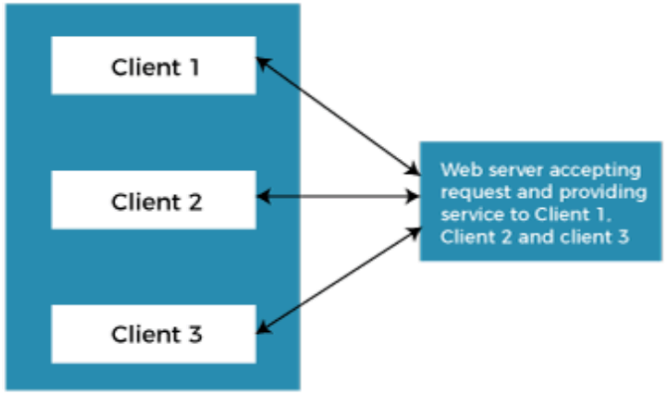
Distributed Operating System

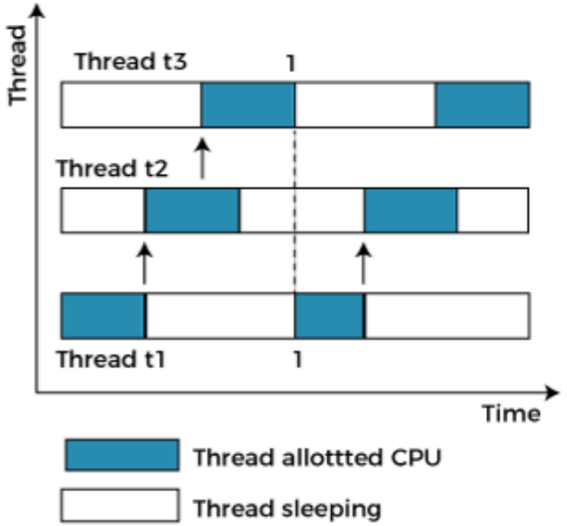


Multithreading Operating System

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC







Multithreading Operating System

- Multithreading allows the application to divide its task into individual threads.
- In multi-threads, the same process or task can be done by the number of threads, or we can say that there is more than one thread to perform the task in multithreading.
- With the use of multithreading, multitasking can be achieved.
- The main drawback of single threading systems is that only one task can be performed at a time, so to overcome the drawback of this single threading, there is multithreading that allows multiple tasks to be performed.

Operating-System Services

1) User interface Almost all operating systems have a **user interface** (UI).This interface can take several forms.

- **command-line interface (CLI)** which uses text commands.
- **graphical user interface (GUI)** the interface is a window system with a pointing device to direct I/O, choose from menus, and make selections and keyboard to enter text.

2) Program execution:- The system should to load a program into memory and to run that program. The program must be able to end its execution.

3) I/O operations:- A running program may require I/O operation(such as recording to a CD or DVD drive or blanking a CRT screen). For efficiency and protection, users usually cannot control I/O devices directly. Therefore the operating system must provide a means to do I/O.

4)File-system manipulation:- Programs need to read and write files. They also need to create and delete. Finally, some programs include permissions management to allow or deny access to files or directories based on file ownership.

5**) 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

6) **Communications.** There are many circumstances in which one process needs to exchange information with another process.

Such communication may occur between processes that are executing on the same computer or between processes that are executing on different computer systems tied together by a computer network.

7) **Error detection.** Errors may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/O devices (a network failure, or lack of paper in the printer), and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location, or too-great use of CPU time).

For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing.

8. Resource allocation. When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them. Many different types of resources are managed by the operating system trough various methods such as CPU-scheduling.

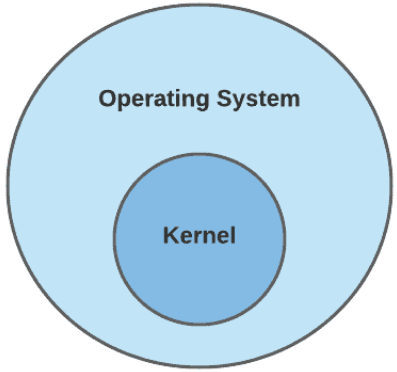
9. Accounting. OS keeps track of which users use how much and what kind of computer resources. This record keeping may be used for accounting

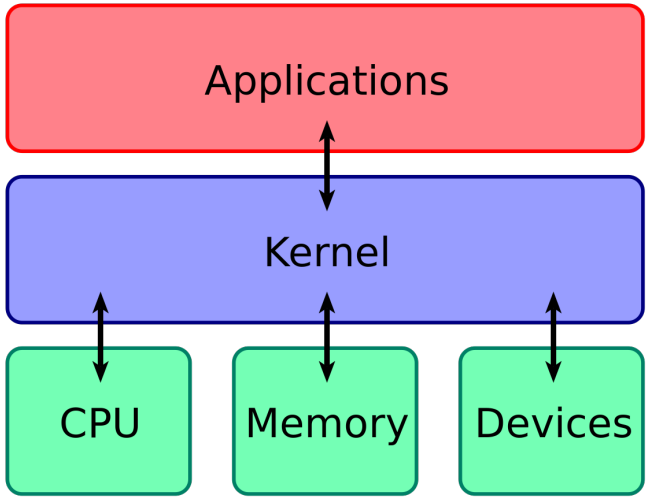
10. Protection and security. When several separate processes execute concurrently, it should not be possible for one process to interfere with the others or with the operating system itself.

Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important by means of a password, to gain access to system resources.

Kernel

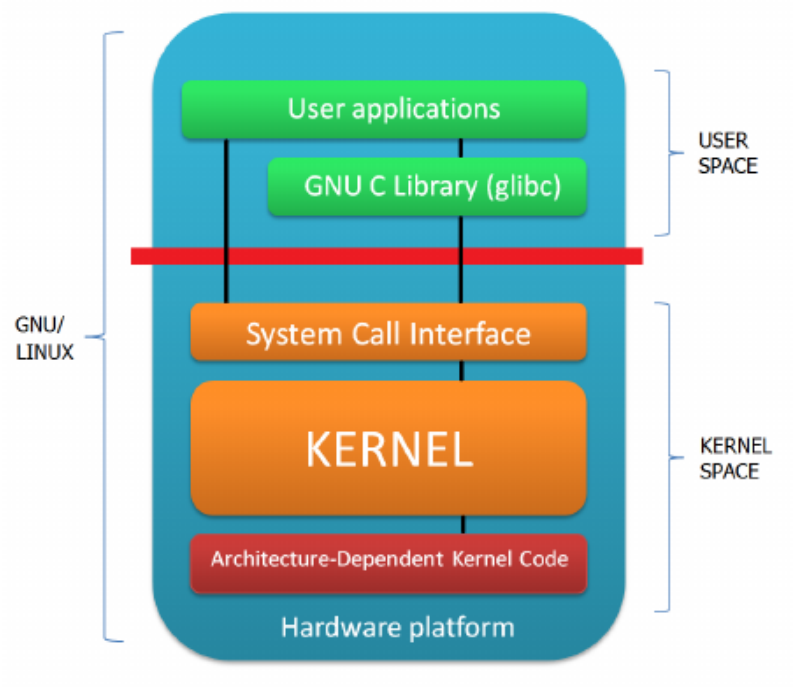
- Kernel is a part & lowest layer of a operating system.
- It provides service to all other parts of OS.
- It acts as interface between hardware and processes of a computer.
- It loads first in memory after OS has been loaded and remains in memory until shutdown.

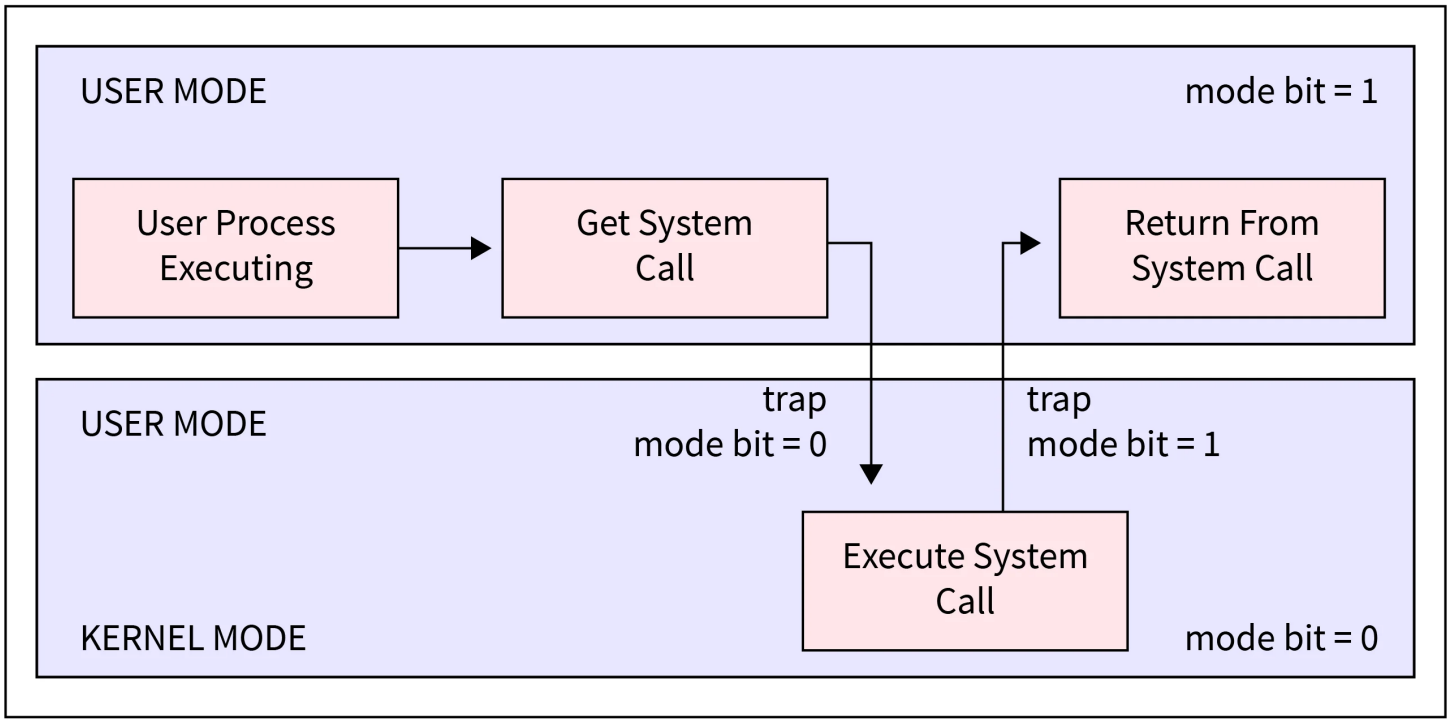




Kernel

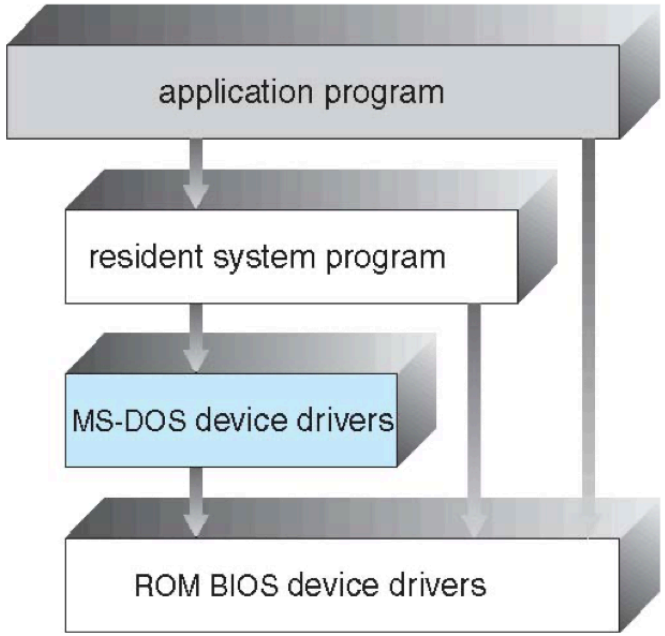
- Functions are device, task and memory management.
- Memory has user space and protected kernel space.

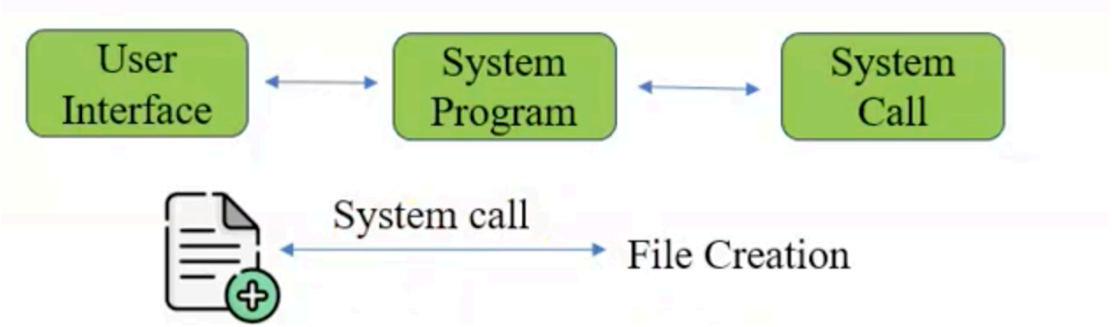




OS Structures

- **Simple Structure- MS-DOS**

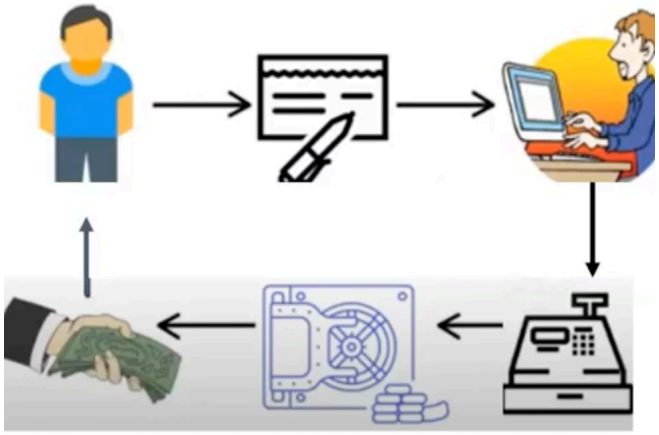


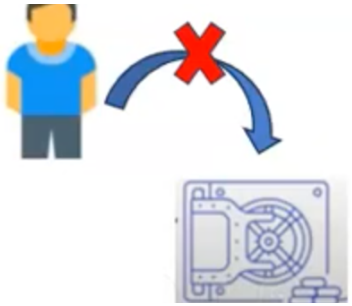


System becomes **vulnerable** and malicious programs can access base H/W

- Basic Input Output System stored in ROM
- Drivers to start the system

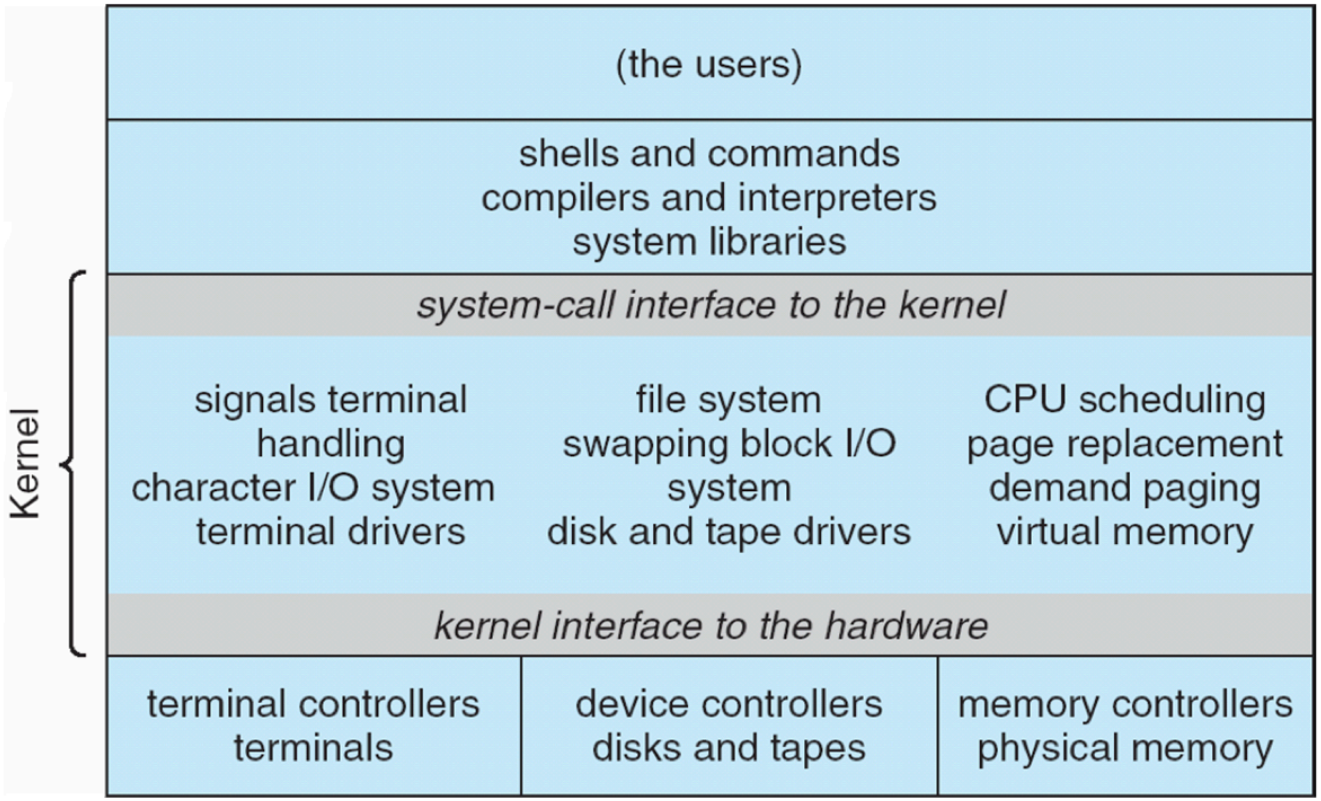
- Easy to Develop, maintain and provides good performance as layers are limited.
- Vulnerable to to unauthorised access and BIOS and Device drivers is accessible to all layers.





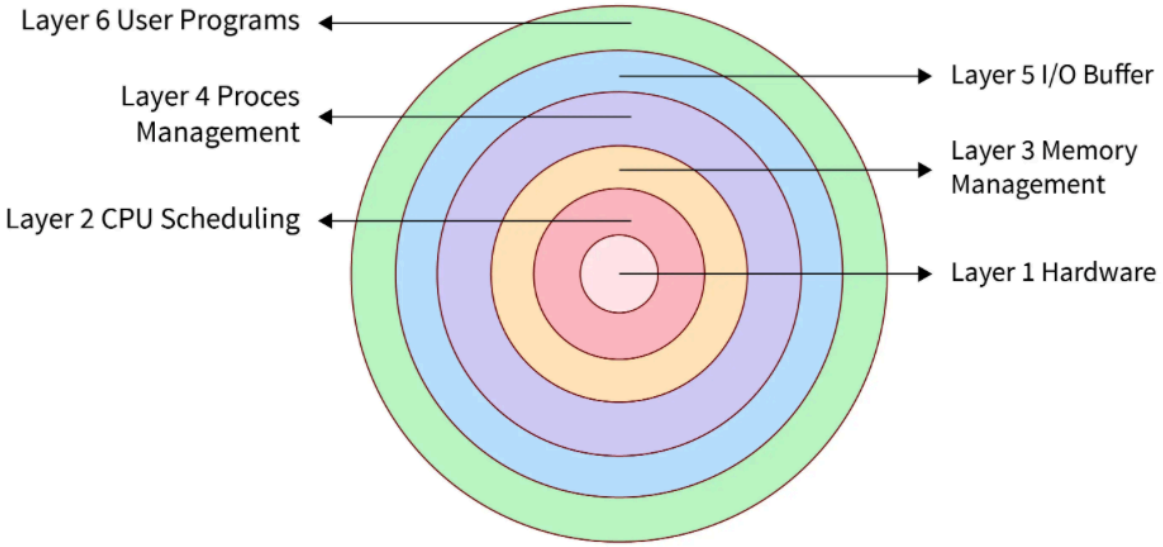
- **Monolithic Structure - UNIX**

Too Many functions packed into one level. maintenance and implementation difficult.



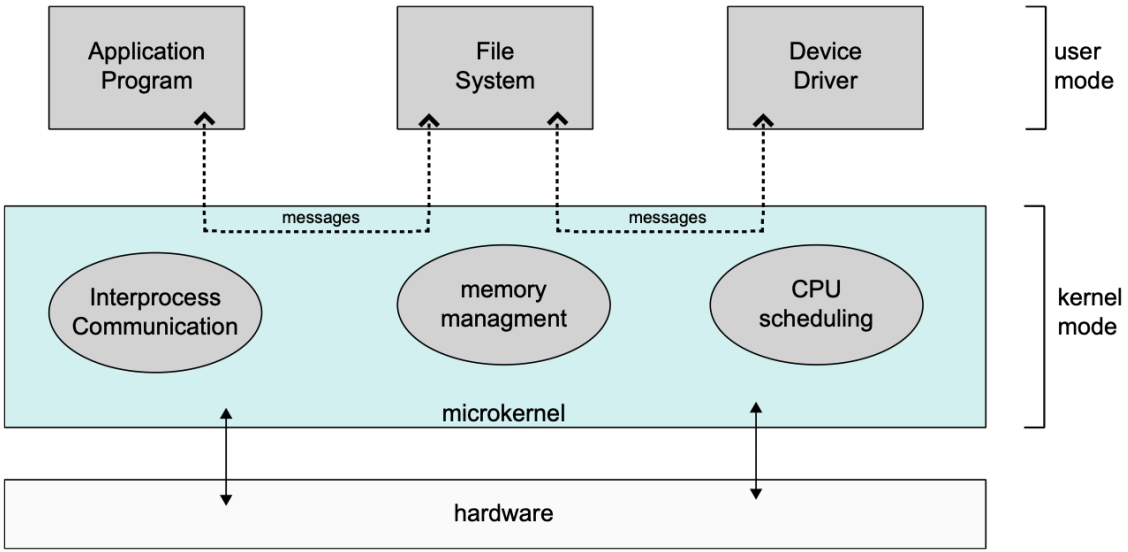
- **Layered structure**

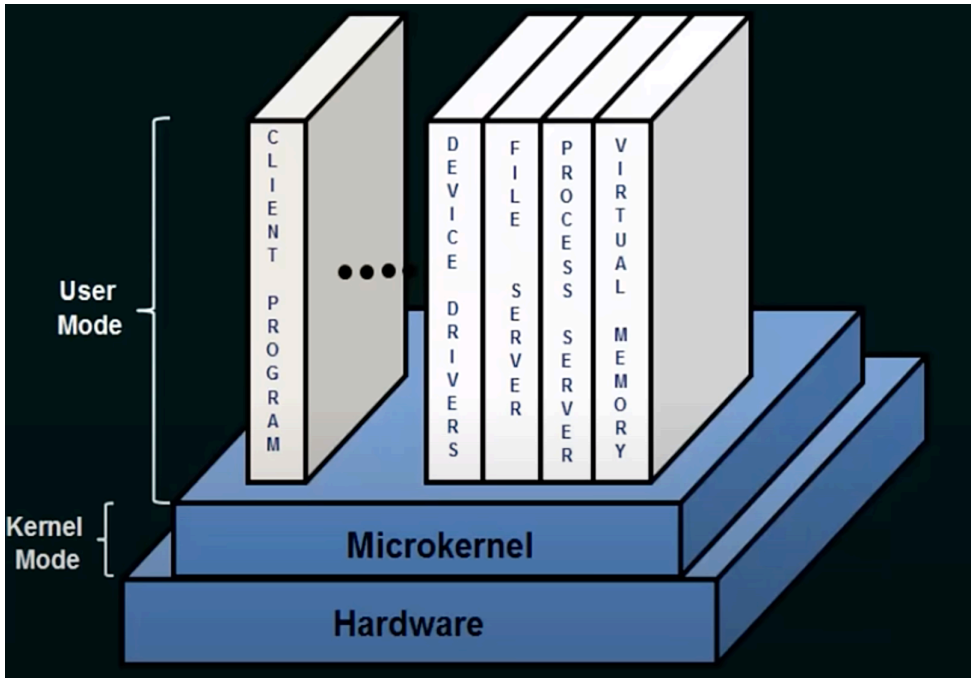
Easy to implement and debug.
But designing & deciding of layers are difficult.
also response to top layer from bottom layer may be slow.



- Layered Approach - With proper hardware support, operating systems can be broken into pieces. The operating system can then retain much greater control over the computer and over the applications that make use of that computer.
- Implementers have more freedom in changing the inner workings of the system and in creating modular operating systems.
- Under a top-down approach, the overall functionality and features are determined and are separated into components.
- A system can be made modular in many ways. One method is the layered approach, in which the operating system is broken into a number of layers (levels). The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface.

- **Microkernel structure- Mach**





- This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel.
- One benefit of the microkernel approach is that it makes extending the operating system easier. All new services are added to user space and consequently do not require modification of the kernel.
- When the kernel does have to be modified, the changes tend to be fewer, because the microkernel is a smaller kernel.
- The MINIX 3 microkernel, for example, has only approximately 12,000 lines of code. Developer Andrew S. Tanenbaum.

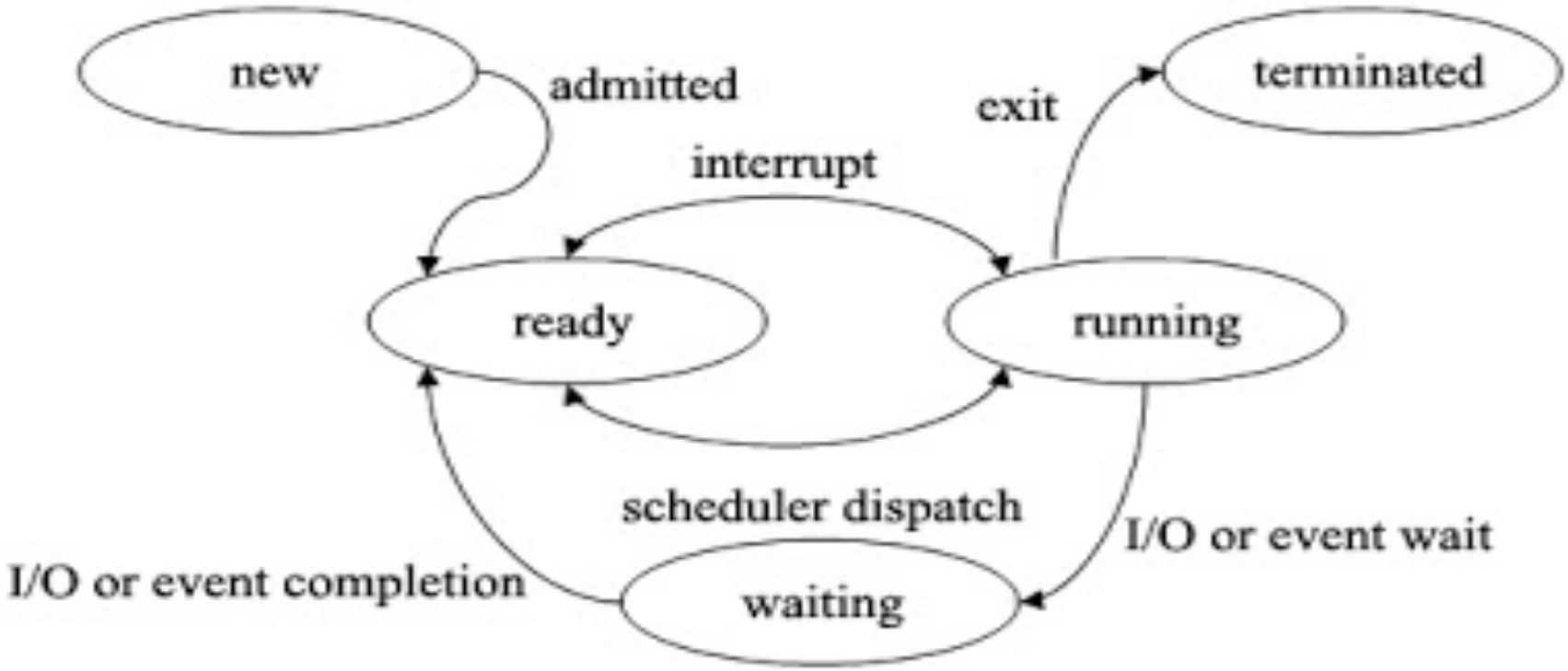
Definition of Process

- A program in execution
  - A process has its own address space consisting of:
    - Text region
      - Stores the code that the processor executes
    - Data region
      - Stores variables and dynamically allocated memory
    - Stack region
      - Stores instructions and local variables for active procedure calls

Process States: Life Cycle of a Process

- **Process State**
- As a process executes, it changes **state.** The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states:
    - **New.** The process is being created.
    - **Running.** Instructions are being executed.
    - **Waiting.** The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
    - **Ready.** The process is waiting to be assigned to a processor
    - **Terminated.** The process has finished execution.

Process state transition diagram



Process Management

- Operating systems provide fundamental services to processes including:
  - Creating processes
  - Destroying processes
  - Suspending processes
  - Resuming processes
  - Changing a process's priority
  - Blocking processes
  - Waking up processes
  - Dispatching processes
  - Interprocess communication (IPC)

Process States and State Transitions

- Process states
  - The act of assigning a processor to the first process on the ready list is called dispatching
  - The OS may use an interval timer to allow a process to run for a specific time interval or quantum
  - Cooperative multitasking lets each process run to completion
- State Transitions
  - At this point, there are four possible state transitions
    - When a process is dispatched, it transitions from *ready* to *running*
    - When the quantum expires, it transitions from *running* to *ready*
    - When a process blocks, it transitions from *running* to *blocked*
    - When the event occurs, it transitions from *blocked* to *ready*

Process Control Blocks (PCBs)

- Each process is represented in the operating system by a **process control block (PCB)** also called a *task control block.* A PCB contains many pieces of information associated with a specific process, including these:
  - **Process state.** The state may be new, ready, running, waiting, halted, and so on.
  - **Program counter.** The counter indicates the address of the next instruction to be executed for this process.
  - **CPU registers.** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.

Block diagram of PCB

- • **CPU-scheduling information.** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- • **Memory-management information.** This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
- • **Accounting information.** This information includes the amount of CPU and real time used, time limits, account members, job or process numbers, and so on.
- • **I/O status information.** This information includes the list of I/O devices allocated to the process, a list of open files, and so on.
- In brief, the PCB simply serves as the repository for any information that may vary from process to process.
-

Process Scheduling

- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
- The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running.
- To meet these objectives, the **process scheduler** selects an available process (possibly from a set of several available processes) for program execution on the CPU.

- As processes enter the system, they are put into a **job queue,** which consists of all processes in the system. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the **ready queue.** A ready-queue header contains pointers to the first and final PCBs in the list.
- The list of processes waiting for a particular I/O device is called a device **queue.**
- A new process is initially put in the ready queue. It waits there until it is selected for execution, or is **dispatched.** Once the process is allocated the CPU and is executing, one of several events could occur:

- The process could issue an I/O request and then be placed in an I/O queue.
- The process could create a new subprocess and wait for the subprocess's termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.
- A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.

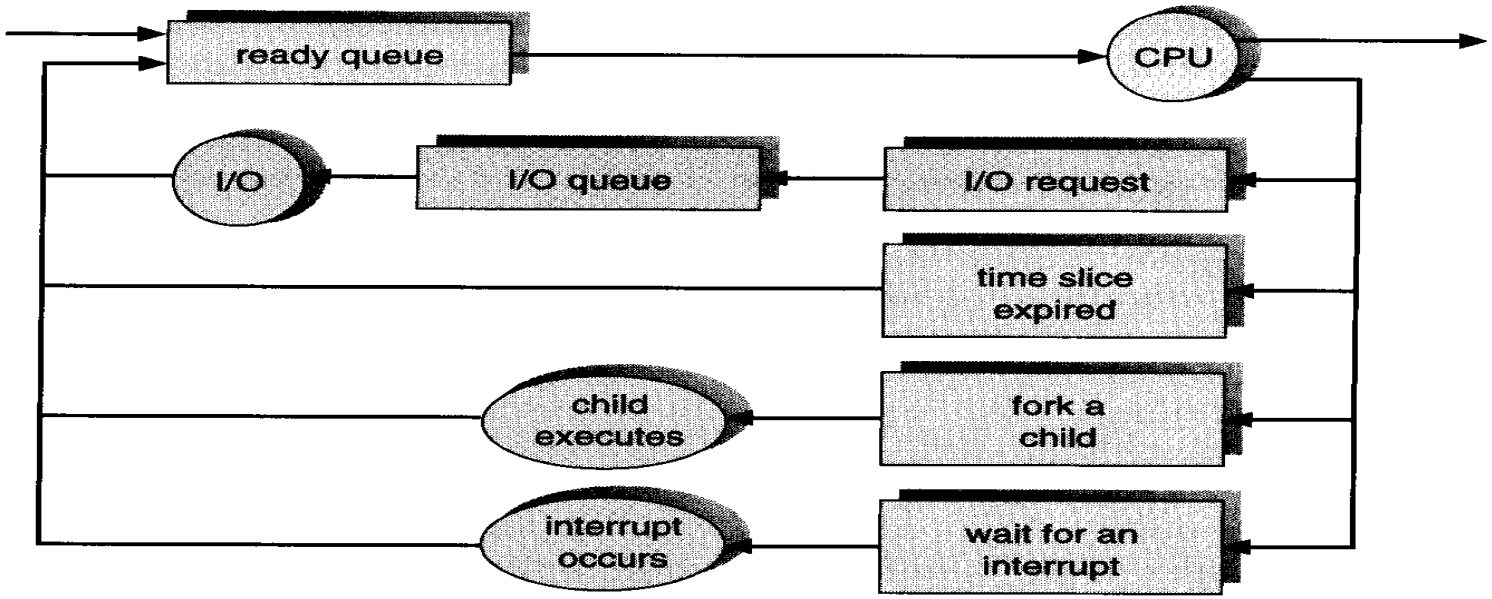


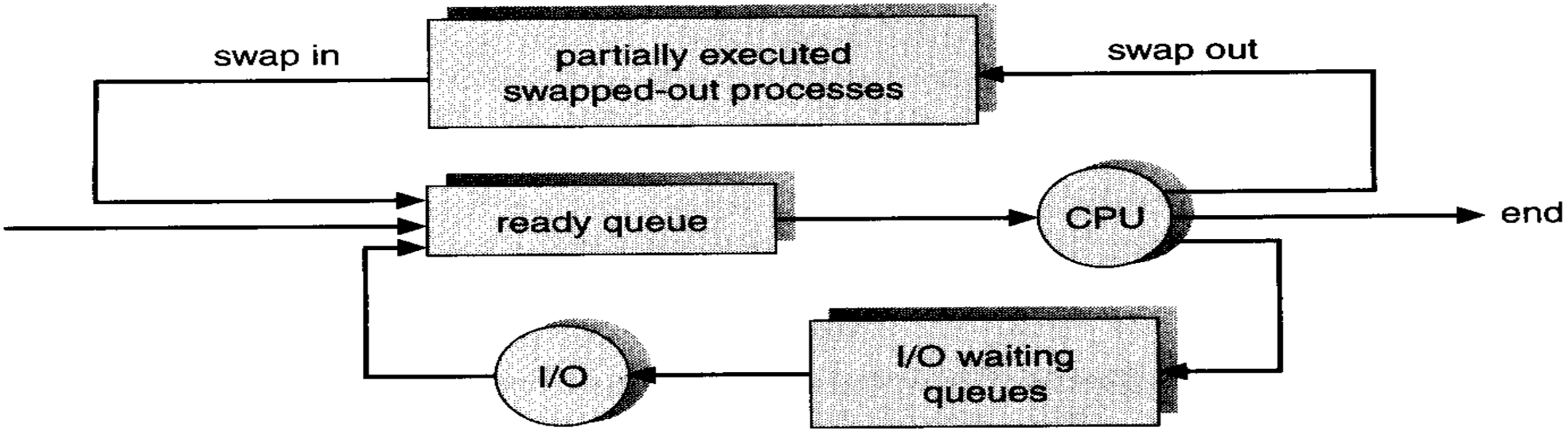
Fig: **Queuing diagram representation of process scheduling**

Schedulers

- A process migrates among the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate **scheduler.**
- Types of schedulers:

- **i) Long term Scheduler** In batch system, more processes are submitted than can be executed immediately. These processes are spooled to a mass-storage device (typically a disk), where they are kept for later execution. The **long-term scheduler,** or **job scheduler,** selects processes from this pool and loads them into memory for execution. The long-term scheduler executes much less frequently; minutes may separate the creation of one new process and the next. The long-term scheduler controls the **degree of multiprogramming(**the number of processes in memory**).** It is important that the long-term scheduler make a careful selection.
- **ii) Short term Scheduler** The **short-term scheduler, or** CPU **scheduler,** selects from among the processes that are ready to execute and allocates the CPU to one of them. Because of the short time between executions, the short-term scheduler must be fast.

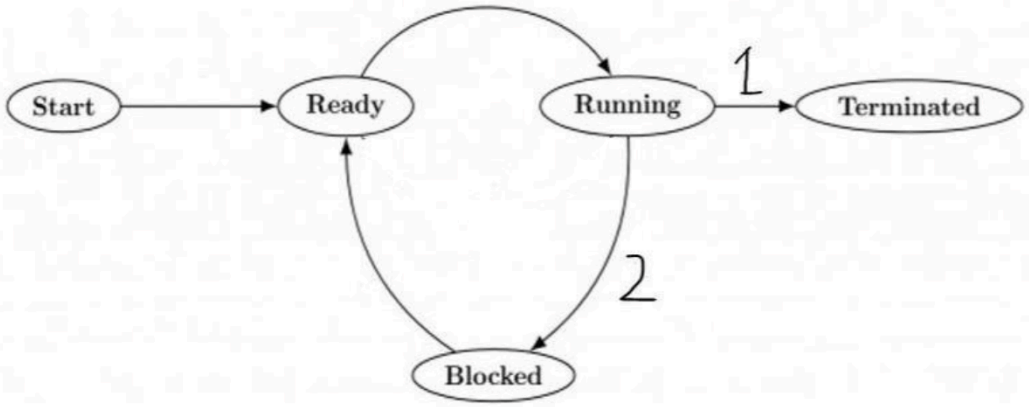
- **medium-term scheduler** The key idea behind a medium-term scheduler is that sometimes it can be advantageous to remove processes from memory for I/O operation and thus reduce the degree of multiprogramming. Later, the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called swapping. The process is swapped out, and is later swapped in, by the medium-term scheduler. Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up.



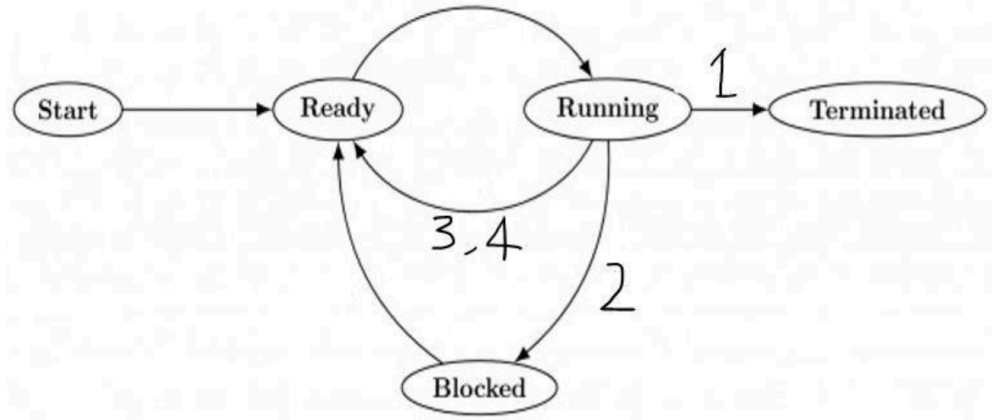
Scheduling Algorithms

- A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.
- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

- These algorithms are either **non-preemptive or preemptive**.
- Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time
- preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.



Non - Preemptive Scheduling



Preemptive Scheduling

Scheduling Criteria

- **Arrival Time:** Time at which the process arrives in the ready queue.
- **Completion Time:** Time at which process completes its execution.
- **Burst Time:** Time required by a process for CPU execution.
- **Turn Around Time:** Time Difference between completion time and arrival time.
  Turn Around Time = Completion Time – Arrival Time

- **Waiting Time(W.T):** Time Difference between turn around time and burst time.
  Waiting Time = Turn Around Time – Burst Time
- **ResponseTime(R.T) :** The time at which a process got CPU first time - Arrival time