Operating System Deadlocks

Prepared By:

Dr. Sanjeev Gangwar

Assistant Professor,

Department of Computer Applications,

VBS Purvanchal University, Jaunpur

Deadlocks

In a multiprogramming environment several processes may compete for a finite number of resources. A process request resources; if the resource is available at that time a process enters the wait state. Waiting process may never change its state because the resources requested are held by other waiting process. This situation is known as deadlock.

Generally, a system has a finite set of resources (such as memory, IO devices, etc.) and a finite set of processes that need to use these resources. A process which wishes to use any of these resources makes a request to use that resource. If the resource is free, the process gets it. If it is used by another process, it waits for it to become free. The assumption is that the resource will eventually become free and the waiting process will continue on to use the resource. But what if the other process is also waiting for some resource?

"A set of processes is in a deadlock state when every process in the set is waiting for an event that can only be caused by another process in the set."

If a process is in the need of some resource, physical or logical, it requests the kernel of operating system. The kernel, being the resource manager, allocates the resources to the processes. If there is a delay in the allocation of the resource to the process, it results in the idling of process. The deadlock is a situation in which some processes in the system faces indefinite delays in resource allocation.

System Model: A process must request a resource before using it, and must release the resource after using it. A process may request as many resources as it requires to carry out its designated tasks. The number of resources requested may not exceed the total number of resources available in the system.

Under the normal mode of operation, a process may utilize a resource in only the following sequence:

- **Request:** if the request cannot be granted immediately then the requesting process must wait until it can acquire the resource.
- **Use:** In this state the process operates on the resource.
- **Release:** the process releases the resource.

To illustrate a deadlocked state, consider a system with 3 CDRW drives. Each of 3 processes holds one of these CDRW drives. If each process now requests another drive, the 3 processes will be in a deadlocked state. Each is waiting for the event "CDRW is released" which can be caused only by one of the other waiting processes. This example illustrates a deadlock involving the same resource type.

Deadlocks may also involve different resource types. Consider a system with one printer and one DVD drive. The process Pi is holding the DVD and process Pj is holding the printer. If Pi requests the printer and Pj requests the DVD drive, a deadlock occurs.

Preemptable and Nonpreemptable Resources: Resources come in two flavors: preemptable and nonpreemptable. A preemptable resource is one that can be taken away from the process with no ill effects. Memory is an example of a preemptable resource. On the other hand, a nonpreemptable resource is one that cannot be taken away from process (without causing ill effect). For example, CD resources are not preemptable at an arbitrary moment.

Reallocating resources can resolve deadlocks that involve preemptable resources. Deadlocks that involve nonpreemptable resources are difficult to deal with.

Deadlock Characterization: In deadlock, processes never finish executing and system resources are tied up, preventing other jobs from ever starting.

Necessary Conditions: A deadlock situation can arise if the following four conditions hold simultaneously in a system

- **Mutual exclusion:** At least one resource must be held in a nonsharable mode; that is, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.
- **Hold and wait:** There must exist a process that is holding at least one resource and is waiting to acquire additional resources that are currently being held by other processes.
- **No preemption:** Resources cannot be preempted; that is, a resource can be released only voluntarily by the process holding it, after that process, has completed its task.
- **Circular wait:** There must exist a set {P0, P1, ..., Pn } of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, …., Pn-1 is waiting for a resource that is held by Pn, and Pn is waiting for a resource that is held by P0.

As an example, consider the traffic deadlock in the following Figure (1):

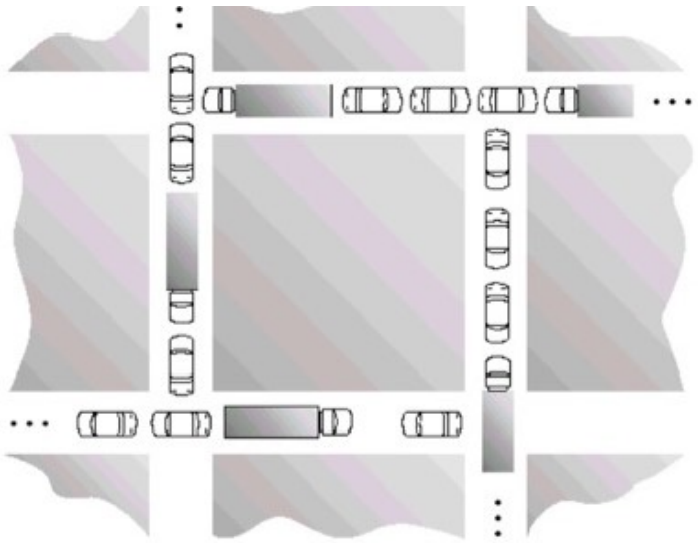


Fig 1: traffic example

Consider each section of the street as a resource.

- Mutual exclusion condition applies, since only one vehicle can be on a section of the street at a time.
- Hold-and-wait condition applies, since each vehicle is occupying a section of the street, and waiting to move on to the next section of the street.
- No-preemptive condition applies, since a section of the street that is occupied by a vehicle cannot be taken away from it.
- Circular wait condition applies, since each vehicle is waiting on the next vehicle to move. That is, each vehicle in the traffic is waiting for a section of street held by the next vehicle in the traffic.

The simple rule to avoid traffic deadlock is that a vehicle should only enter an intersection if it is assured that it will not have to stop inside the intersection.

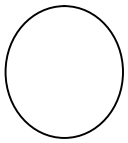
It is not possible to have a deadlock involving only one single process. The deadlock involves a circular "hold-and-wait" condition between two or more processes, so "one" process cannot hold a resource, yet be waiting for another resource that it is holding. In addition, deadlock is not possible between two threads in a process, because it is the process that holds resources, not the thread that is, each thread has access to the resources held by the process.

Resource-Allocation Graph: Deadlocks can be described more precisely in terms of a directed graph called a system resource allocation graph. This graph consists of a set of vertices V and a set of edges E. the set of vertices V is partitioned into 2 different types of nodes: P = {P1, P2….Pn}, the set consisting of all the active processes in the system. R= {R1, R2….Rm}, the set consisting of all resource types in the system.

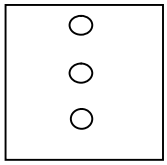
A directed edge from process Pi to resource type Rj is denoted by Pi ->Rj. It signifies that process Pi has requested an instance of resource type Rj and is currently waiting for that resource. A directed edge from resource type Rj to process Pi is denoted by Rj ->Pi, it signifies that an instance of resource type Rj has been allocated to process Pi. A directed edge Pi ->Rj is called a request edge. A directed edge Rj->Pi is called an assignment edge.

We represent each process Pi as a circle, each resource type Rj as a square. Since resource type Rj may have more than one instance. We represent each such instance as a dot within the square. A request edge points to only the rectangle Rj. An assignment edge must also designate one of the dots in the square.

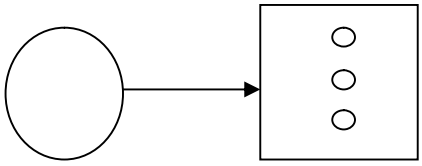
Process



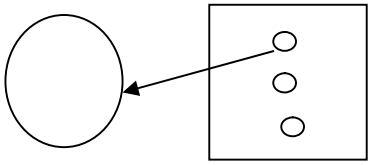
Resource type with three instances



Pi request instance of Rj



Pi is holding an instance of Rj



When process Pi requests an instance of resource type Rj, a request edge is inserted in the resource allocation graph. When this request can be fulfilled, the request edge is instantaneously transformed to an assignment edge. When the process no longer needs access to the resource, it releases the resource, as a result, the assignment edge is deleted.

The resource-allocation graph shown in Figure 2 depicts the following situation

The sets P, R, E:

- P= {P1, P2, P3}
- R= {R1, R2, R3, R4}
- E= {P1 ->R1, P2 ->R3, R1 ->P2, R2 ->P2, R2 ->P1, R3 ->P3}

Resource instances:

- One instance of resource type R1
- Two instances of resource type R2
- One instance of resource type R3
- Three instances of resource type R4

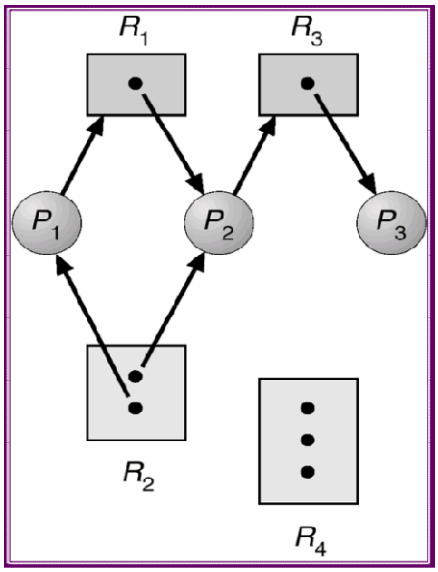


Fig 2: Resource-allocation graph

Process States:

- Process P1 is holding an instance of resource type R2 and is waiting for an instance of resource type R1.
- Process P2 is holding an instance of R1 and an instance of R2 and is waiting for instance of R3.
- Process P3 is holding an instance of R3.

If the graph contains no cycles, then no process in the system is deadlocked. If the graph does contain a cycle, then a deadlock may exist.

Suppose that process P3 requests an instance of resource type R2. Since no resource instance is currently available, a request edge P3 ->R2 is added to the graph (Figure 3). At this point, two minimal cycles exist in the system

P1 ->R1 ->P2 ->R3 ->P3 ->R2 - >P1

P2 ->R3 ->P3 ->R2 ->P2

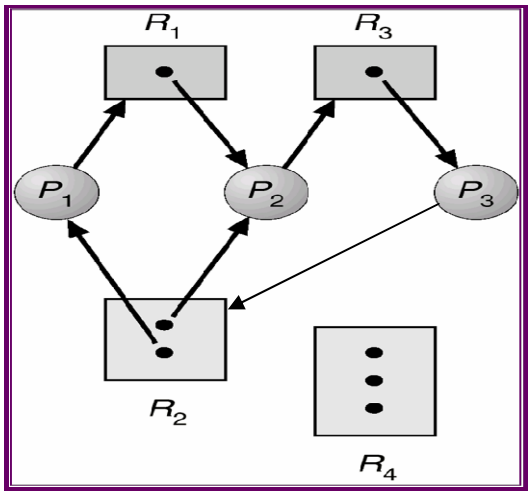


Fig 3: Resource-allocation graph with a deadlock

Processes P1, P2, P3 are deadlocked. Process P2 is waiting for the resource R3, which is held by process P3.process P3 is waiting for either process P1 or P2 to release resource R2. In addition, process P1 is waiting for process P2 to release resource R1.

Now consider the resource-allocation graph in Figure 4. In this example we also have a cycle

P1 ->R1 ->P3 ->R2 ->P1

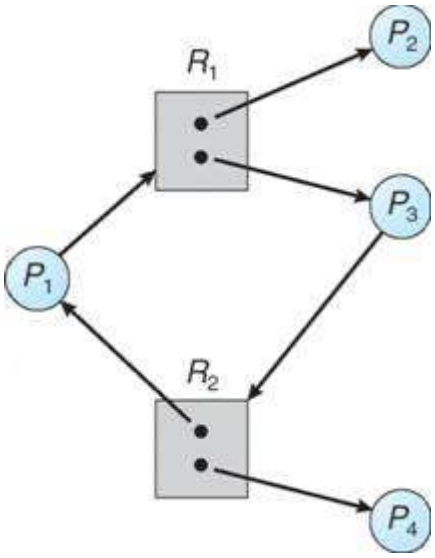


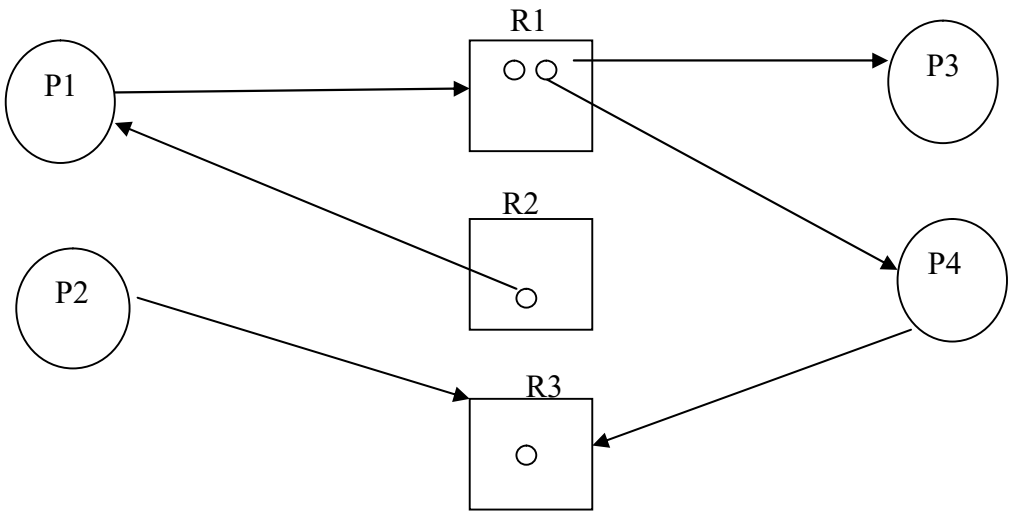
Fig 4: resource-allocation graph with a cycle but no deadlock

However there is no deadlock. Process P4 may release its instance of resource type R2. That resource can then be allocated to P3, breaking the cycle.

Problem 1- in a system, the following state of processes and resource are given

P1 ->R1, P2 ->R3, R2 ->P1, R1 ->P3, P4 ->R3, R1 ->P4

Solution-



Problem 2- A system contains one tape and one printer and two processes Pi and Pj that use three resources as follows:

Process Pi	Process Pj
Request Tape	Request Printer
Request Printer	Request Tape
Use Tape and Printer	Use Tape and Printer
Release Printer	Release Printer
Release Tape	Release Printer

Show that the set of process {Pi, Pj} is in deadlock state.

Solution- Resource requests by Pi and Pj take place in the following order:

(1) Process Pi requests the tape.
(2) Process Pj requests the printer.
(3) Process Pi requests the printer.
(4) Process Pj requests the tape.

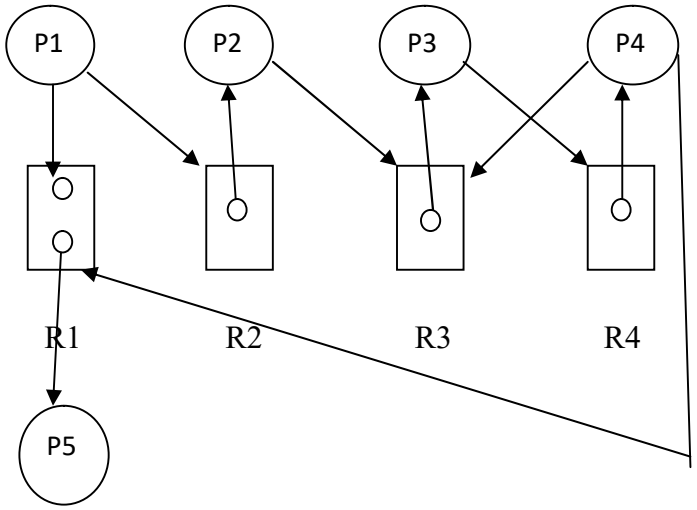
The first two (1) and (2) requests are granted immediately because a tape and a printer exist in the system. Now process Pi holds the tape and Pj holds the printer. When Pi asks for the printer, it is blocked until Pj releases the printer. Similarly Pj is blocked until Pi releases the tape. So the set of processes {Pi, Pj} is in deadlock state.

Problem 3- In a system, the following state of processes and resources are given:

R1 ->P1, P1 ->R2, P2 ->R3, R2 ->P2, R3 ->P3, P3 ->R4, P4 ->R3, R4 ->P4, P4 ->R1, R1 ->P5

Draw the resource-allocation graph (RAG) for the system and check for deadlock condition.

Solution-



In this scenario, processes P1, P2, P3 and P4 are holding one resource each and requesting for one more resource, which is held by the next process. From the RAG drawn above, it can be seen that there is a cycle, in the form of circular wait in the graph, thereby, causing a deadlock situation.

References:

(1) Abraham Silberschatz, Galvin & Gagne, Operating System Concepts, John Wiley & Sons, INC.

(2) Harvay M.Deital, Introduction to Operating System, Addition Wesley Publication Company.

(3) Vijay Shukla, Operating System, S.K. Kataria & Sons.

(4) Naresh Chauhan, Principles of Operating System, Oxford University Press.

Operating System

Prepared By:

Dr. Sanjeev Gangwar

Assistant Professor

Department of Computer Applications,

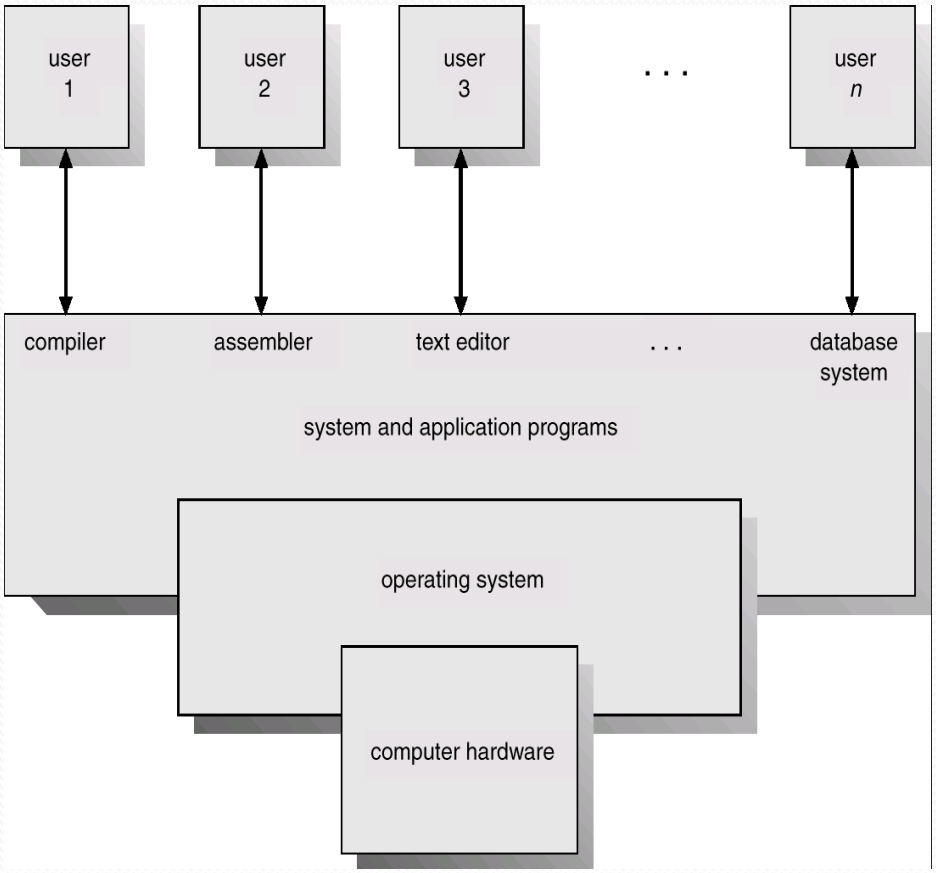
VBS Purvanchal University, Jaunpur

Introduction

- What is an operating system?
- Goals of an Operating System
- Computer System Components
- Operating System Definitions
- Operating System Functions
- Simple Batch Systems
- Multi-user Operating System
- Multi-processor Operating System
- Multiprogramming Operating System
- Time-Sharing System
- Real-Time System

What is an Operating System?

- An operating system runs on computer hardware and serves as a platform for other software to run on the computer system.
- An Operating System (OS) is an interface between computer user and computer hardware.



Goals of an Operating System

- The purpose of an operating system is to be provided an environment in which a user can execute programs.
- Its primary goals are to make the computer system convenience for the user.
- An operating system allows the computer hardware to be used in an efficient manner.
- The Operating System is also responsible for security and ensuring that unauthorized users do not access the system.

Computer System Components

A computer system can be divided roughly into four components

- **Hardware-** includes the physical parts of a computer such as CPU, Memory, I/O devices
- **Operating system-** is software that communicates with the hardware and allows other programs to run.
- **Applications programs-** is a software program that runs on your computer (Word processors, Database, Media players, Video games, Compilers).
- **Users-** A user is a person who utilizes a computer (People, Machines, other computers).

Operating System Definitions

- **Resource Allocator-** Operating system can be viewed as resource allocator where in resources are – CPU time, memory space, file storage space, I/O devices etc. Operating system must decide how to allocate these resources to specific programs and users so that it can operate the computer system efficiently.
- **Control Program**- Operating system is also a control program. A control program manages the execution of user programs to prevent errors and improper use of computer. It is concerned with the operation and control of I/O devices.

Operating System Functions

The various functions of operating system are as follows:

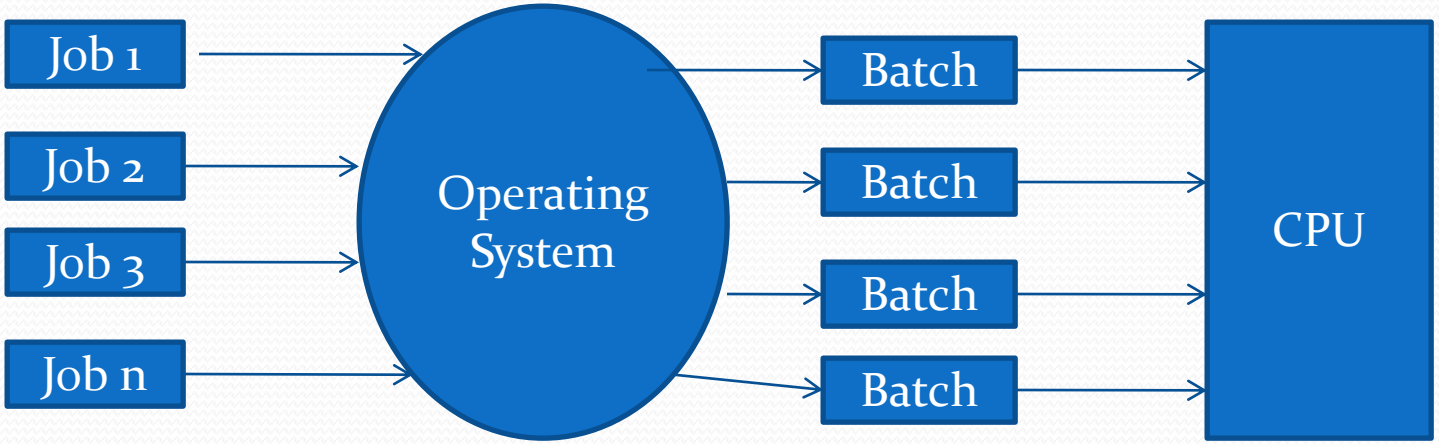
- **Process Management function** allocates the processor to execute a chosen process.
- **Memory Management function** finds free space in memory and allocate it to different processes.
- **File Management function** keeps track of all information about files that how they are opened or closed.
- **Device Management function** allocates a device to a process.

Simple Batch Systems

- Batch Operating Systems are the oldest types of operating system.
- The users of a batch operating system do not interact with the computer directly.
- Each user prepares his job on an off-line device like punch cards and submit it to the computer operator.
- Automatically transfers control from one job to another.
- To speed up processing, jobs with similar needs are batched together and run as a group.

Simple Batch Systems (Cont.)

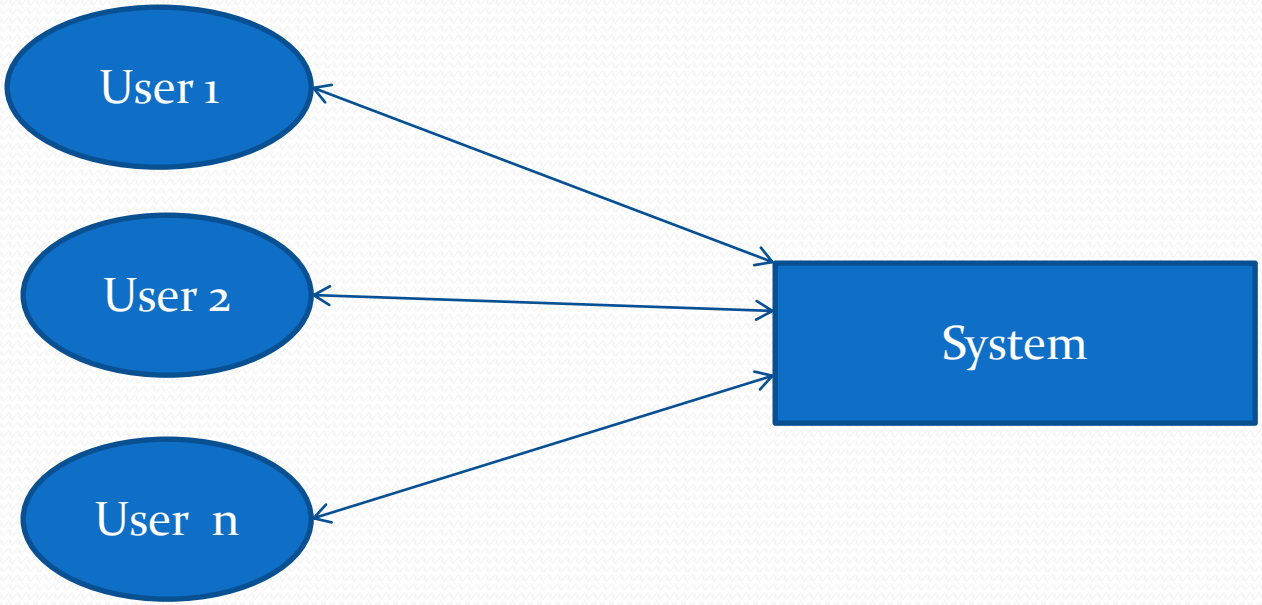
- The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.
- Common output devices were line printers, tape drives and card punches.



Multi-user Operating System

- Known as network operating systems.
- It allows for multiple users to use the same computer at the same time and/or different times.
- It is used as a single server and multiple number of clients.
- Remote access is provided via a network so that users can access the computer remotely using a terminal or other computer.
- Time sharing system and Internet servers as the multi user systems.
- Examples are Linux, Unix Windows etc.

Multi-user Operating System (Cont.)



Multi-user operating system

Multi-processor Operating System

- Multi-processor operating system refers to the use of two or more central processing units (CPU) within a single computer system.
- These multiple CPUs are in a close communication sharing the computer bus, memory and other peripheral devices. These systems are referred as tightly coupled systems or parallel systems.
- In a loosely coupled system, each processor has its own logical address space and its own memory. These systems are referred as distributed memory systems.

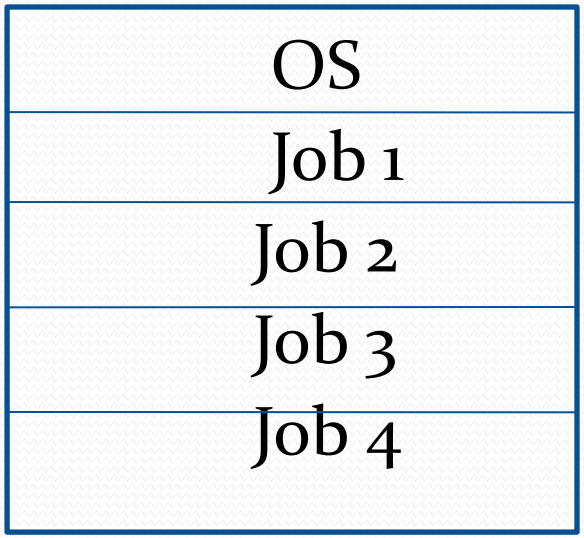
Multi-processor Operating System (Cont.)

Advantages:

- Economical
- Increased throughput
- Increased reliability

- Symmetric Multiprocessing- Each processor runs an identical copy of the operating system.
- Asymmetric Multiprocessing- Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.

Multiprogramming Operating System



- Multiprogramming means more than one process in main memory which are ready to execute
- The idea is to reduce the CPU idle time for as long as possible.

➢ Advantages:

- Efficient memory utilization
- Throughput increases
- CPU is never idle, so performance increases.

Time Sharing System

- A system that allows multiple users to interact with a computer at the same time.
- Time sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time sharing.
- Time sharing OS also allows many programs to run at the same time.

➢ Advantages:

- Quick response
- Reduces CPU idle time

Time Sharing System (Cont.)

- The main difference between Multiprogramming operating system and Time-Sharing system is that in case of Multiprogrammed OS, objective is to maximize processor use, whereas in Time sharing system objective is to minimize response time.

Examples: Multics, Unix, etc.

Real-Time System

- A real-time requires that results be produced within a specified deadline period.
- Most real-time system are embedded.
- System that interact predictably with events in the outside world.

➢ Examples:

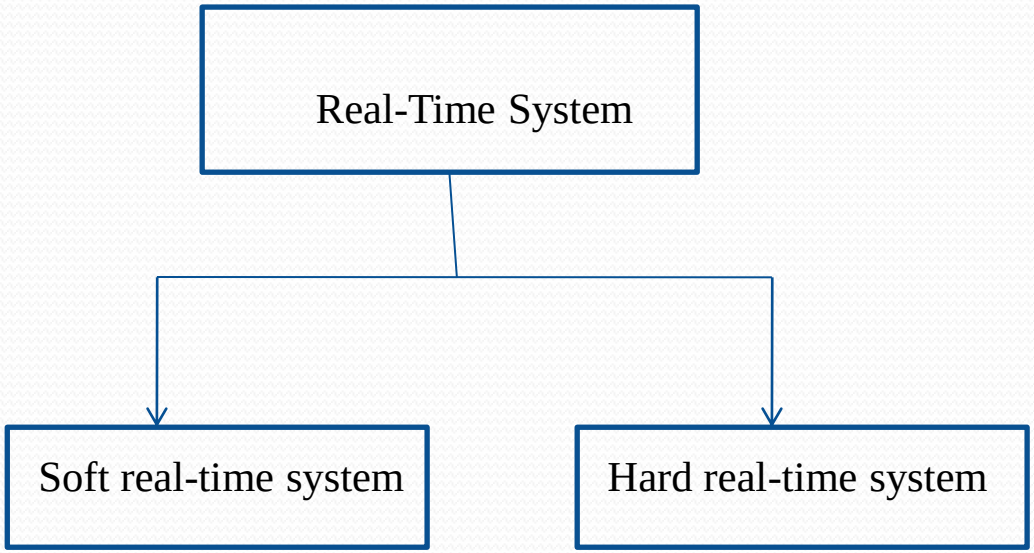
➢ Flight control system

➢ Satellite guidance system

➢ Patient monitoring system

➢ Airbag system

Real-Time System (Cont.)



A hard real-time system guarantees that real-time tasks be completed within their required deadlines.

Example: Air traffic control, medical systems.

A soft real-time system provides priority of real-time tasks over non real-time tasks.

Example: Multimedia streaming, Computer game

References

- Abraham Silberschatz, Galvin & Gagne, Operating System Concepts, John Wiley & Sons, INC.
- Harvay M.Deital, Introduction to Operating System, Addition Wesley Publication Company.
- Andrew S.Tanenbaum, Operating System Design and Implementation, PHI

Thank You