Operating system: is a program that acts as an *intermediary* between users/applications and the computer hardware.

OS is a fundamental and essential component of modern computing systems. It is a software program that acts as an intermediary between the hardware of a computer system and the applications or user-level programs running on that computer. it manages the computer's resources and provides a platform for applications to run effectively and efficiently.

GOALS OF OS

1. **Convenience**: The primary goal of an OS is to make the computer system convenient to use, also Provides a *layer of abstraction* for hardware resources Allows user programs to deal with higher-level, simpler, and more portable concepts than the raw hardware • E.g., files rather than disk blocks, Makes finite resources seem "infinite" .
   - Convenience refers to the ease and user-friendliness with which users can interact with the computer and perform tasks. OS attempt to provide a convenient computing environment that simplifies various operations, reduces complexity, and enhances user productivity
2. **Efficiency**: A secondary goal of an OS is to use the computer hardware in an efficient manner, ensuring that no user gets more than his fair share.
   - Efficiency, relates to optimizing the use of computer resources to achieve maximum performance, responsiveness, and throughput. An efficient operating system ensures that computing tasks are executed swiftly and effectively, minimizing resource wastage and enhancing overall system performance

Basic components of a computer system: place of OS: A computer system can be divided into four components

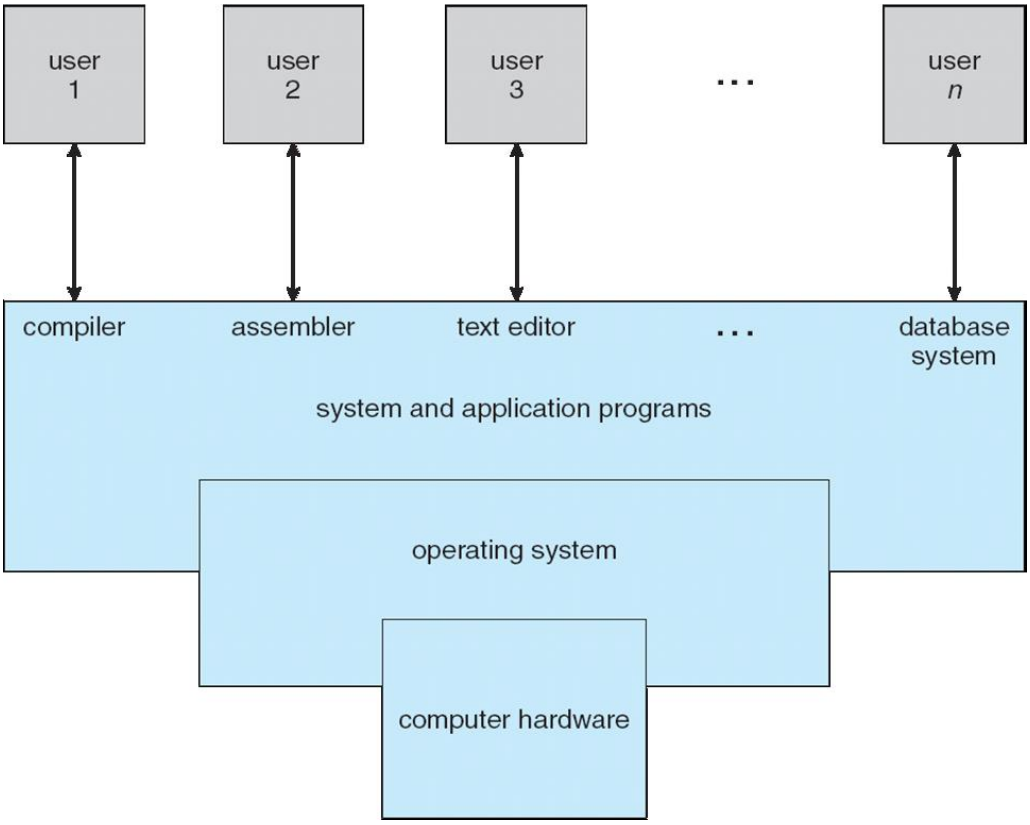
1. Hardware – provides basic computing resources example CPU, memory, I/O devices

2. Operating system that Controls and coordinates use of hardware among various applications and users

3. Application programs –solve the problems of the users: use system resources

- Word processors, compilers, web browsers, database systems, video games

4. Users -People, machines, other computers



Why study Operating Systems?

- **Need to understand interaction between the hardware and applications**

New applications, new hardware.

Inherent aspect of society today

- **Need to understand basic principles in the design of computer systems**

efficient resource management, security, flexibility

- **Increasing need for specialized operating systems**

e.g. embedded operating systems for devices - cell phones, sensors and controllers

studying OS is crucial for gaining a comprehensive understanding of computing, enabling efficient resource management, and fostering innovation in the rapidly evolving world of technology.

Computer System Organization

- **Computer-system operation**

A modern Computer System consist of One or more CPUs, and a number of device controllers connected through common bus that provide access to shared memory

The CPU and the Device controllers can execute concurrently competing for memory cycles

Computer system operation: I/O and device interaction

- **I/O devices and the CPU can execute concurrently**

- **Each device controller is in charge of a specific device and has a local memory call buffer**
- **I/O is from the device to local buffer of controller**
- **CPU moves data from/to main memory to/from the local buffers**
- **device controller informs CPU that it has finished its current operation or it has something by causing an *interrupt***

INTERRUPTS

A mechanism of informing the CPU that something has happened and you need to do something. an interrupt is a signal or event generated by hardware or software that temporarily halts the normal flow of the CPU's execution to handle a specific task or event.

Interrupt could be caused by hardware or software

- Hardware interrupt – interrupts generated by hardware devices
- Software interrupts also known as traps – interrupts generated by a software due to an error or user requests

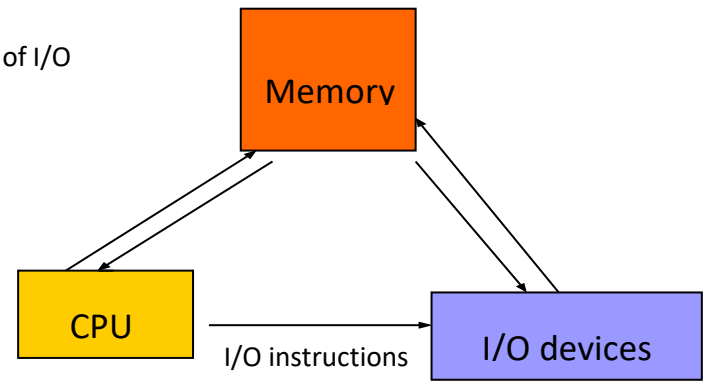
When interrupt occurs, hardware does the following:

- CPU is interrupted
  at that time application code or kernel code might be running
  registers and the program counter saved in RAM to preserve CPU state
  CPU starts running the respective Interrupt Service Routing (ISR)
  (kernel routine) ISR is found through **interrupt vector**

Direct Memory Access (DMA)

(DMA) is a computer system feature that allows certain hardware devices, such as disk controllers or network adapters, to access the computer's memory directly without involving the central processing unit (CPU). DMA enables high-speed data transfers between these devices and the system memory, reducing the CPU's involvement in the data transfer process and improving overall system performance.

- Typically used for I/O devices with a lot of data to transfer (in order to reduce load on CPU).
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Device controller interrupts CPU on complete on of I/O



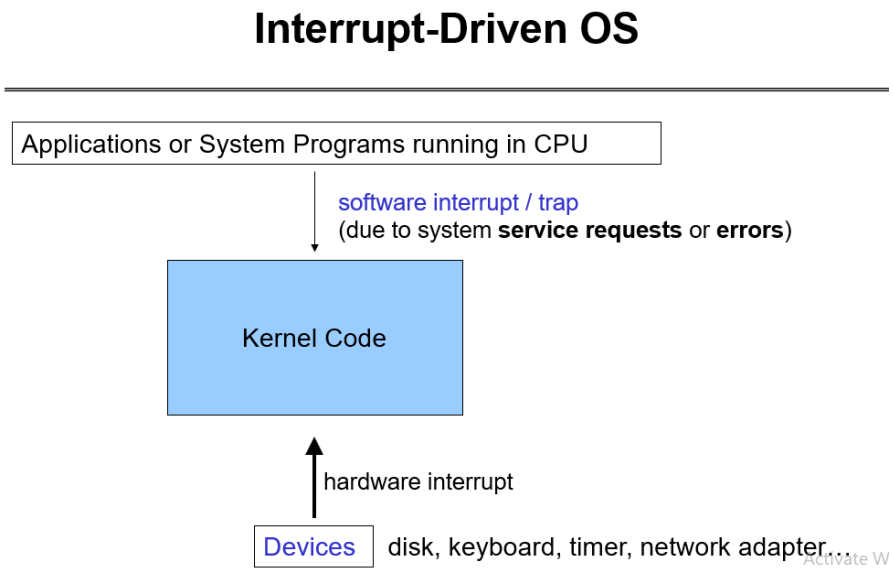
Interrupt-Driven OS

- An operating system (kernel) is interrupt-driven (event driven)

i.e. it executes only when an event occurs which is usually signaled by an interrupt.

- For each type of interrupt, a separate segment of code in the OS (i.e. ISR) is provided to handle the interrupt.

An Interrupt-Driven OS: is an operating system design that relies heavily on the use of interrupts to manage and respond to various events and tasks in the computer system



TYPES OF OPERATING SYSTEM

1. **Single-User, Single-Tasking OS:** These operating systems allow only one user to perform one task at a time. They are simple and found in early personal computers

2. **Single-User, Multi-Tasking OS:** These OSes permit one user to run multiple applications or tasks simultaneously, switching between them quickly.

3. **Multi-User OS:** These operating systems support multiple users to log in and use the system concurrently. Each user can have separate accounts, preferences, and data

4. **Real-Time OS (RTOS):** RTOSes are designed for systems that require precise and predictable response times.

5. **Distributed OS:** Distributed OSes run on a network of interconnected computers and enable them to work together as a single system

6. **Embedded OS:** These operating systems are designed for embedded systems with limited resources and specific purposes, such as microcontrollers, IoT devices, and consumer electronics

7. **Mobile OS:** Mobile operating systems power smartphones, tablets, and other mobile devices

8. **Virtualization OS:** These OSes enable virtualization, allowing multiple virtual machines to run on a single physical machine.

9. **Batch Processing OS:** These operating systems process a set of tasks or jobs in batches without user intervention

10. **Time-Sharing OS:** Time-sharing OSes enable multiple users to interact with a computer simultaneously, providing each user with a time slice or portion of CPU time

What happens to a new interrupt when the CPU is handling one interrupt?

Here is how it works:

1. **Interrupt Prioritization:** Different interrupts may have different priorities assigned to them. When a new interrupt occurs, the system checks its priority level compared to the currently executing interrupt

2. **Interrupt Masking:** If the new interrupt has a higher priority than the one currently being handled; the CPU may temporarily mask or disable lower-priority interrupts. This ensures that the CPU prioritizes the higher-priority interrupt over the current one.

3. **Interrupt Handling Routine:** The CPU saves the state of the current interrupt handling routine, including the program counter, registers, and other relevant information, into a data structure

4. **Context Switching:** The CPU then switches to the interrupt handling routine of the higher-priority interrupt, known as the Interrupt Service Routine (ISR). The ISR handles the new interrupt

5. **Resuming Interrupt Handling:** After the ISR completes its task, the CPU performs a context switch back to the original interrupt handling routine, restoring the saved state. The CPU can now resume the processing of the original interrupt from where it left off

6. **Interrupt Unmasking:** Once the CPU has completed handling the higher-priority interrupt and resumed processing the original interrupt, it may unmask or enable lower-priority interrupts again

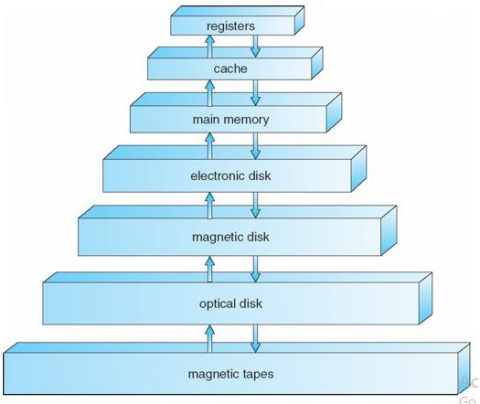
Storage Structure

- **Main memory** – CPU can access directly: the CPU accesses RAM through a memory bus and a memory controller. It sends memory addresses and control signals to read or write data from/to specific locations in RAM. Modern CPUs also have cache memory to reduce memory access latency
- **Secondary storage** – extension of main memory that provides large nonvolatile storage capacity
    - Magnetic disks
        - platters
        - The **disk controller** determines the interaction between the device and the computer

secondary storage is a type of storage that supplements the main memory of a computer system. It provides a large storage capacity that retains data even when the computer is powered off. This storage is often achieved using magnetic disks, which are circular plates

Storage Hierarchy

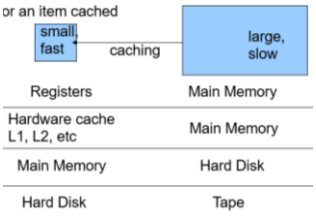
- Storage systems **organized in hierarchy**
  - Speed, Cost, Volatility



Caching

caching is a technique used to accelerate data access by storing frequently used data in faster, smaller storage areas closer to the CPU

- **Caching** – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage



Computer System Architecture:

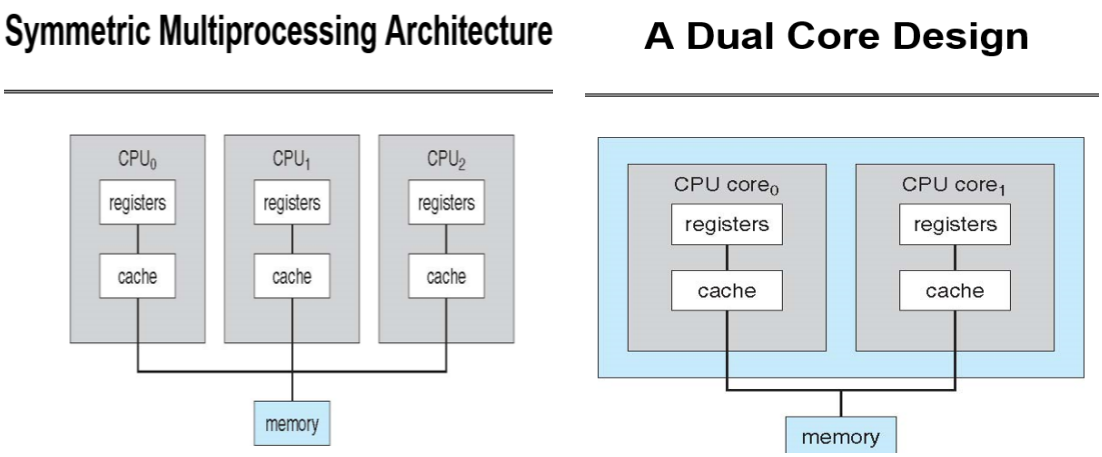
- **Advantages include**
  1. **Increased throughput**
  2. **Economy of scale (cheaper than using multiple computers)**
  3. **Increased reliability – graceful degradation or fault tolerance**

Single processor systems

single processor systems are computer architectures centered around a single CPU. They are straightforward in design, cost-effective, and compatible with a broad range of software applications.

Multiprocessor systems

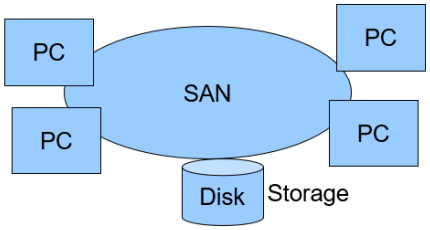
multiprocessor systems utilize multiple CPUs to achieve concurrent execution, parallelism, and improved performance. They are suitable for tasks that can be divided into parallelizable components and require efficient multitasking.



Clustered Systems

is a type of computing environment where multiple individual computers, referred to as nodes, work together as a single integrated system

a clustered system involves multiple interconnected computers working together to provide improved performance, high availability, and fault tolerance



Operating System Structures and operations

Operating System Structures refer to the organization and design principles that govern how an OS is structured and how its components interact

OS Operations encompass the core functions and services provided by an operating system to manage hardware, software, and user interactions

Operating Systems Structure: Concepts

- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

Response time should be < 1 second

program loaded in memory ⇨**process**

If several processes ready to run at the same time ⇨ **CPU scheduling**

Operating System operations: Interrupt Driven

- The Operating System is interrupt driven?

If there are no processes to execute, no I/O devices to service, and no users to whom to respond, the operating system will sit quietly, waiting for something to happen. Events are almost always signaled by the occurrence of an interrupt or a trap

Hardware interrupt causes ISR to run (which is a routine of OS)

Software error or request creates **exception** or **trap**

Division by zero, for example (exception)

request for an operating system service (trap)

For each type of interrupt, separate segments of code in the operating system determine what action should be taken. An interrupt service routine is provided to deal with the interrupt

Operating System Operations: Dual Mode

…

Operating System Functions

- **Process Management**
    - **Process creation and deletion**
    - **Process suspension and resumption**
    - **Process synchronization and inter-process communication**
    - **Process interactions - deadlock detection, avoidance and correction**

- **Memory management**
    - **Allocating and de-allocating memory to processes.**
    - **Managing multiple processes within memory - keep track of which parts of memory are used by which processes. Manage the sharing of memory between processes.**
    - **Determining which processes to load when memory becomes available.**
- **Storage (disk) management and I/O management**
- **Protection and security**
- **File System Management**
    - **File creation and deletion**
    - **Directory creation and deletion**
    - **Supporting primitives for file/directory manipulation.**
    - **Mapping files to disks (secondary storage).**

OS Services

- **The operating system provides the environment for the execution of programs. It provides certain services for users and programs of the users.**

I/O operations – OS provides user apps means to do I/O

Communications – OS allow user processes to communicate through shared memory or message passing

Error Detection – detecting and correcting errors that may occurs in hardware or user programs

Resource Allocation – allocate resources to multiple processes running at the same time.

User Interface

Protection and security

Accounting

PROCESS

A process is basically a running program that's being executed by the CPU. It includes all the data and code that the program needs to run, as well as a bunch of info that helps the OS manage it.

- a process is a program in the state of execution
- or an instance of a program running on a computer
- or the entity that can be assigned to and executed on a processor
- or it is a unit of activity characterized by the execution of a sequence of instruction, a current state and an associated set of system resources.

A process is an entity that consist of a number of elements. And these essential elements are

- program code (which may be shared with other processes that are executing the same program)
- set of data associated with that code
- and a process control block

a process can be characterized by the following elements:

- **identifier:** a unique identifier associated with the process, to distinguish it from other processes
- **State:** if the process is executing, it is in the running state
- **Priority:** priority level to other processes
- **Program counter:** the address of the next instruction in the program to be executed

- **Memory pointers:** includes pointer to the program code, and data associated with this process, plus any memory block shared with other processes.
- **Context data:** the data that are present in the register in the processor while the process is executing.
- **I/O status information:** include outstanding I/O request, I/O devices assigned to the process and so on.
- **Accounting information:** include the amount of processor time and clock time used and so on

We can store the above elements in a data structures called the **process block control**

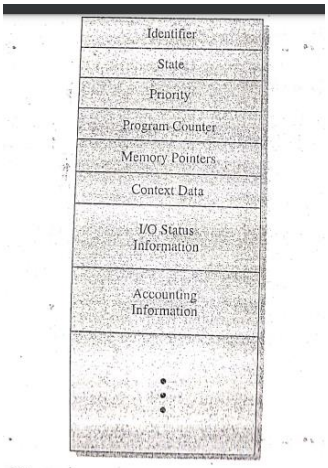


Figure 3.1 Simplified Process Control Block

process block control is the key tool that enables the OS to support multiple processes and to provide for multiprocessing. It also refers to the current condition or status of a process as it executes on the system.

- The significant point about **process block control** is that it contains sufficient information so that it is possible to interrupt a running process and later resume execution as if the interrupt had not occurred.

Trace: is the act of listening the sequence of instruction that execute for a particular process.

PROCESS STATE

Process state are the different stages that a process goes through during its lifetime.

Here are the main states:

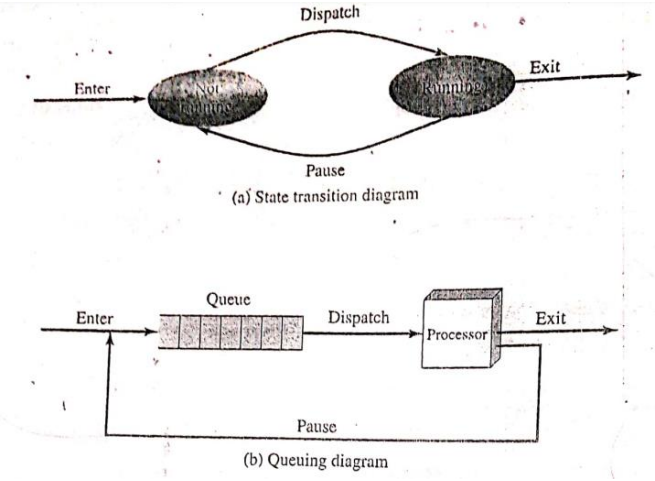
- **New:** a process that has just been created but hasn't started running.
- **Ready:** a process that's ready to run, but it is waiting for the CPU to become available.
- **Running:** a process that's actively executing instruction on the CPU.
- **Blocked:** a process that's waiting for a resource, like I/O or a lock.

- **Terminated:** a process that's done its thing and is now terminated.

A Two Process Model:

A process may only be in two state; either **running** or **not running**. It is a simplified representation of a process lifecycle.

- **Running State**: When a process is in the running state, it is actively executing on the CPU. It can perform computations and access resources as needed
- **Not Running State**: Processes enter the not running state when they cannot proceed further because they are waiting for some event or resource. When a process is in the blocked state, it is not actively using CPU time, and it remains in this state until the event or resource it is waiting for becomes available.



Process Creation and Termination

Process creation: is the act of spawning a new process in the OS, allocating resources like memory and setting up its environment.

Table 3.1 Reasons for Process Creation

New batch job	The OS is provided with a batch job control stream, usually on tape or disk. When the OS is prepared to take on new work, it will read the next sequence of job control commands.
Interactive logon	A user at a terminal logs on to the system.
Created by OS to provide a service	The OS can create a process to perform a function on behalf of a user program, without the user having to wait (e.g., a process to control printing).
Spawned by existing process	For purposes of modularity or to exploit parallelism, a user program can dictate the creation of a number of processes.

Process termination: is the end of a process's execution, where it releases its resources, exit its main function, and is removed from the system.

Process spawning: is the act, when the OS creates a process at the explicit request of another process.

Five State Process Model

Is another way to describe the various state a process can be during its lifecycle. The five states are:

- **New:** a process that is being created but hasn't started executing
- **Ready:** a process that's ready to run, but it is waiting for a turn on the CPU
- **Running:** a process that's currently executed by the CPU.
- **Blocked:** a process that's waiting for a resource, like I/O or a lock.
- **Exit:** a process that has been released from the pool of executable processes by an OS

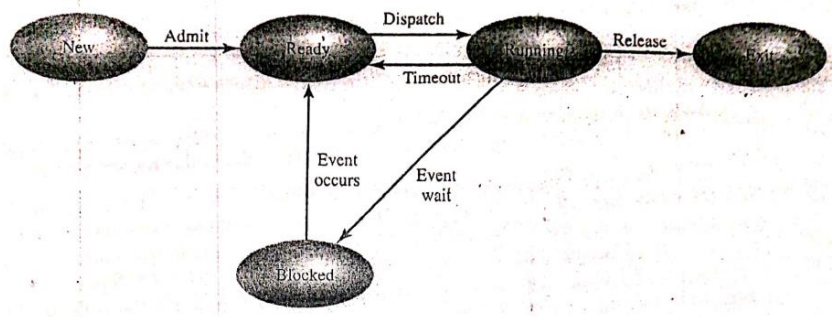
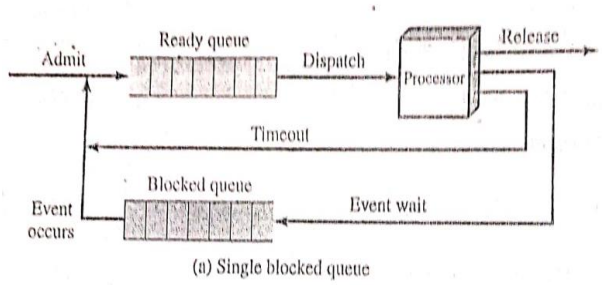


Figure 3.6 Five-State Process Model

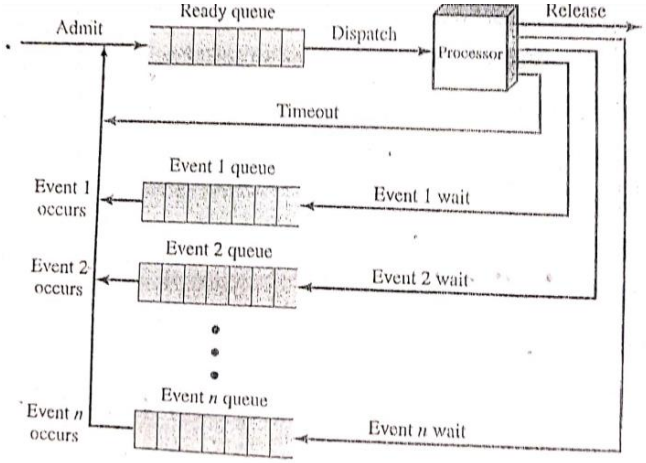
Possible Transitions in the five processes model:

- **Null-> New:** a new process is created to execute a program
- **New-> Ready:** the OS will move a process from new to ready state when it is prepared to take on an additional process
- **Ready-> Running:** the OS choose one of the processes in the ready state when it is time for a process to run
- **Running-> Exit:** when the process is completed, it is terminated by the OS
- **Running-> Ready:** A process in the **Running** state may transition back to the **Ready** state for various reasons, such as when its time quantum expires
- **Running-> Blocked:** a process is put into blocked state if it requests something for which it must wait for.
- **Blocked-> Ready:** a process in the blocked state is moved to the ready state when the event for which it has been waiting occurs
- **Ready-> Exit:** A process in the **Ready** state may be terminated without ever entering the **Running** state if it is removed from the queue before it gets a chance to execute.

- **Blocked-> Exit:** the comment under the preceding item apply. If a process in the **Blocked** state is terminated for any reason while waiting for an event, it moves directly to the **Exit** state.



(a) Single blocked queue



SUSPENDED PROCESSES

Swapping: is the process of moving some or all part of a process from main memory to disk. It is simply moving data temporarily from the RAM to the hard drive when there is not memory available.

The need for swapping: it allows the computer to keep more processes running at the same time even if the available memory is limited.

- With the use of swapping, one state must be added to our process behavior model **The Suspended State.**

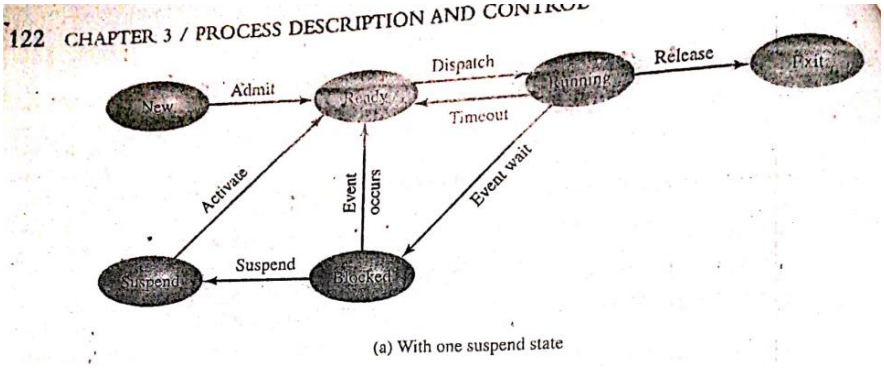
Possible state when a suspended state is added:

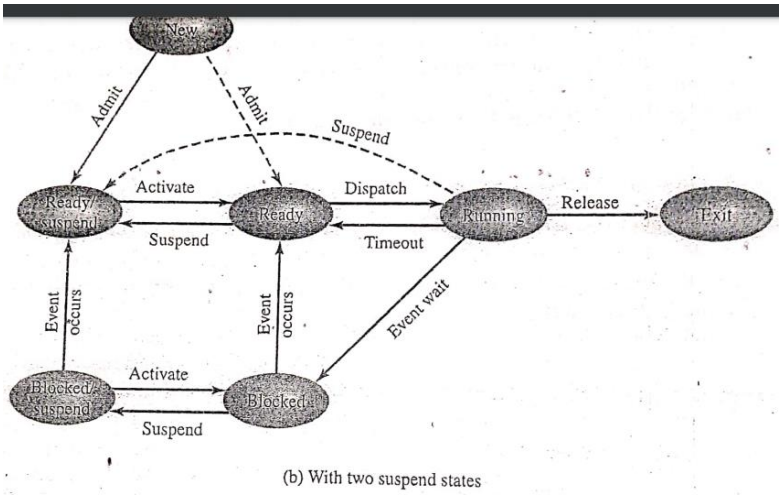
- **New**: This is the initial state when a process is first created.
- **Ready**: After the process has been created and initialized
- **Running**: When the operating system scheduler assigns the CPU to a process from the ready queue, it enters the running state.

- **Blocked (or Waiting)**: Processes often need to wait for various events, such as input/output operations (I/O)
- **Terminated (or Exit)**: When a process completes its execution or is terminated by the operating system or a user, it enters the terminated state.
- **Suspended**: The "Suspended" state is a special state that indicates a process is not currently executing. There are two variations:
  - **Suspended /Ready:** the process is in secondary memory but it is available for execution as soon as it is loaded in to the main memory.
  - **Suspended/Blocked:** the process is in secondary memory and is waiting for an event.

Transitions in this state

- **Blocked to Blocked/Suspended:** if there are no ready process, then at least one blocked is swapped out to make room for another process that is not blocked.
- **Blocked /Suspended to Ready/Suspended:** a process in this state will moved to R/S state when the event for which it has been waiting occurs.
- **Ready/Suspended to Ready:** when there are no processes in the main memory, OS will bring one to continue with execution.
- **Ready to Ready/Suspended:** ready process can be suspended if it is the only way to free up sufficient space in the main memory.
- **New to Ready/Suspended and New to Ready:** new process can be added to either of the two states.
- **Blocked/Suspended to Blocked:** A process in the **Suspended Blocked** state can transition back to the **Blocked** state when the event it was waiting for is satisfied.
- **Running to Ready/Suspend:** When a process in the **Suspended Ready** state is resumed, it transitions back to the **Running** state and continues execution from where it left off.
- **Any state to exit:** a process can exit while running because it is completed or because of fault conditions.





(b) With two suspend states

Table 3.3 Reasons for Process Suspension

Swapping	The OS needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS reason	The OS may suspend a background or utility process or a process that is suspected of causing a problem.
Interactive user request	A user may wish to suspend execution of a program for purposes of debugging or in connection with the use of a resource.
Timing	A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time interval.
Parent process request	A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendants.

We can generalize the concept of a suspended process. Let us define a suspended process as having the following characteristics:

1. The process is not immediately available for execution.
2. The process may or may not be waiting on an event. If it is, this blocked condition is independent of the suspend condition, and occurrence of the blocking event does not enable the process to be executed immediately.
3. The process was placed in a suspended state by an agent: either itself, a parent process, or the OS, for the purpose of preventing its execution.
4. The process may not be removed from this state until the agent explicitly orders the removal.

PROCESSOR SCHEDULING

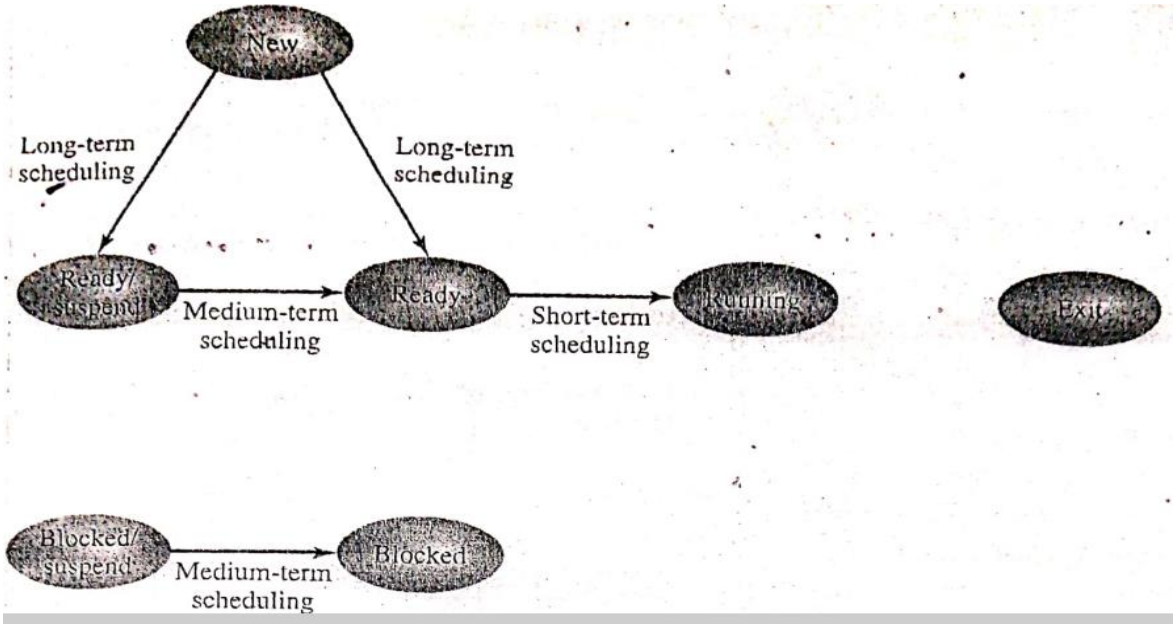
It refers to the mechanism by which the OS manages and allocates CPU time to multiple processes or that are competing for execution on a computer system. The aim of processor scheduling is to assign process to be executed by the processor over time in a way that meets system objectives.

TYPES OF PROCESSOR SCHEDULING

1. **Long-Term Scheduling:** is performed when a new process is created. This is a decision whether to add a new process to the set of processes that are currently active.
2. **Medium-Term Scheduling:** is a part of the swapping function. This is a decision whether to add a new process to those that are partially in main memory and

available for execution. It comes into play when a process is temporarily removed from the main memory (RAM) and placed in secondary storage (e.g., disk) to free up memory space.

3. **Short-Term Scheduling:** is the actual decision of which ready process to execute next.



The short-term scheduler is invoked whenever an event occurs that may lead to the blocking of the current process or that may provide an opportunity to preempt a currently running process in favor of another. Examples of such events include

- Clock interrupts
- I/O interrupts
- Operating system calls
- Signals (e.g., semaphores)

CATEGORIES OF DECISION MODE

1. **Non-Preemptive Scheduling:** once a process starts executing, it continues until it completes or block its self to wait for an I/O
2. **Preemptive Scheduling:** allows the OS to interrupt a currently executing process and allocate the CPU to another process based on certain criteria, such as priority or time quantum.

SCHEDULING ALGORITHMS

1. **First-Come-First-Served (FCFS):** In this algorithm, processes are executed in the order they arrive in the ready queue. It's a non-preemptive algorithm, meaning that once a process starts executing, it continues until it completes.
2. **Round Robin (RR):** Round Robin scheduling allocates a fixed time slice (quantum) to each process in a cyclic manner. When a time slice expires, the process is moved to the back of the queue, allowing the next process to execute. RR is a preemptive algorithm that provides fair access to the CPU for all processes.
3. **Shortest Process Next (SPN)** selects the process with the smallest execution time next. It aims to minimize the average waiting time.

Memory Management

Is the process of controlling and coordinating the computer's main memory. It ensures that blocks of memory spaces are properly allocated so the OS, applications and other running processes have the memory they need to carry out their operations.

Memory Management Techniques

Memory partitioning: means dividing the main memory into chunks of the same or different sizes so that they can be assigned to processes in the main memory.

Types of Memory partitioning

1. Fixed Partitioning: is a simplest technique used to put more than one process in the main memory. In this partitioning, the sizes of partitions are fixed. And it has two types
    - **Equal size:** here the partitions sizes are fixed and each partition will be equal. And any process less than the size can be loaded in to it.
      Problems with equal size
        1. a program may be too big to fit in to a partition. In this case, the programmer must design the program with the use of **overlay**

Overlay: is a technique used to manage memory by overlaying a portion of memory with another program.

2. main memory utilization is inefficient. Any program, no matter how small it is, it will occupy an entire partition. This causes the problem of **internal fragmentation.**

internal fragmentation: when a memory block gets allocated with a process, and it happens to be that the process is smaller than the amount of requested memory, a free space is created in this memory block. That is what causes the **internal fragmentation**

By using unequal partition these problems can lessened but not solved. But will definitely lead to a lesser internal fragmentation.

- **Unequal size:** here the partitions are fixed and each partition has a different size.

Dynamic Partitioning: are of variable length and number, when a process is brought to main memory, it is allocated exactly as much memory as it requires.

Problem with dynamic partitioning: **External fragmentation**

External fragmentation occurs whenever a method of dynamic memory allocation happens to allocate some memory and leave a small amount of unusable memory.

It occurs when memory is divided into variable size partition based on the size of the processes.

Solution to **External fragmentation** is **Compaction**

Compaction: all the free partitions are made contiguous and all the loaded partitions are brought together

The difficulty with **Compaction** is: it is time consuming procedure and wasteful of CPU time

Placement Algorithms

- **Best Fit:** will search the entire list of available blocks and chooses the block that is closest in size to the request.
- **First fit:** begin to scan memory from the beginning and chooses the first available block that is large enough
- **Next Fit:** begin to scan memory from the location of last placement, and chooses the next available block that is large enough.

PAGING

is the process whereby processes are divided into small fixed sized chunks of the same size.

FRAMES

Is when the main memory is partitioned into equal fixed size chunks that are relatively small.

SEGMENTATION

Segmentation is another memory management technique used in OS. it divides memory into variable-sized segments based on logical or functional units

segmentation is a memory management technique that divides a process's logical address space into variable-sized segments, each with its own properties, permissions, and base address

- it suffers from external fragmentation.

HOW DOES PAGING DIFFER FROM SEGMENTATION?

1. Division of Memory:
    - **Paging:** divides memory into fixed-size blocks called pages. Pages are typically of equal size.
    - **Segmentation:** Segmentation divides the logical address space of a process into variable-sized segments
2. Size of Memory Units:
    - **Paging:** Pages are of the same size throughout the entire memory space
    - **Segmentation:** Segments can vary in size
3. Fragmentation:
    - **Paging:** Paging eliminates external fragmentation because pages are of uniform size
    - **Segmentation:** Segmentation can lead to external fragmentation because segments can have different sizes

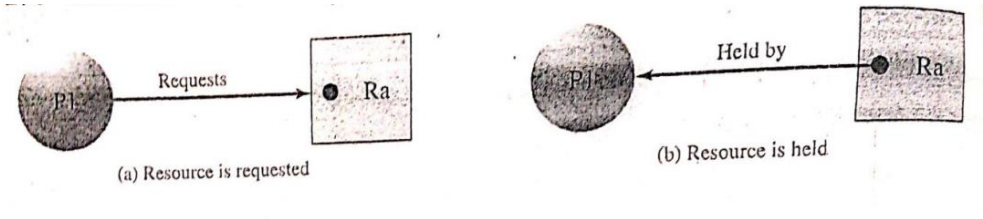
PRINCIPLE OF DEADLOCK

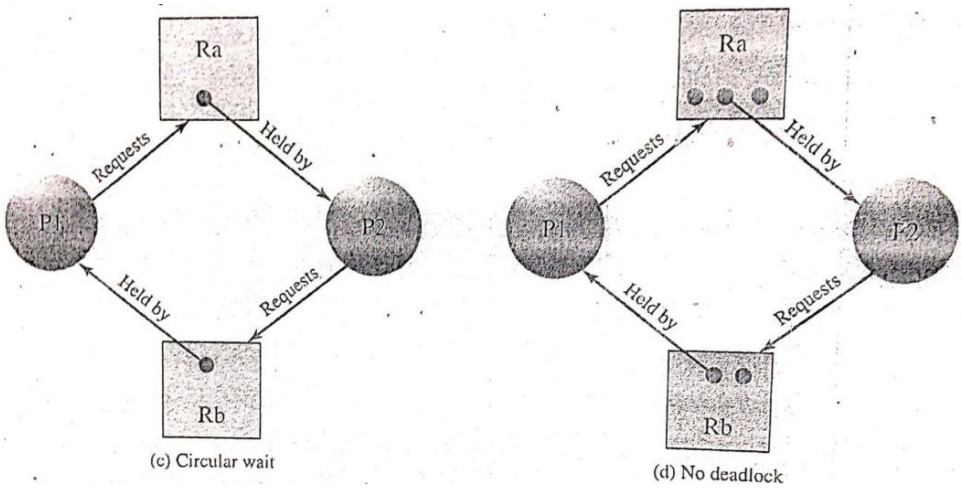
Deadlock is defined as the permanent blocking of a processes that either compete for system resources or communicate with each other.

Deadlock: is a state where two or more processes are unable to proceed because each is waiting for a resource held by another, resulting in a standstill in the system's operation.

RESOURCE ALLOCATION GRAPH: is a useful tool in characterizing the allocation of resources to processes.

is a graphical representation that is used to track and analyze resource allocation and potential deadlocks. It is a tool to visualize and understand the relationships between processes and the resources they request and hold. Below are examples of RAG:





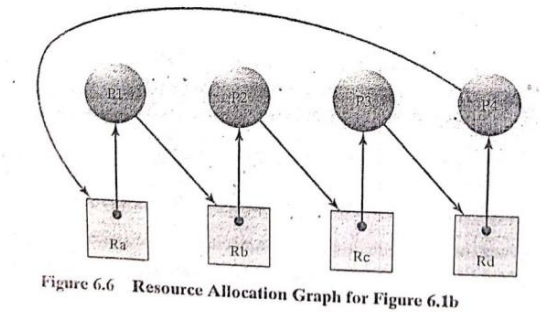


Figure 6.6 Resource Allocation Graph for Figure 6.1b

FOUR NECESSARY CONDITIONS FOR DEADLOCK

1. **Mutual Exclusion**: only one process may use a resource at a time. No process may access a resource unit that has been allocated to another process.
2. **Hold and Wait**: A process may hold allocated resource while awaiting assignment of other resources.
3. **No Preemption**: no resources can be forcibly removed from a process holding it

The first 3 conditions are necessary but not sufficient for a deadlock to occur. For a deadlock to take place, the fourth condition is required:

4. **Circular Wait:** A cycle of processes exists, where each process is waiting for a resource held by the next process in the cycle.

APPROACHES FOR DEALING WITH DEADLOCK

1. Deadlock prevention
2. Deadlock Avoidance
3. Deadlock detection

DEADLOCK PREVENTION

Deadlock prevention is a strategy designed to eliminate one or more of the necessary conditions for a deadlock to occur, thereby preventing deadlocks from happening in the first place.

Direct Method: is to prevent the occurrence of a circular wait

Indirect Method: is to prevent the occurrence of one of the necessary conditions

DEADLOCK AVOIDANCE

Deadlock Avoidance: is a strategy that manages resource requests to prevent the possibility of deadlocks by satisfying resource allocation requests in a way that avoids circular waiting among processes.

TWO APPROACHES TO DEADLOCK AVOIDANCE

- Process initiation denial
- Resource allocation denial

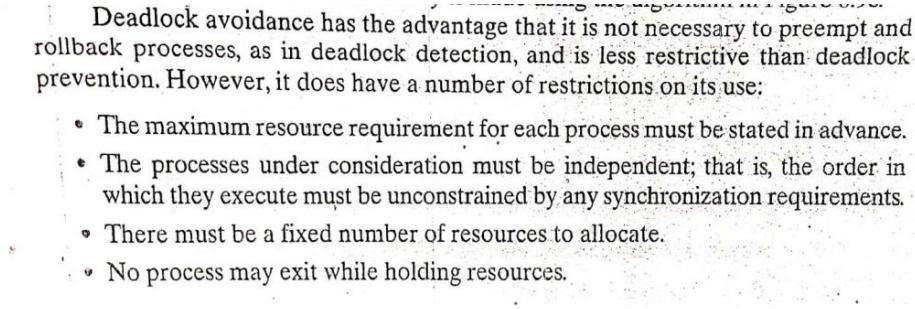
Process initiation denial: In this approach, the OS controls the initiation of new processes to ensure that the system does not become overloaded with too many processes simultaneously requesting resources

Resource allocation denial: this strategy is referred to as the banker's algorithm. This approach involves denying resource allocation requests if granting them would lead to an unsafe state in terms of potential deadlocks.

- State: reflect the current allocation of resources to processes. The state consists of the two vectors **Resources and Available,** and the 2 matrices **Claim and Allocation.**

Safe state: is one in which, there is at least one sequence of resource allocation to processes that does not result in a deadlock.

unsafe state: is of course a state that is not safe.



Deadlock avoidance has the advantage that it is not necessary to preempt and rollback processes, as in deadlock detection, and is less restrictive than deadlock prevention. However, it does have a number of restrictions on its use:

- The maximum resource requirement for each process must be stated in advance.
- The processes under consideration must be independent; that is, the order in which they execute must be unconstrained by any synchronization requirements.
- There must be a fixed number of resources to allocate.
- No process may exit while holding resources.

DEADLOCK DETECTION: solve the problem of deadlock by limiting access to resources and by imposing restrictions on processes.

OTHER POINTS TO TAKE NOTE

1.

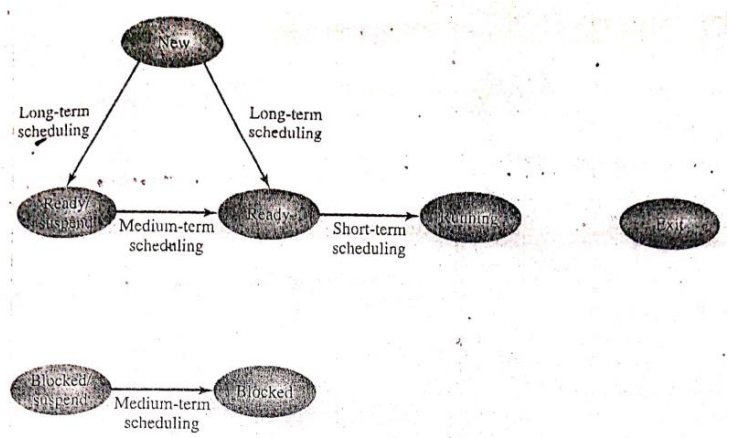


Figure 9.1 Scheduling and Process State Transitions

2.

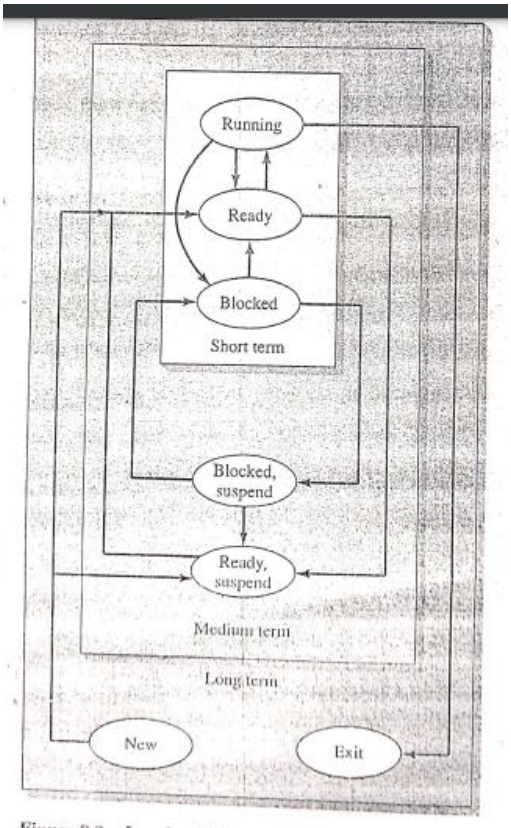


Figure 9.2 Levels of Scheduling

3.

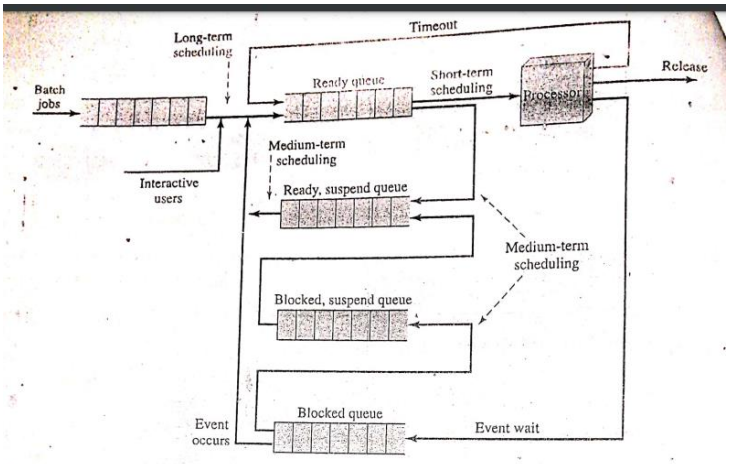


Figure 9.3 **Queuing Diagram for Scheduling**

4.

Possibility of Deadlock	Existence of Deadlock
1. Mutual exclusion	1. Mutual exclusion
2. No preemption	2. No preemption
3. Hold and wait	3. Hold and wait
	4. Circular wait

5.

Table 7.2 Memory Management Techniques

Technique	Description	Strengths	Weaknesses
Fixed Partitioning	Main memory is divided into a number of static partitions at system generation time. A process may be loaded into a partition of equal or greater size.	Simple to implement; little operating system overhead.	Inefficient use of memory due to internal fragmentation; maximum number of active processes is fixed.
Dynamic Partitioning	Partitions are created dynamically, so that each process is loaded into a partition of exactly the same size as that process.	No internal fragmentation; more efficient use of main memory.	Inefficient use of processor due to the need for compaction to counter external fragmentation.
Simple Paging	Main memory is divided into a number of equal-size frames. Each process is divided into a number of equal-size pages of the same length as frames. A process is loaded by loading all of its pages into available, not necessarily contiguous, frames.	No external fragmentation.	A small amount of internal fragmentation.

Simple Segmentation	Each process is divided into a number of segments. A process is loaded by loading all of its segments into dynamic partitions that need not be contiguous.	No internal fragmentation; improved memory utilization and reduced overhead compared to dynamic partitioning.	External fragmentation.
Virtual Memory Paging	As with simple paging, except that it is not necessary to load all of the pages of a process. Nonresident pages that are needed are brought in later automatically.	No external fragmentation; higher degree of multiprogramming; large virtual address space.	Overhead of complex memory management.
Virtual Memory Segmentation	As with simple segmentation, except that it is not necessary to load all of the segments of a process. Nonresident segments that are needed are brought in later automatically.	No internal fragmentation, higher degree of multiprogramming; large virtual address space; protection and sharing support.	Overhead of complex memory management.