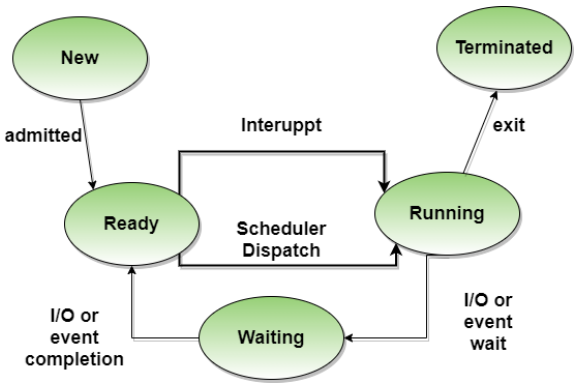
Q.1 Write a short note on process.

Ans. A process is a program in execution which then forms the basis of all computation. The process is not as same as program code but a lot more than it. A process is an 'active' entity as opposed to the program which is considered to be a 'passive' entity.

Processes in the operating system can be in any of the following states:

- NEW- The process is being created.
- READY- The process is waiting to be assigned to a processor.
- 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).
- TERMINATED- The process has finished execution.

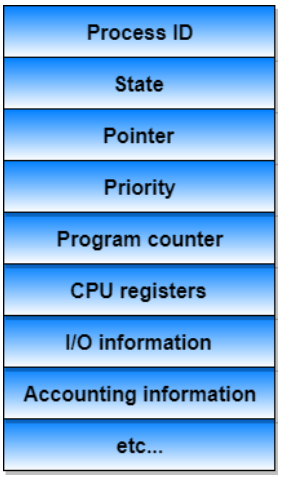


Process Control Block

There is a Process Control Block for each process, enclosing all the information about the process. It is also known as the task control block. It is a data structure, which contains the following:

- **Process State**: It can be running, waiting, etc.
- **Process ID** and the **parent process ID**.
- CPU registers and Program Counter. **Program Counter** holds the address of the next instruction to be executed for that process.

- **CPU Scheduling** information: Such as priority information and pointers to scheduling queues.
- **Memory Management information**: For example, page tables or segment tables.
- **Accounting information**: The User and kernel CPU time consumed, account numbers, limits, etc.
- **I/O Status information**: Devices allocated, open file tables, etc.

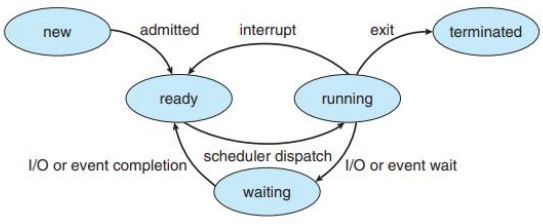


Q.2 Explain various states of a process.

Ans. There are different states of a process as it executes. The state of a process is determined based on the current activity of the process.

1. **New:** The process is being created.
2. **Running:** The program instructions are being executed.
3. **Waiting:** The process is waiting for some event to occur (such as an I/O completion or reception of a signal.
4. **Ready:** The process is waiting in the ready queue to be assigned to a processor.
5. **Terminated:** The process has finished execution.

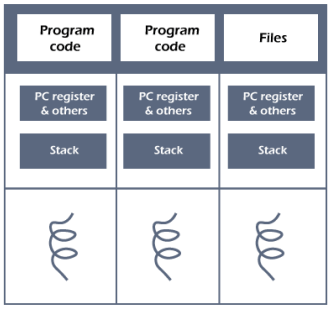
It is important to realize that only one process can be in ***running*** state on any processor at any instant. Many processes may be ***ready*** and ***waiting***, however.



Q.3 Write a short note on thread.

Ans.

- A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control.
- There is a way of thread execution inside the process of any operating system.
- Apart from this, there can be more than one thread inside a process.
- Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks.
- Thread is often referred to as a lightweight process.
- The process can be split down into so many threads.
- **For example**, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.



Three threads of same process

Need of Thread:

- It takes far less time to create a new thread in an existing process than to create a new process.
- Threads can share the common data, they do not need to use Inter- Process communication.
- Context switching is faster when working with threads.
- It takes less time to terminate a thread than a process.

Q.4 Differentiate process and thread.

Ans.

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means a segment of a process.
2.	The process takes more time to terminate.	The thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
6.	Multiprogramming holds the concepts of multi-process.	We don't need multi programs in action for multiple threads because a single process consists of multiple threads.
7.	The process is isolated.	Threads share memory.
8.	The process is called the heavyweight process.	A Thread is lightweight as each thread in a process shares code, data, and resources.
9.	Process switching uses an interface in an operating system.	Thread switching does not require calling an operating system and causes an interrupt to the kernel.
10.	If one process is blocked then it will not affect the execution of other processes	If a user-level thread is blocked, then all other user-level threads are blocked.
11.	The process has its own Process Control Block, Stack, and Address Space.	Thread has Parents' PCB, its own Thread Control Block, and Stack and common Address space.
12.	Changes to the parent process do not affect child processes.	Since all threads of the same process share address space and other resources so any changes to the main thread may affect the behavior of the other threads of the process.
13.	A system call is involved in it.	No system call is involved, it is created using APIs.
14.	The process does not share data with each other.	Threads share data with each other.

Q.5 Write a short note on CPU scheduling algorithm.

Ans. CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended. The main function of the CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line.

- Types: Preemptive and non-preemptive
- Assumption initially: No process has I/O requirement
- Some terminologies:
  - Arrival Time(AT): Process arrival time(admitted by LTS)
  - Burst/Service time(BT): Time required to run on the CPU
  - Completion time (CT): Time when process completes execution
  - Turn around time (TAT): Time from arrival until completion (TAT=CT-AT)

- Waiting time (WT): Waiting for CPU in ready queue (WT=TAT-BT)
- Response time: Time from arrival to the first response on CPU
- Scheduling length: max(CT)-min(AT)
- Throughput: No of processes executed per unit time.

Q.6 Explain FCFS with suitable example.

Ans. FCFS considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

Characteristics of FCFS:

- FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
- Tasks are always executed on a First-come, First-serve concept.
- FCFS is easy to implement and use.
- This algorithm is not much efficient in performance, and the wait time is quite high.

Advantages of FCFS:

- Easy to implement.
- First come, first serve method.

Disadvantages of FCFS:

- FCFS suffers from **Convoy effect**.
- The average waiting time is much higher than the other algorithms.
- FCFS is very simple and easy to implement and hence not much efficient.

Example-

Process	AT	BT	CT	TAT	WT	RT
P1	5	4	9	4	0	0
P2	8	2	21	13	11	11
P3	6	3	12	6	3	3
P4	3	1	5	2	1	1
P5	2	2	4	2	0	0
P6	7	7	19	12	5	5

Gantt Chart

	P5	P4	P1	P3	P6	p2	
0	2	4	5	9	12	19	21

Avg WT: 20/6=3.3

Avg TAT: 39/6=6.5

Avg RT: 20/6=3.3

L: max(CT)-min(AT)=21-2=19

Throughput = 6/19

Q.7 Explain SJF with suitable example.

Ans. Shortest job first (SJF) is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.

Characteristics of SJF:

- Shortest Job first has the advantage of having a minimum average waiting time among all operating system scheduling algorithms.
- It is associated with each task as a unit of time to complete.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

Advantages of Shortest Job first:

- As SJF reduces the average waiting time thus, it is better than the first come first serve scheduling algorithm.
- SJF is generally used for long term scheduling

Disadvantages of SJF:

- One of the demerit SJF has is starvation.
- Many times it becomes complicated to predict the length of the upcoming CPU request.

Example –

Process ID	Burst Time	Arrival Time	Completion time	Waiting Time	Turnaround Time
P0	8	5	21	8	16
P1	5	0	5	0	5
P2	9	4	16	3	12
P3	2	1	7	4	6

The waiting time and turnaround time are calculated with the help of the following formula.

Waiting Time = Turnaround time – Burst Time

Turnaround Time = Completion time – Arrival time

Process waiting time:

P0= 16-8=8

P1= 5-5=0

P2=12-9=3

P3=6-2=4

Average waiting time= 8+0+3+4/4

=15/4

=3.75

Process turnaround time:

P0=21-5=16

P1=5-0=5

P2=16-4=12

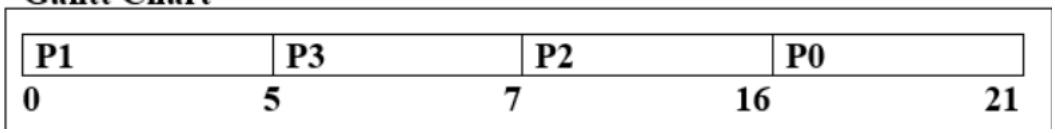
P3=7-1=6

Average turnaround time=16+5+12+6/4

=39/4

=9.75

Gantt Chart



Q.8 Explain Round Robin with suitable example.

Ans. Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

Characteristics of Round robin:

- It's simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.
- One of the most widely used methods in CPU scheduling as a core.
- It is considered preemptive as the processes are given to the CPU for a very limited time.

Advantage of Round-robin Scheduling

Here, are pros/benefits of Round-robin scheduling method:

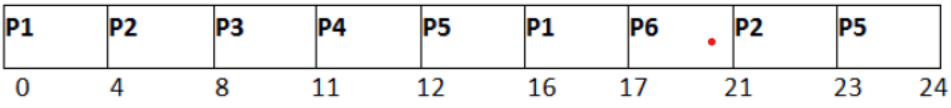
- It doesn't face the issues of starvation or convoy effect.
- All the jobs get a fair allocation of CPU.
- It deals with all process without any priority
- If you know the total number of processes on the run queue, then you can also assume the worst-case response time for the same process.
- This scheduling method does not depend upon burst time. That's why it is easily implementable on the system.
- Once a process is executed for a specific set of the period, the process is preempted, and another process executes for that given time period.
- Allows OS to use the Context switching method to save states of preempted processes.
- It gives the best performance in terms of average response time.

Disadvantages of Round-robin Scheduling

Here, are drawbacks/cons of using Round-robin scheduling:

- If slicing time of OS is low, the processor output will be reduced.
- This method spends more time on context switching
- Its performance heavily depends on time quantum.
- Priorities cannot be set for the processes.
- Round-robin scheduling doesn't give special priority to more important tasks.
- Decreases comprehension
- Lower time quantum results in higher the context switching overhead in the system.
- Finding a correct time quantum is a quite difficult task in this system.

Example –



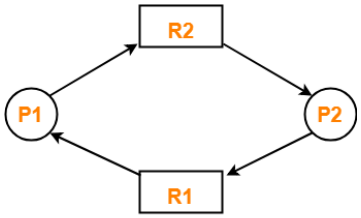
Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	5	17	17	12
2	1	6	23	22	16
3	2	3	11	9	6
4	3	1	12	9	8
5	4	5	24	20	15
6	6	4	21	15	11

Avg Waiting Time = (12+16+6+8+15+11)/6 = 76/6 units

Q.9 What is deadlock? Explain necessary conditions for deadlock.

Ans. Deadlock is a situation where-

- The execution of two or more processes is blocked because each process holds some resource and waits for another resource held by some other process.



Example of a deadlock

Here

- Process P1 holds resource R1 and waits for resource R2 which is held by process P2.
- Process P2 holds resource R2 and waits for resource R1 which is held by process P1.
- None of the two processes can complete and release their resource.
- Thus, both the processes keep waiting infinitely.

Conditions For Deadlock-

There are following 4 necessary conditions for the occurrence of deadlock-

1. **Mutual Exclusion**
2. **Hold and Wait**

3. **No preemption**
4. **Circular wait**

1. Mutual Exclusion-

By this condition,

- There must exist at least one resource in the system which can be used by only one process at a time.
- If there exists no such resource, then deadlock will never occur.
- Printer is an example of a resource that can be used by only one process at a time.

2. Hold and Wait-

By this condition,

- There must exist a process which holds some resource and waits for another resource held by some other process.

3. No Preemption-

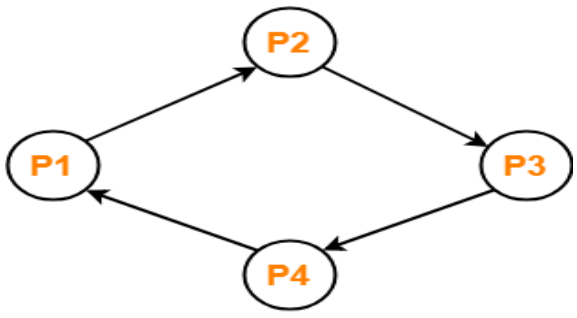
By this condition,

- Once the resource has been allocated to the process, it cannot be preempted.
- It means resource cannot be snatched forcefully from one process and given to the other process.
- The process must release the resource voluntarily by itself.

4. Circular Wait-

By this condition,

- All the processes must wait for the resource in a cyclic manner where the last process waits for the resource held by the first process.



Circular Wait

Here,

- Process P1 waits for a resource held by process P2.
- Process P2 waits for a resource held by process P3.
- Process P3 waits for a resource held by process P4.
- Process P4 waits for a resource held by process P1.

Q.10 Explain deadlock prevention.

Ans. As discussed in the previous, deadlock has following characteristics.

1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular wait

Deadlock Prevention
We can prevent Deadlock by eliminating any of the above four conditions.

Eliminate Mutual Exclusion
It is not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.

Eliminate Hold and wait

1. Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization. For example, if a process requires printer at a later time and we have allocated printer before the start of its execution printer will remain blocked till it has completed its execution.

2. The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.

Eliminate No Preemption
Preempt resources from the process when resources required by other high priority processes.

Eliminate Circular Wait
Each resource will be assigned with a numerical number. A process can request the resources increasing/decreasing. Order of numbering.

For Example, if P1 process is allocated R5 resources, now next time if P1 ask for R4, R3 lesser than R5 such request will not be granted, only request for resources more than R5 will be granted.

Q.11 Explain Bankers algorithm with suitable example.

Ans. It is a banker algorithm used to **avoid deadlock** and **allocate resources** safely to each process in the computer system. The '**S-State'** examines all possible tests or activities before deciding whether the allocation should be allowed to each process. It also helps the operating system to successfully share the resources between all the processes. The banker's algorithm is named because it checks whether a person should be sanctioned a loan amount or not to help the bank system safely simulate allocation resources.

Following are the essential characteristics of the Banker's algorithm:

1. It contains various resources that meet the requirements of each process.
2. Each process should provide information to the operating system for upcoming resource requests, the number of resources, and how long the resources will be held.
3. It helps the operating system manage and control process requests for each type of resource in the computer system.
4. The algorithm has a Max resource attribute that represents indicates each process can hold the maximum number of resources in a system.

Disadvantages

1. It requires a fixed number of processes, and no additional processes can be started in the system while executing the process.
2. The algorithm does no longer allows the processes to exchange its maximum needs while processing its tasks.
3. Each process has to know and state their maximum resource requirement in advance for the system.
4. The number of resource requests can be granted in a finite time, but the time limit for allocating the resources is one year.

When working with a banker's algorithm, it requests to know about three things:

1. How much each process can request for each resource in the system. It is denoted by the [**MAX**] request.
2. How much each process is currently holding each resource in a system. It is denoted by the [**ALLOCATED**] resource.
3. It represents the number of each resource currently available in the system. It is denoted by the [**AVAILABLE**] resource.

Following are the important data structures terms applied in the banker's algorithm as follows:

Suppose n is the number of processes, and m is the number of each type of resource used in a computer system.

1. **Available**: It is an array of length 'm' that defines each type of resource available in the system. When Available[j] = K, means that 'K' instances of Resources type R[j] are available in the system.
2. **Max:** It is a [n x m] matrix that indicates each process P[i] can store the maximum number of resources R[j] (each type) in a system.
3. **Allocation:** It is a matrix of m x n orders that indicates the type of resources currently allocated to each process in the system. When Allocation [i, j] = K, it means that process P[i] is currently allocated K instances of Resources type R[j] in the system.
4. **Need:** It is an M x N matrix sequence representing the number of remaining resources for each process. When the Need[i] [j] = k, then process P[i] may require K more instances of resources type Rj to complete the assigned work. Nedd[i][j] = Max[i][j] - Allocation[i][j].
5. **Finish**: It is the vector of the order **m**. It includes a Boolean value (true/false) indicating whether the process has been allocated to the requested resources, and all resources have been released after finishing its task.
6. The Banker's Algorithm is the combination of the safety algorithm and the resource request algorithm to control the processes and avoid deadlock in a system:

SAFETY ALGORITHM

7. It is a safety algorithm used to check whether or not a system is in a safe state or follows the safe sequence in a banker's algorithm:
8. 1. There are two vectors **Wok** and **Finish** of length m and n in a safety algorithm.
9. Initialize:Work=Available
   Finish[i] = false; for I = 0, 1, 2, 3, 4… n - 1.
10. 2. Check the availability status for each type of resources [i], such as:
11. Need[i]<=Work
    Finish[i]==false
    If the i does not exist, go to step 4.
12. 3. Work = Work +Allocation(i) // to get new resource allocation
13. Finish[i] = true
14. Go to step 2 to check the status of resource availability for the next process.
15. 4. If Finish[i] == true; it means that the system is safe for all processes.

Resource-Request Algorithm for Process P_i

Request = request vector for process P_i. If $Request_i[j] = k$ then process P_i wants k instances of resource type R_j

1. If $Request_i \leq Need_i$ go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim
2. If $Request_i \leq Available$, go to step 3. Otherwise P_i must wait, since resources are not available
3. Pretend to allocate requested resources to P_i by modifying the state as follows:
   $Available = Available - Request;$
   $Allocation_i = Allocation_i + Request_i;$
   $Need_i = Need_i - Request_i;$
   - *If safe ⇒ the resources are allocated to Pi*
   - *If unsafe ⇒ Pi must wait, and the old resource-allocation state is restored.*

Example-

Example: Consider a system that contains five processes P1, P2, P3, P4, P5 and the three resource types A, B and C. Following are the resources types: A has 10, B has 5 and the resource type C has 7 instances.

Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P1	0	1	0	7	5	3	3	3	2
P2	2	0	0	3	2	2			
P3	3	0	2	9	0	2			
P4	2	1	1	2	2	2			
P5	0	0	2	4	3	3			

Need [i] = Max [i] - Allocation [i]
Need for P1: (7, 5, 3) - (0, 1, 0) = 7, 4, 3
Need for P2: (3, 2, 2) - (2, 0, 0) = 1, 2, 2
Need for P3: (9, 0, 2) - (3, 0, 2) = 6, 0, 0
Need for P4: (2, 2, 2) - (2, 1, 1) = 0, 1, 1
Need for P5: (4, 3, 3) - (0, 0, 2) = 4, 3, 1

Process	Need		
	A	B	C
P1	7	4	3
P2	1	2	2
P3	6	0	0
P4	0	1	1
P5	4	3	1

Hence, we created the context of need matrix.

Ans. 2: Apply the Banker's Algorithm:

Available Resources of A, B and C are 3, 3, and 2.

Now we check if each type of resource request is available for each process.

Step 1: For Process P1:

Need <= Available

7, 4, 3 <= 3, 3, 2 condition is **false**.

So, we examine another process, P2.

Step 2: For Process P2:

Need <= Available

1, 2, 2 <= 3, 3, 2 condition **true**

New available = available + Allocation

(3, 3, 2) + (2, 0, 0) => 5, 3, 2

Similarly, we examine another process P3.

Step 3: For Process P3:

P3 Need <= Available

6, 0, 0 < = 5, 3, 2 condition is **false**.

Similarly, we examine another process, P4.

Step 4: For Process P4:

P4 Need <= Available

0, 1, 1 <= 5, 3, 2 condition is **true**

New Available resource = Available + Allocation

5, 3, 2 + 2, 1, 1 => 7, 4, 3

Similarly, we examine another process P5.

Step 5: For Process P5:

P5 Need <= Available

4, 3, 1 <= 7, 4, 3 condition is **true**

New available resource = Available + Allocation

7, 4, 3 + 0, 0, 2 => 7, 4, 5

Now, we again examine each type of resource request for processes P1 and P3.

Step 6: For Process P1:

P1 Need <= Available

7, 4, 3 <= 7, 4, 5 condition is **true**

New Available Resource = Available + Allocation

7, 4, 5 + 0, 1, 0 => 7, 5, 5

So, we examine another process P2.

Step 7: For Process P3:

P3 Need <= Available

6, 0, 0 <= 7, 5, 5 condition is true

New Available Resource = Available + Allocation

7, 5, 5 + 3, 0, 2 => 10, 5, 7

Hence, we execute the banker's algorithm to find the safe state and the safe sequence like P2, P4, P5, P1 and P3.

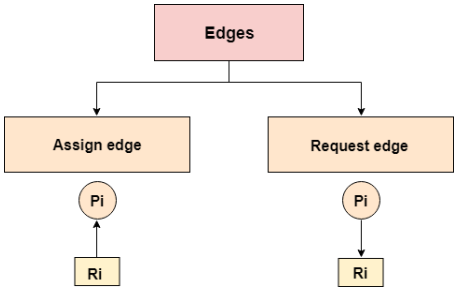
Q.12 Explain the concept of resource allocation graph.

Ans. The resource allocation graph is the pictorial representation of the state of a system. As its name suggests, the resource allocation graph is the complete information about all the processes which are holding some resources or waiting for some resources.

It also contains the information about all the instances of all the resources whether they are available or being used by the processes.

In Resource allocation graph, the process is represented by a Circle while the Resource is represented by a rectangle.

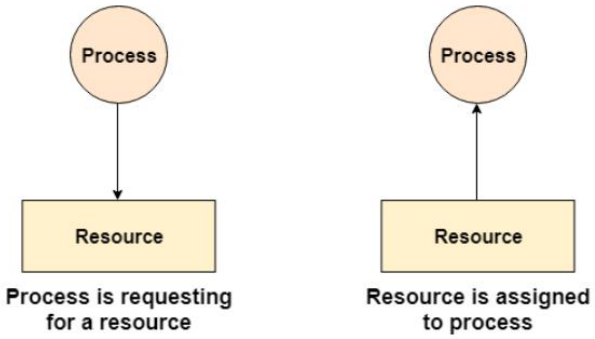
Vertices are mainly of two types, Resource and process. Each of them will be represented by a different shape. Circle represents process while rectangle represents resource.



Edges in RAG are also of two types, one represents assignment and other represents the wait of a process for a resource. The above image shows each of them.

A resource is shown as assigned to a process if the tail of the arrow is attached to an instance to the resource and the head is attached to a process.

A process is shown as waiting for a resource if the tail of an arrow is attached to the process while the head is pointing towards the resource.

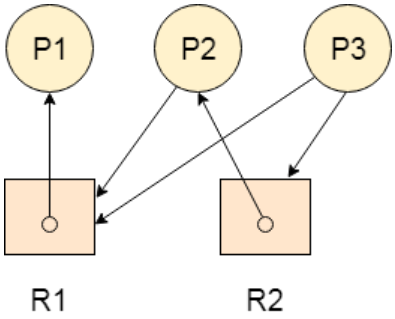


Example

Let's consider 3 processes P1, P2 and P3, and two types of resources R1 and R2. The resources are having 1 instance each.

According to the graph, R1 is being used by P1, P2 is holding R2 and waiting for R1, P3 is waiting for R1 as well as R2.

The graph is deadlock free since no cycle is being formed in the graph.



Q.13 What is scheduler? Explain various types of scheduler.

Ans. The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

There are three types of process scheduler.

1. **Long Term or job scheduler:**
   It brings the new process to the 'Ready State'. It controls ***Degree of Multi-programming***, i.e., number of process present in ready state at any point of time. It is important that the long-term scheduler make a careful selection of both I/O and CPU-bound processes. I/O bound tasks are which use much of their time in input and output operations while CPU bound processes are which spend their time on CPU. The job scheduler increases efficiency by maintaining a balance between the two.

2. **Short term or CPU scheduler:**
   It is responsible for selecting one process from ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn't load the process on running. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring there is no starvation owing to high burst time processes.
   Dispatcher is responsible for loading the process selected by Short-term scheduler on the CPU (Ready to Running State) Context switching is done by dispatcher only. A dispatcher does the following:
   1. Switching context.
   2. Switching to user mode.
   3. Jumping to the proper location in the newly loaded program.

3. **Medium-term scheduler:**
   It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). 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. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.

Q.14 Explain deadlock avoidance.

Ans. Requires that the system has some additional *a priori* information available.

- Simplest and most useful model requires that each process declare the *maximum number* of resources of each type that it may need.
- The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition.
- Resource-allocation *state* is defined by the number of available and allocated resources, and the maximum demands of the processes.

Q.15 Explain the concept of safe state with suitable example.

Ans. If the system can allocate resources to the process in such a way that it can avoid deadlock. Then the system is in a safe state.

It's true that "All *safe states* are *deadlock* free", but don't forgot that "all unsafe *states not always* lead *to deadlocks"*.

Safe state in Operating system can be achieved if the system can safely allocate all resources requested by all the processes in the system without entering a deadlock **state**.

Example of safe and unsafe state

Let's see Deadlock avoidance safe and unsafe state example.

Consider the following set of processes to answer deadlock question. Is the system in a safe state

Free resources : 3

PROCESS	Allocated R	Needed resources
P1	4	10
P2	2	4
P3	2	7

Process P1 have 4 resources and 10 resources required for completion.

Process P2 have 2 resources and 4 resources required for completion.

Process P3 have 2 resources and 7 resources required for completion.

Total free resources for P1, P2 and P3 are 3.

Now, let's move to other calculations.

Free resources: 1

PROCESS	Allocated R	Needed resources
P1	4	10
P2	4	4
P3	2	7

Free resources: 5

PROCESS	Allocated R	Needed resources
P1	4	10
P2	0	0
P3	2	7

Free resources : 0

PROCESS	Allocated R	Needed resources
P1	4	10
P2	0	0
P3	7	7

Free resources: 7

PROCESS	Allocated R	Needed resources
P1	4	10
P2	0	0
P3	0	0

Free resources: 1

PROCESS	Allocated R	Needed resources
P1	10	10
P2	0	0
P3	0	0

Free resources: 11

PROCESS	Allocated R	Needed resources
P1	0	0
P2	0	0
P3	0	0

Now, after the termination of processes, we can see following results;

- P1 have 0 resources and 0 resources required because P1 is no more in system.
- P2 have 0 resources and 0 resources required because P2 is no more in system.
- P3 have 0 resources and 0 resources required because P3 is no more in system.

Result: All processes execute successfully, so there is no deadlock and the system is in a safe state.

****

NAME – VISHNU TAPARIA

ROLLNO. – 2203303

DIVISION – TY CSE CORE-1