Unit-2 Process Management

❖ Overview of the Process & threads:

Process:-

In computing, a process is an instance of a computer program that is being executed.
It contains the program code and its current activity. Whereas a program is a set of Instructions
Process is a part of code which under execution or in detail u can say that in main memory whereas a program is a complete task or work to be done
A process is a program in execution. it also includes the current activity, as represented by the value of program counter and the contents of the program counter.
A process includes a process stack which contains the temporary data and a data section which contains the global variables. while a program is a sequence of instructions written in a manner to obtain the desired output.

Thread:-

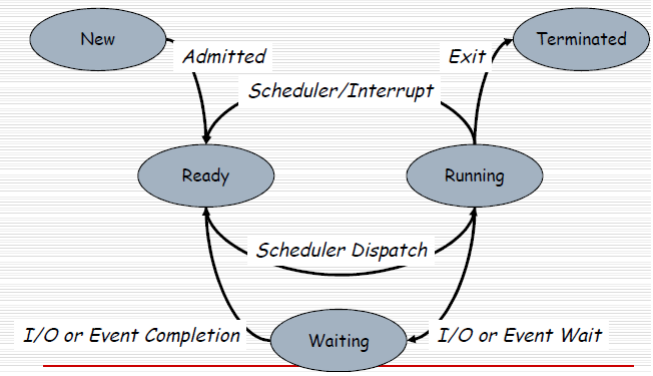
Thread is part of program that can be executed simultaneously independently from the other part of a program. The Operating System has a scheduler for each thread (process) that is currently running. It divides up time slides for each of them, which are executed in the order that Operating System seems fit. It simply runs each one in some arbitrary order for a set number of milliseconds and then switches between them constantly.

User Threads
They are supported above the kernel and are implemented by a thread library at the user level.
The library provides support for thread creation, scheduling and management with no support from kernel. Kernel is unaware of user-level threads; all thread creation and scheduling is done in the user space so they are fast to create and manage.
User-thread libraries include POSIX P-threads, Mach C-Thread and solaria UI-threads.

Kernel Threads
Kernel threads are supported by Operating System: the kernel performs thread creation, scheduling and management in kernel space. Kernel threads are slower to create and manage.
Windows NT, Windows 2000, Solaris 2, BeOS, and Tru64 UNIX support kernel threads.

❖ Process Life Cycle/ Process States

In computing, a process is an instance of a computer program that is being executed.
It contains the program code and its current activity.
A process includes a process stack which contains the temporary data and a data section which contains the global variables.
A process moves through various states during its life:



1. New

A program which is going to be picked up by the OS into the main memory is called a new process.

2. Ready

Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned. The OS picks the new processes from the secondary memory and put all of them in the main memory.

The processes which are ready for the execution and reside in the main memory are called ready state processes. There can be many processes present in the ready state.

3. Running

One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm. Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one. If we have n processors in the system then we can have n processes running simultaneously.

4. Block or wait

From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process.

When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes.

5. Completion or termination

When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.

❖ Process Control Block

Each process is represented by a *process control block* (PCB) which contains information such as: state of the process (ready, running, blocked) ,register values ,priorities and scheduling information memory management information, accounting information such as open files allocated resources

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them. This is also called context of the process. Attributes which are stored in the PCB are described below.

1. Process ID

When a process is created, a unique id is assigned to the process which is used for unique identification of the process in the system.

2. Program counter

A program counter stores the address of the last instruction of the process on which the process was suspended. The CPU uses this address when the execution of this process is resumed.

3. Process State

The Process, from its creation to the completion, goes through various states which are new, ready, running and waiting. We will discuss about them later in detail.

4. Priority

Every process has its own priority. The process with the highest priority among the processes gets the CPU first. This is also stored on the process control block.

5. General Purpose Registers

Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

6. List of open files

During the Execution, Every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.

7. List of open devices

OS also maintain the list of all open devices which are used during the execution of the process.

❖ Scheduling Criteria:

CPU utilization -keep the CPU as busy as possible.

Throughput -Number of processes that complete their execution per time unit.

Arrival Time-The time at which the process enters into the ready queue is called the arrival time.

Burst Time- The total amount of time required by the CPU to execute the whole process is called the Burst Time. This does not include the waiting time. It is confusing to calculate the execution time for a process even before executing it hence the scheduling problems based on the burst time cannot be implemented in reality.

Completion Time-The Time at which the process enters into the completion state or the time at which the process completes its execution, is called completion time.

Turnaround time-The total amount of time spent by the process from its arrival to its completion, is called Turnaround time.

Waiting Time-The Total amount of time for which the process waits for the CPU to be assigned is called waiting time.

Response Time-The difference between the arrival time and the time at which the process first gets the CPU is called Response Time.

❖ Scheduling Algorithms

1. First Come First Serve

First come first serve (FCFS) scheduling algorithm simply schedules the jobs according to their arrival time. The job which comes first in the ready queue will get the CPU first. The lesser the arrival time of the job, the sooner will the job get the CPU. FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs.

Advantages of FCFS

- Simple
- Easy
- First come, First serv

Disadvantages of FCFS

1. The scheduling method is non preemptive, the process will run to the completion.

2. Due to the non-preemptive nature of the algorithm, the problem of starvation may occur.
3. Although it is easy to implement, but it is poor in performance since the average waiting time is higher as compare to other scheduling algorithms.

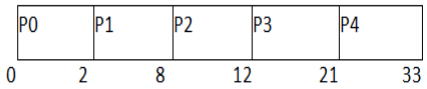
Example

Let's take an example of The FCFS scheduling algorithm. In the Following schedule, there are 5 processes with process ID **P0, P1, P2, P3 and P4**. P0 arrives at time 0, P1 at time 1, P2 at time 2, P3 arrives at time 3 and Process P4 arrives at time 4 in the ready queue. The processes and their respective Arrival and Burst time are given in the following table.

The Turnaround time and the waiting time are calculated by using the following formula.

1. Turn Around Time = Completion Time - Arrival Time
2. Waiting Time = Turnaround time - Burst Time

The average waiting Time is determined by summing the respective waiting time of all the processes and divided the sum by the total number of processes.



Process ID	Arrival Time(T0)	Burst Time(T1)	Completion Time(CT)	Turn Around Time(TAT=CT-T0)	Waiting Time(WT=TAT-T1)
P0	0	2	2	2	0
P1	1	6	8	7	1
P2	2	4	12	10	6
P3	3	9	21	18	9
P4	6	12	33	29	17

Average Waiting Time=31/5 =6.2 ms

Average Turn around Time=66/5 =13.2 ms

2. Shortest Job First

Till now, we were scheduling the processes according to their arrival time (in FCFS scheduling). However, SJF scheduling algorithm, schedules the processes according to their burst time.

In SJF scheduling, the process with the lowest burst time, among the list of available processes in the ready queue, is going to be scheduled next.

However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.

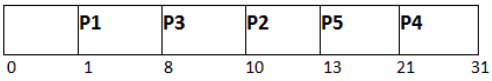
Advantages of SJF

1. Maximum throughput
2. Minimum average waiting and turnaround time

Disadvantages of SJF

1. May suffer with the problem of starvation
2. It is not implementable because the exact Burst time for a process can't be known in advance.

There are different techniques available by which, the CPU burst time of the process can be determined. We will discuss them later in detail.



Example

In the following example, there are five jobs named as P1, P2, P3, P4 and P5. Their arrival time and burst time are given in the table below.

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7
3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

Since, No Process arrives at time 0 hence; there will be an empty slot in the **Gantt chart** from time 0 to 1 (the time at which the first process arrives).

According to the algorithm, the OS schedules the process which is having the lowest burst time among the available processes in the ready queue.

Average Waiting Time=27/5 =5.4 ms

Average Turn around Time=57/5 =11.4 ms

3. Round Robin

Round Robin scheduling algorithm is one of the most popular scheduling algorithm which can actually be implemented in most of the operating systems. This is the **preemptive version** of first come first serve scheduling.

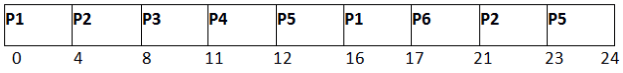
Advantages

1. It can be actually implementable in the system because it is not depending on the burst time.
2. It doesn't suffer from the problem of starvation or convoy effect.
3. All the jobs get a fare allocation of CPU.

Disadvantages

1. The higher the time quantum, the higher the response time in the system.
2. The lower the time quantum, the higher the context switching overhead in the system.
3. Deciding a perfect time quantum is really a very difficult task in the system.

Here time quantum is 4ms



The completion time, Turnaround time and waiting time will be calculated as shown in the table below.

As, we know,

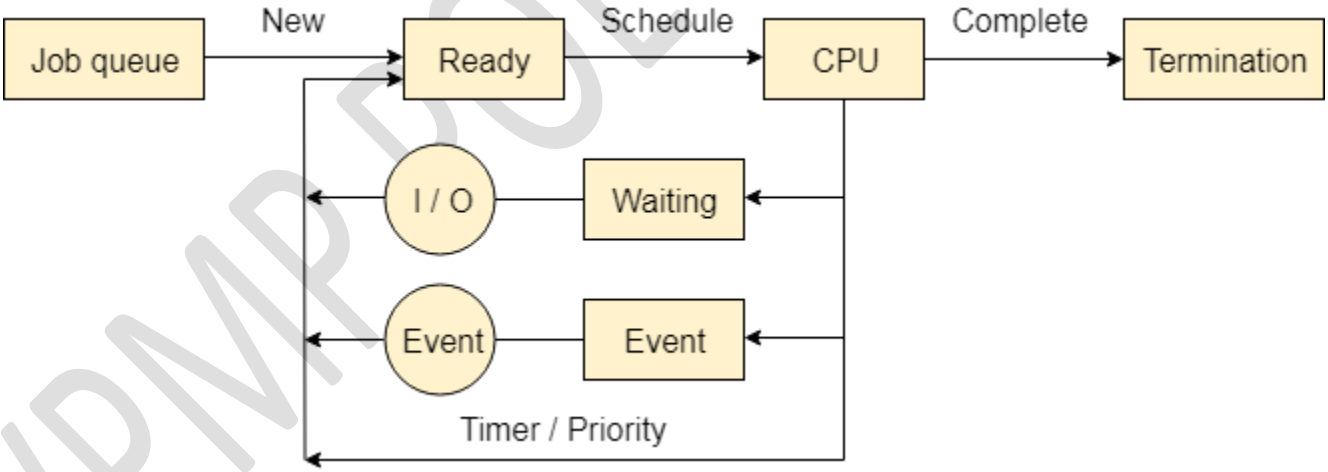
Turn Around Time = Completion Time - Arrival Time
Waiting Time = Turn Around Time - Burst Time

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	5	17	17	12
2	1	6	23	22	16
3	2	3	11	9	6
4	3	1	12	9	8
5	4	5	24	20	15
6	6	4	21	15	11

Average Waiting Time = (12+16+6+8+15+11)/6 = 76/6 =12.66

❖ Scheduling Queues

The Operating system manages various types of queues for each of the process states. The PCB related to the process is also stored in the queue of the same state. If the Process is moved from one state to another state then its PCB is also unlinked from the corresponding queue and added to the other state queue in which the transition is made.



There are the following queues maintained by the Operating system.

1. Job Queue

In starting, all the processes get stored in the job queue. It is maintained in the secondary memory. The long term scheduler (Job scheduler) picks some of the jobs and put them in the primary memory.

2. Ready Queue

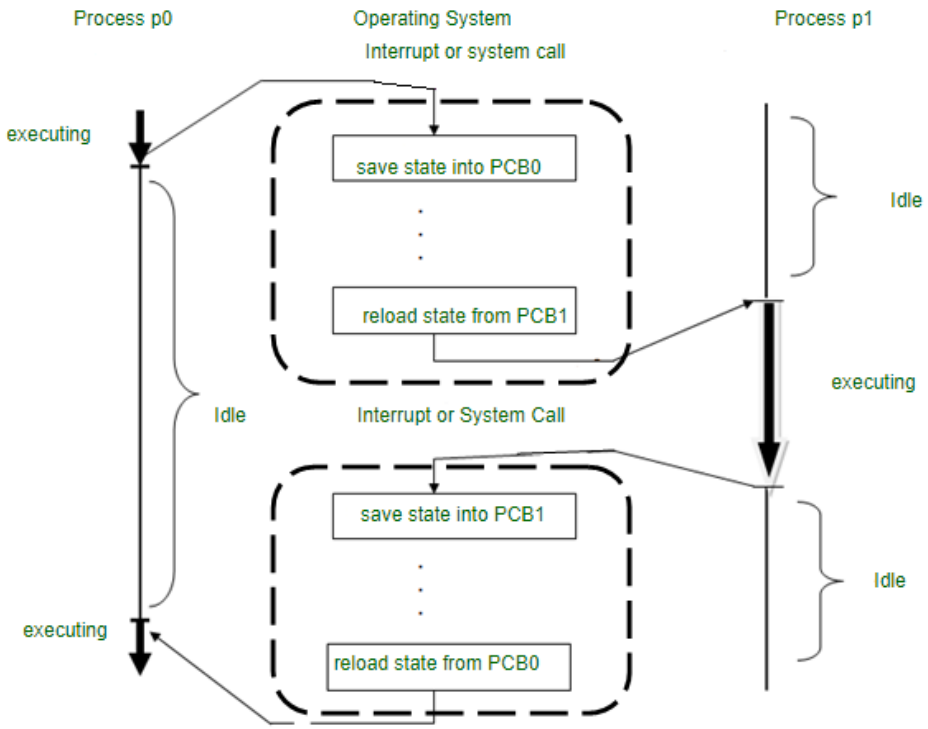
Ready queue is maintained in primary memory. The short term scheduler picks the job from the ready queue and dispatch to the CPU for the execution

3. Waiting Queue

When the process needs some IO operation in order to complete its execution, OS changes the state of the process from running to waiting. The context (PCB) associated with the process gets stored on the waiting queue which will be used by the Processor when the process finishes the IO.

❖ Context Switch

An operating system uses this technique to switch a process between states to execute its functions through CPUs. It is a process of saving the context (state) of the old process(suspend) and loading it into the new process(resume). It occurs whenever the CPU switches between one process and another. Basically, the state of CPU's registers and program counter at any time represent a context. Here, the saved state of the currently executing process means to copy all live registers to PCB (Process Control Block). Moreover, after that, restore the state of the process to run or execute next, which means copying live registers' values from PCB to registers.



❖

❖ Process Synchronization

Process synchronization is a mechanism to ensure systematic sharing of resources among concurrent (parallel) process.

Two issues related to Process synchronization:

- Two or more process should not come into each other's way
- Proper sequencing should be maintained during execution.

In IPC one problem arises called Race condition.

Race condition / Racing problem:

It is situation where two or more process are reading / writing some shared data and final result depends on relative order of their execution ,is called race condition.

A=1000,At the end of two process A should be 1100.but if P0 and P1 permitted to execute in any arbitrary fashion then output will be not same**.**

Po	P1
Read (A);	Read(A)
A: A-100	A :A +200;
Write (A)	Write (A)

Possibility :1

Here, Answer will be 1200

Po	P1
Read (A);	
	Read(A)
A: A-100	
Write (A)	
	A :A +200;
	Write (A)

Possibility : 2

Here, Answer will be 900

Po	P1
Read (A);	
	Read(A)
	A :A +200;
	Write (A)
A : A-100	
Write (A)	

Here two processes are reading and writing common variable 'A' .The final value depends on relative execution order of P0 and P1, such situation is called race condition.

This implies that concurrent process are racing with each other to access a shared resource in arbitrary order and procedure wrong final results ,so race condition must be avoided.

❖ Critical Section

A critical section is a code segment of a process where the shared resources are accessed.

It is also known as critical region.

It is a part of a program or code segment of a process where a process may change common variables, updating a table, writing a file or accessing some physical device.

Only one process should be allowed to execute in its critical section.

OR

Only one process should be allowed to access a shared resource at a time.

It is the responsibility of an OS to ensure that no more than one process is present in its critical section simultaneously.

Requirement of critical section solution:

An ideal critical section solution meet the following three condition:

Mutual exclusion:

At any time, at most one of the co-operating process should be executing in its critical section

Progress:

When no any process is running in its critical section and other process wants to enter its critical section, it should be allowed immediately.

If more than one process is waiting to enter their critical sections, one of them should be selected to enter.

This termed as requirement of progress, which must be met under all possible conditions.

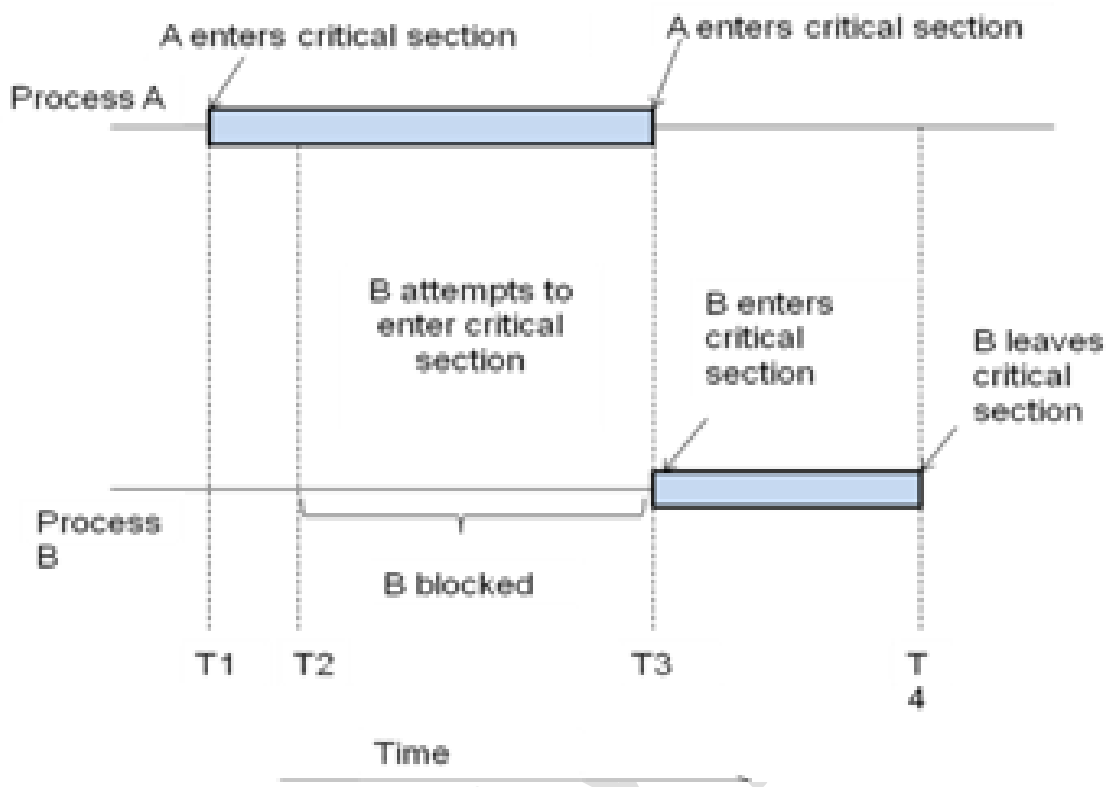
Bounded waiting:

When more than one process is waiting to enter their critical sections, one of them is selected to enter in critical section.

Other processes simply wait. But such type of waiting must be limited. There should not be starvation.

It should not be such that one process waits forever and other processes get entry in critical section more than once.

There should be some upper bound on the no, of times that other processes allowed to enter in their critical sections before a requesting process is granted to enter in its critical section.



❖ Mutual Exclusion

Mutual exclusion means only one process at a time can use a resource.

If some other process requests that resource, the requesting process must be waiting until the resource has been released.

We can deny this condition by simple protocol i.e., "convert the all non-sharable resources to sharable resources". So this condition is not satisfied by the deadlock, then we can prevent the deadlock.

During concurrent execution of processes, processes need to enter the critical section (or the section of the program shared across processes) at times for execution. It might so happen that because of the execution of multiple processes at once, the values stored in the critical section become inconsistent.

In other words, the values depend on the sequence of execution of instructions – also known as a race condition. The primary task of process synchronization is to get rid of race conditions while executing the critical section.

This is primarily achieved through mutual exclusion.

Mutual exclusion is a property of process synchronization which states that "no two processes can exist in the critical section at any given point of time". The term was first coined by Dijkstra. Any process synchronization technique being used must satisfy the property of mutual exclusion, without which it would not be possible to get rid of a race condition.

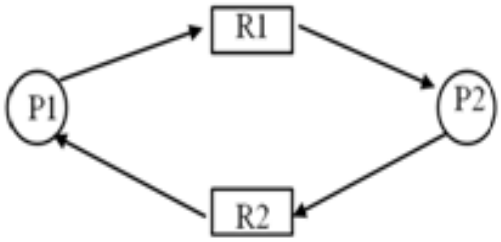
❖ **Deadlock** A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.

A process request the resources, the resources are not available at that time, so the process enter into the waiting state.

The requesting resources are held by another waiting process, both are in waiting state, this situation is said to be "Deadlock".

For example :

P1 and P2 are two processes, R1 and R2 are two resources. P1 request the resource R1, R1 held by process P2, P2 request the resource R2, R2 is held by P1, then both are entered into the waiting state. There is no work progress for process P1 and P2, it is also a Deadlock. Deadlock can be represented by Resource Allocation Graph.



❖ Conditions for Deadlock

Deadlock situation can arise if the following four conditions hold simultaneously (all at one time) in system.

A Deadlocked system must satisfy the following 4 conditions. These are:

MUTUAL EXCLUSION: Each resource can be assigned to exacty one process if any process requests resources which are not free then that process will wait until resource becomes free.

HOLD AND WAIT: A process must be holding at least one resource and waiting to acquire (get) additional resources that are currently being held by other process.

NO PREEMPTION: A process must be holding at least one resource and waiting to acquire (get) additional resources that are currently being held by other process. No preemption means resources are not released in the middle of the work, they released only after the process has completed its task.

CIRCULAR WAIT: There must be circular chain or two or more process . A set { P1, P2,..…Pn } of waiting process must exist such that P1 is waiting for resource R1, P2 waits for resource R2, P n-1 waits for Pn and Pn waits for P0.it said to be circular wait.

Process { P0 , P1 ,P2 } and Resource { R1, R2 ,R3 ,R4}

❖ Resource allocation graph

Deadlocks can be described accurately using 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 two different types of nodes :

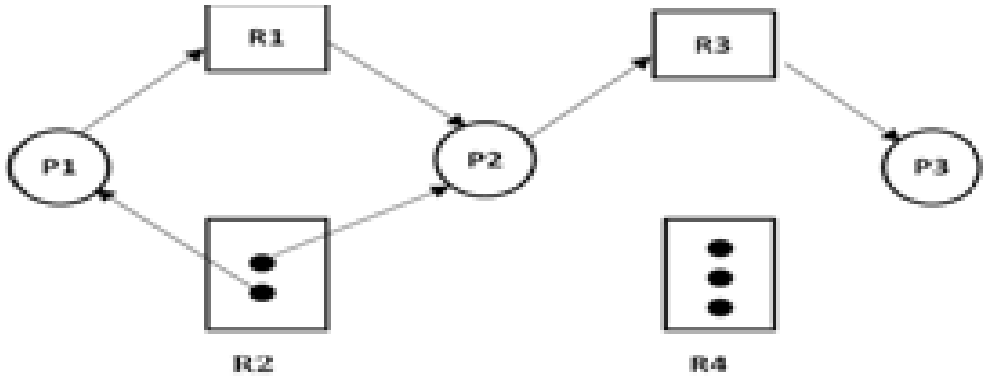
P = {P1, P2, P3…….Pn}, the set of all active processes in the system ;

- Process Pi represented using circle.

R = {R1, R2,……Rm}, the set of all resource types in the system.

- Resource Rj represented using rectangle.
- If resource Rj may have more than instance as dot within circle.
- The set of edges E is in two different types :
- A directed edge from process Pi to resource type Rj is denoted by Pi → Rj, it is called request edge. It means that process Pi requested an instance of resource type Rj and is waiting for that resource.
- A directed edge from resource type Rj to process Pi is denoted by Rj → Pi, It is called assignment edge. It means that an instance of resource type Rj has been allocated to process Pi.

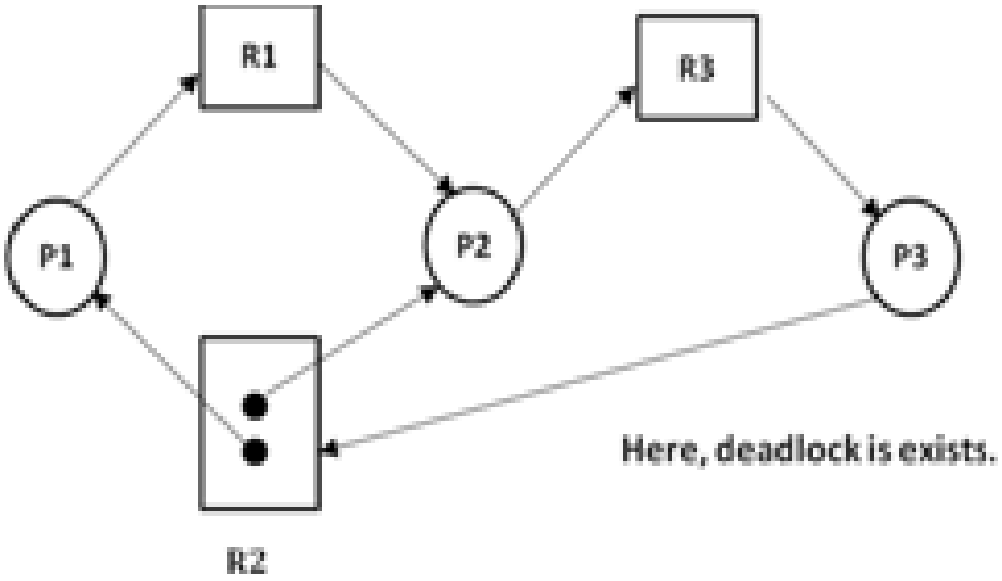
Resource Allocation Modeling using Graphs



Resource allocation graph

- The sets of
- process P = { P1, P2, P3 },
- resource R = {R1, R2, R3, R4 },
- Edges E = { P1→R1, P2→R3, R1 → P2, R2→P2, R2→R1, R3→ P3}
- Resource instances – 1 instance of resource type R1, 2 instances of resource type R2, 1 instance of resource type R3 and 3 instances of resource type R4.
- Here ,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 resource type R1 and R2, and is waiting for an instance of resource type R3.

- Process P3 is holding an instance of resource type R3.



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