Mahatma Gandhi Charitable Trust Managed

Shri Labhubhai Trivedi Institute of Engineering & Technology

CREATING CREATORS - SLTIET

BASICS OF OPERATING SYSTEM

BOS - Subject Code: 4330703

PROF. PARESH S CHAVDA
CSE DEPARTMENT
SLTIET, RAJKOT

Unit - 2

Process Management

Sub Topics:

2.1 Overview of Process and Thread
2.2 Process Lifecycle
2.3 Process Control Block
2.4 Scheduling Criteria
2.5 Scheduling Algorithms
2.6 Overview of Scheduler
2.7 Scheduling Queue
2.8 Context Switching
2.9 Critical Section
2.10 Mutual Exclusion
2.11 Deadlock
2.12 Necessary Condition for Deadlock
2.13 Resource Allocation Graph(RAG)

2.1 Overview of Process and Thread

Process:

A process is an instance of a program that is being executed. When we run a program, it does not execute directly. It takes some time to follow all the steps required to execute the program, and following these execution steps is known as a process.

A process in OS can remain in any of the following states:

NEW: A new process is being created.
READY: A process is ready and waiting to be allocated to a processor.
RUNNING: The program is being executed.
WAITING: Waiting for some event to happen or occur.
TERMINATED: Execution finished.

Continue…

Thread:

- A thread is the subset of a process and is also known as the lightweight process. A process can have more than one thread, and these threads are managed independently by the scheduler. All the threads within one process are interrelated to each other. Threads have some common information, such as data segment, code segment, files, etc. Thread contains its own registers, stack, and counter.

Continue…

Types of Threads

1. User Level Thread

- As the name suggests, the user-level threads are only managed by users, and the kernel does not have its information.
- These are faster, easy to create and manage.
- The user-level threads are implemented by user-level libraries, not by the system calls.

Continue…

2. Kernel-Level Thread

- The kernel-level threads are handled by the Operating system and managed by its kernel. These threads are slower than user-level threads because context information is managed by the kernel. To create and implement a kernel-level thread, we need to make a system call.
- The kernel takes all these threads as a single process and handles them as one process only.

Difference between Process and Thread

Sr.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.

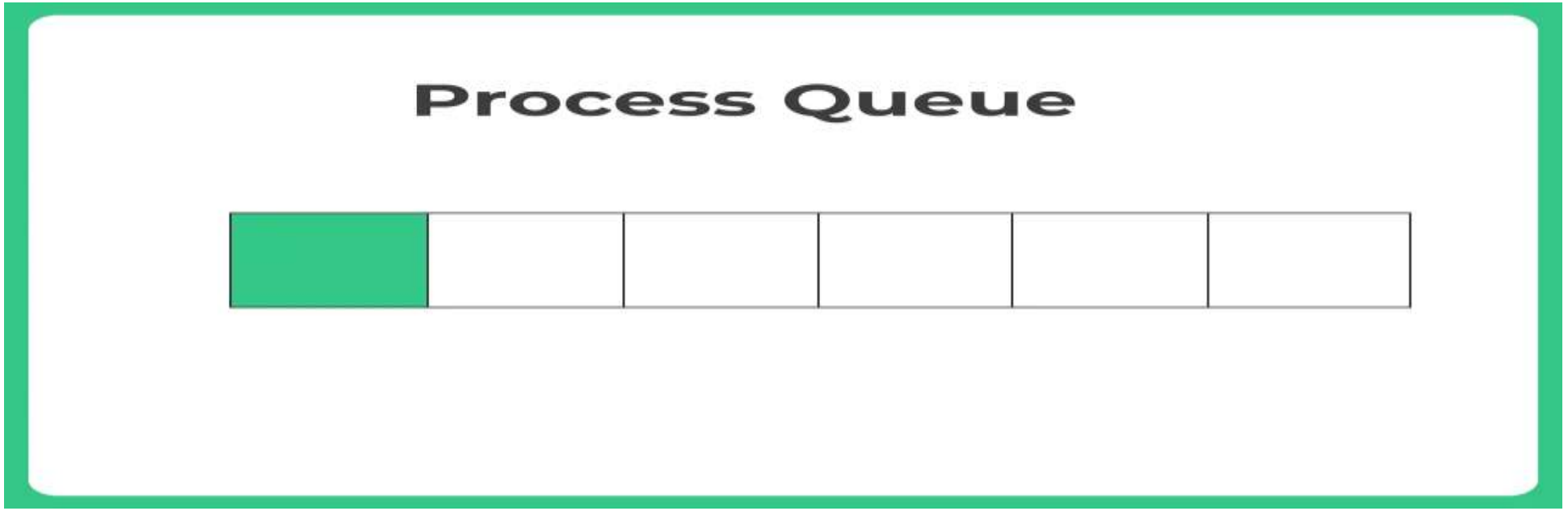
2.2 Process Life Cycle/Process States

A process can be in any of the following states –

- New state
- Ready state
- Running state
- Waiting state
- Terminated

Continue…

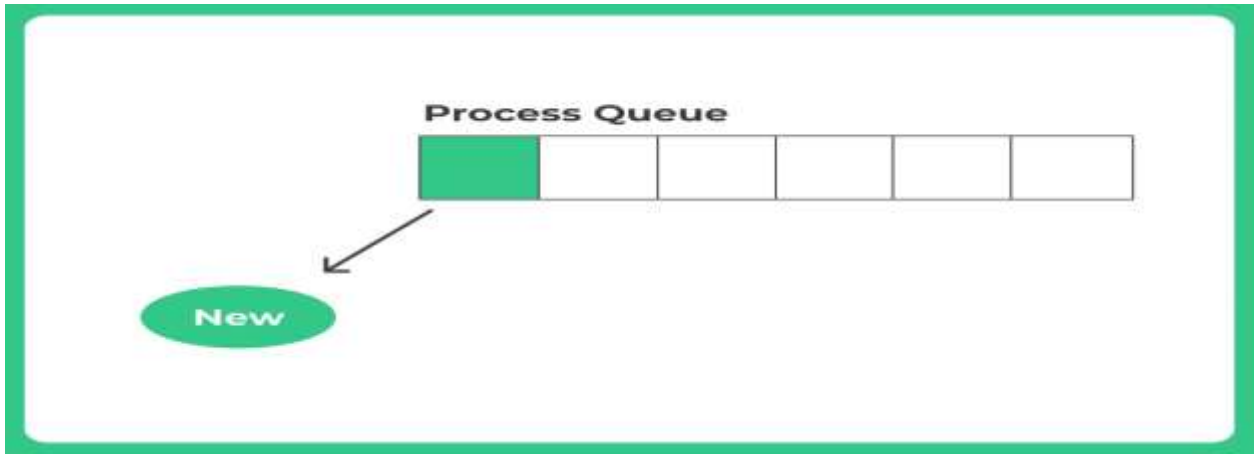
Imagine a unit process that executes a simple addition operation and prints it.



Continue…

NEW :

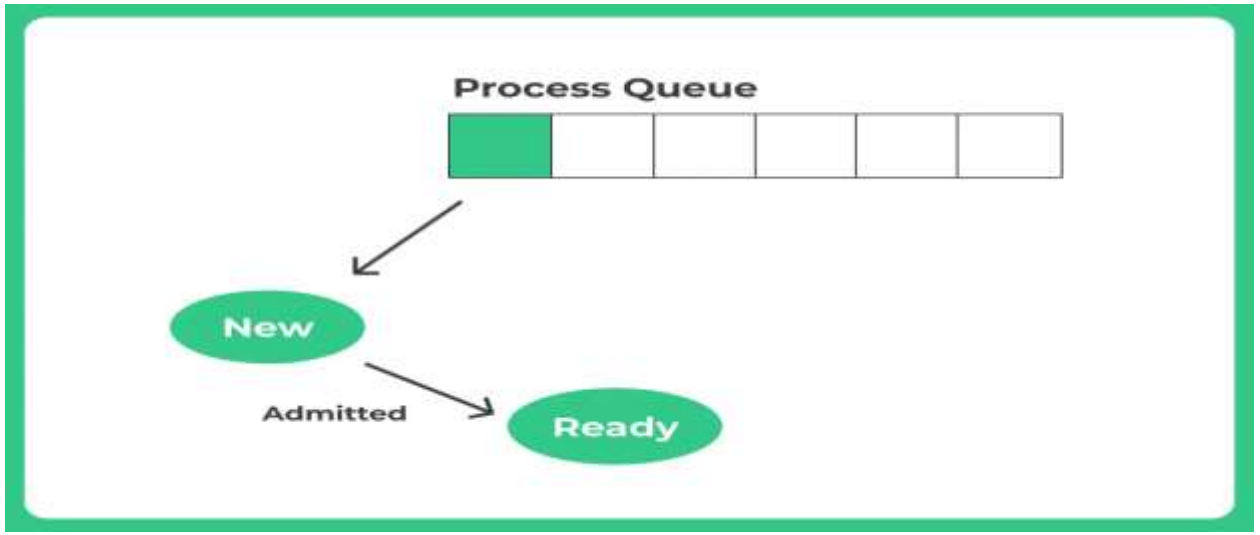
- Process it submitted to the process queue, it in turns acknowledges submission.
- Once submission is acknowledged, the process is given new status.



Continue…

READY STATE :

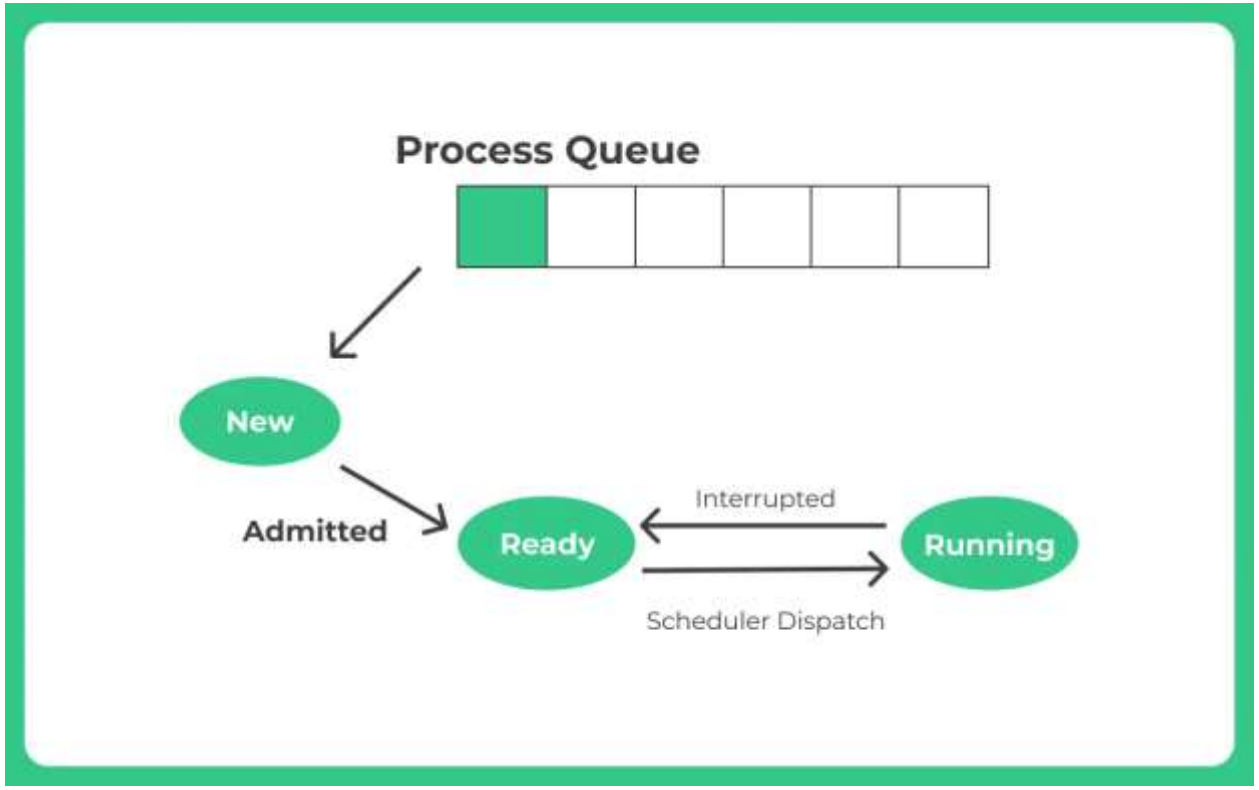
- It then goes to Ready State, at this moment the process is waiting to be assigned a processor by the OS.



Continue…

RUNNING STATE :

- Once the Processor is assigned, the process is being executed and turns in Running State.



Continue…

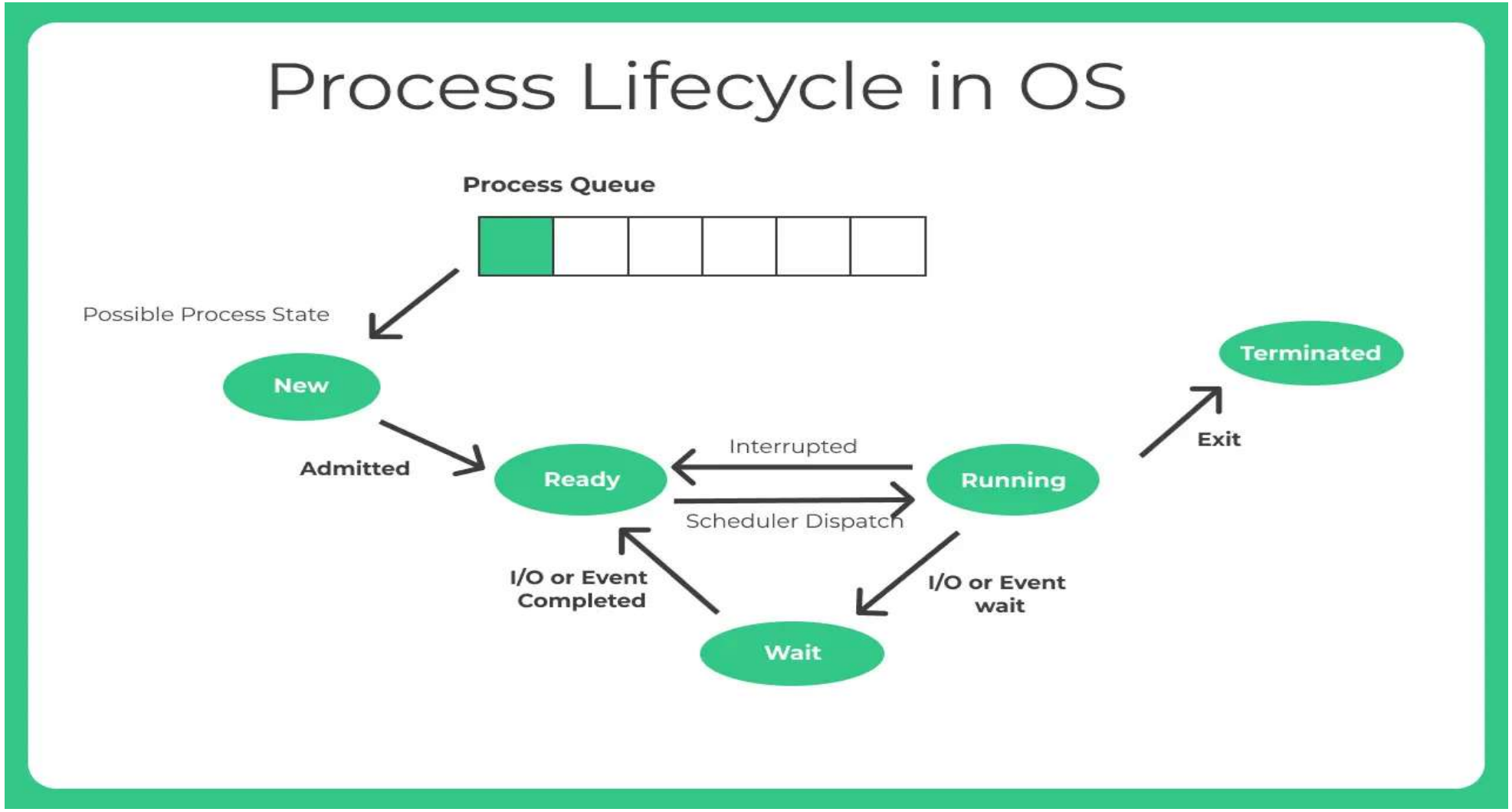
WAIT AND TERMINATION STATE :

Now the process can follow the following transitions –

The process may have all resources it needs and may get directly executed and go to the Termination State. Process may need to go to waiting state any of the following:

- Access to Input/Output device (Asking user the values that are required to be added) via console
- Process maybe intentionally interrupted by OS, as a higher priority operation maybe required, to be completed first
- A resource or memory access that maybe locked by another process, so current process goes to waiting state and waits for the resource to get free.

Continue…


Process Lifecycle in OS
Process Queue
Possible Process State
New
Admitted
Ready
Interrupted
Running
Scheduler Dispatch
Exit
Terminated
I/O or Event Completed
I/O or Event wait
Wait

2.3 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.

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.

Continue…

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.

Continue…

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.

Continue…

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 maintains the list of all open devices which are used during the execution of the process.

2.4 Scheduling Criteria

1) CPU Utilization:

- CPU utilization is a criterion used in CPU scheduling that measures the percentage of time the CPU is busy processing a task.

2) Throughput:

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

3) Burst Time/ Execution 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.

4) 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.

Continue…

5) Turnaround Time :

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

6) Waiting Time :

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

7) Response Time :

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

2.5 Scheduling Algorithms

Important Terms

1. CPU - - - > Central Processing Unit
2. AT - - - > Arrival Time
3. BT - - - > Burst Time
4. WT - - - > Waiting Time
5. TAT - - - > Turn Around Time
6. CT - - - > Completion Time

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

i) First Come First Serve (FCFS)

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.

=> Advantages of FCFS:

o Simple and Easy

o First come, First serve

=> Disadvantages of FCFS:

Although it is easy to implement, but it is poor in performance since the average waiting time is higher as compared to other scheduling algorithms.

Continue…

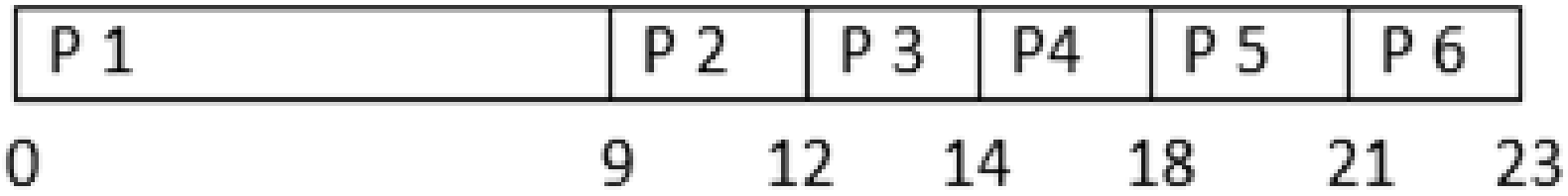
Example :

Sr.No.	Process ID	Arrival Time	Burst Time
1	P1	0	9
2	P2	1	3
3	P3	1	2
4	P4	1	4
5	P5	2	3
6	P6	3	2

Continue…

Answer :

Gantt chart for the above Example is:



Continue…

Sr. No.	Process ID	Arrival Time (T0)	Burst Time (T1)	Completion Time (CT)	Turnaround Time (TAT=CT-T0)	Waiting Time (WT=TAT-T1)
1	P1	0	9	9	9	0
2	P2	1	3	12	11	8
3	P3	1	2	14	13	11
4	P4	1	4	18	17	13
5	P5	2	3	21	19	16
6	P6	3	2	23	20	18

Continue…

The Average Completion Time is:

Average CT = (9 + 12 +14 + 18 + 21 + 23) / 6
Average CT = 97 / 6
Average CT = 16.16667

The Average Turn Around Time is:

Average TAT = (9 + 11 + 13 + 17 + 19 +20) / 6
Average TAT = 89 / 6
Average TAT = 14.83334

The Average Waiting Time is:

Average WT = (0 + 8 + 11 + 13 + 16 + 18) /6
Average WT = 66 / 6
Average WT = 11

ii) Shortest Job First(SJF):

The 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.

=> Advantages of SJF

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

=> Disadvantages of SJF

It is not implementable because the exact Burst time for a process can't be known in advance.

Continue…

Example

Sr.No.	Process ID	Arrival Time	Burst Time
1	P1	1	7
2	P2	3	3
3	P3	6	2
4	P4	7	10
5	P5	9	8

Continue…

Answer

At time = 0

- No, any process arrives.
- Thus, empty slot in gantt chart.

At time = 1

- Process P1 arrives.

At time = 2

- P1 will continue its execution.

Continue…

At time = 3

- Process P2 arrives and is added to the waiting table.
- P1 will continue its execution.

At time = 4

- P1 will continue its execution and P2 is still in the waiting table.

At time = 5

- P1 will continue its execution and P2 is still in the waiting table.

Continue…

At time = 6

- Process P3 arrives and is added to the waiting table.
- P2 is still at the waiting table.
- P1 will continue its execution.

At time = 7

- Process P4 arrives and is added to the waiting table.
- P2 and P3 are still at the waiting table.
- P1 will continue its execution.

Continue…

At time = 8

- Process P1 will finish its execution.
- Then, the burst time of P2, P3 and P4 is compared.
- Process P3 is executed because its burst time is less as compared to P2 and P4.

At time = 9

- Process P5 arrives and is added to the waiting table.
- Process P3 will continue its execution.
- P2 and P4 are still at the waiting table.

Continue…

At time = 10

- Process P3 will finish its execution.
- Then, the burst time of P2, P4 and P5 is compared.
- Process P2 is executed because its burst time is less as compared to P4 and P5.

At time = 13

- Process P2 will finish its execution.
- Then, the burst time of P4 and P5 is compared.
- Process P5 is executed because its burst time is less as compared to P4.

Continue…

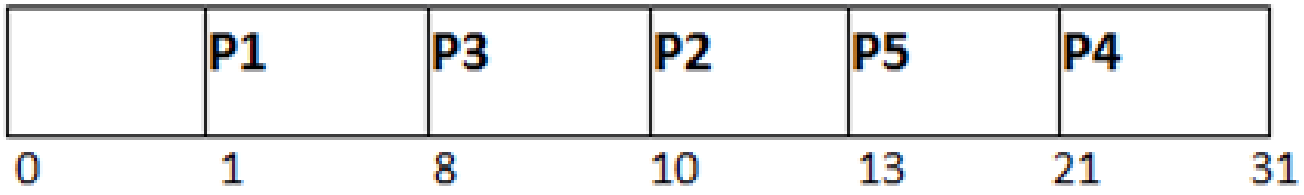
At time = 21

- Process P5 will finish its execution.
- Process P4 will start its execution.

At time = 31

- Process P4 will finish its execution.

Gantt chart for the Example is:



Continue…

Sr. No.	Process ID	Arrival Time(T0)	Burst Time(T1)	Completion Time (CT)	Turnaround Time (TAT=CT-T0)	Waiting Time (WT=TAT-T1)
1	P1	1	7	8	7	0
2	P2	3	3	13	10	7
3	P3	6	2	10	4	2
4	P4	7	10	31	24	14
5	P5	9	8	21	12	4

Continue…

The Average Completion Time is:

- Average CT = (8 + 13 + 10 + 31 + 21) / 5
- Average CT = 83 / 5
- Average CT = 16.6

The Average Waiting Time is:

Average WT = (0 + 7 + 2 + 14 + 4) /5
Average WT = 27 / 5
Average WT = 5.4

The Average Turn Around Time is:

- Average TAT = (7 + 10 + 4 + 24 + 12) / 5
- Average TAT = 57 / 5
- Average TAT = 11.4

iii) 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.
- Round Robin CPU Scheduling uses Time Slot (TS).

Advantages :

1. It can be actually implementable in the system because it is not depending on the burst time.
2. All the jobs get a fare allocation of CPU.

Continue…

Disadvantages :

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

Continue…

Example :

Sr.No.	Process ID	Arrival Time	Burst Time
1	P1	0	7
2	P2	1	4
3	P3	2	15
4	P4	3	11
5	P5	4	20
6	P6	4	9

Continue…

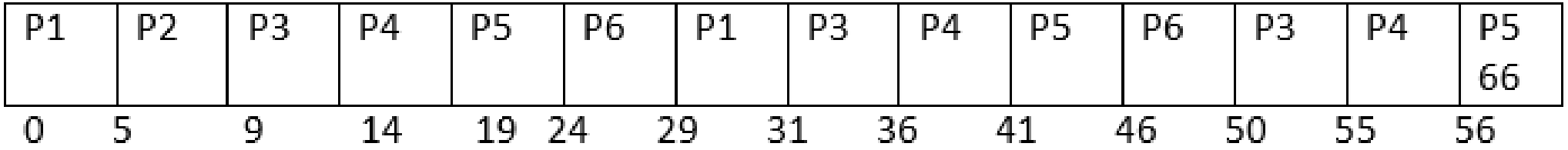
Answer:

Assume Time Slot TS = 5

Ready Queue:

P1, P2, P3, P4, P5, P6, P1, P3, P4, P5, P6, P3, P4, P5

Gantt Chart:



Continue…

Sr. No.	Process ID	Arrival Time(T0)	Burst Time(T1)	Completion Time (CT)	Turnaround Time (TAT=CT-T0)	Waiting Time (WT=TAT-T1)
1	P1	0	7	31	31	24
2	P2	1	4	9	8	4
3	P3	2	15	55	53	38
4	P4	3	11	56	53	42
5	P5	4	20	66	62	42
6	P6	4	9	50	46	37

Continue…

Average Completion Time

Average Completion Time = (31 +9 + 55 +56 +66 + 50) / 6

Average Completion Time = 267 / 6

Average Completion Time = 44.5

Average Waiting Time

Average Waiting Time = (5 + 26 + 5 + 42 + 42 + 37) / 6

Average Waiting Time = 157 / 6

Average Waiting Time = 26.16667

Continue…

Average Turn Around Time

Average Turn Around Time = (31 + 8 + 53 + 53 + 62 + 46) / 6

Average Turn Around Time = 253 / 6

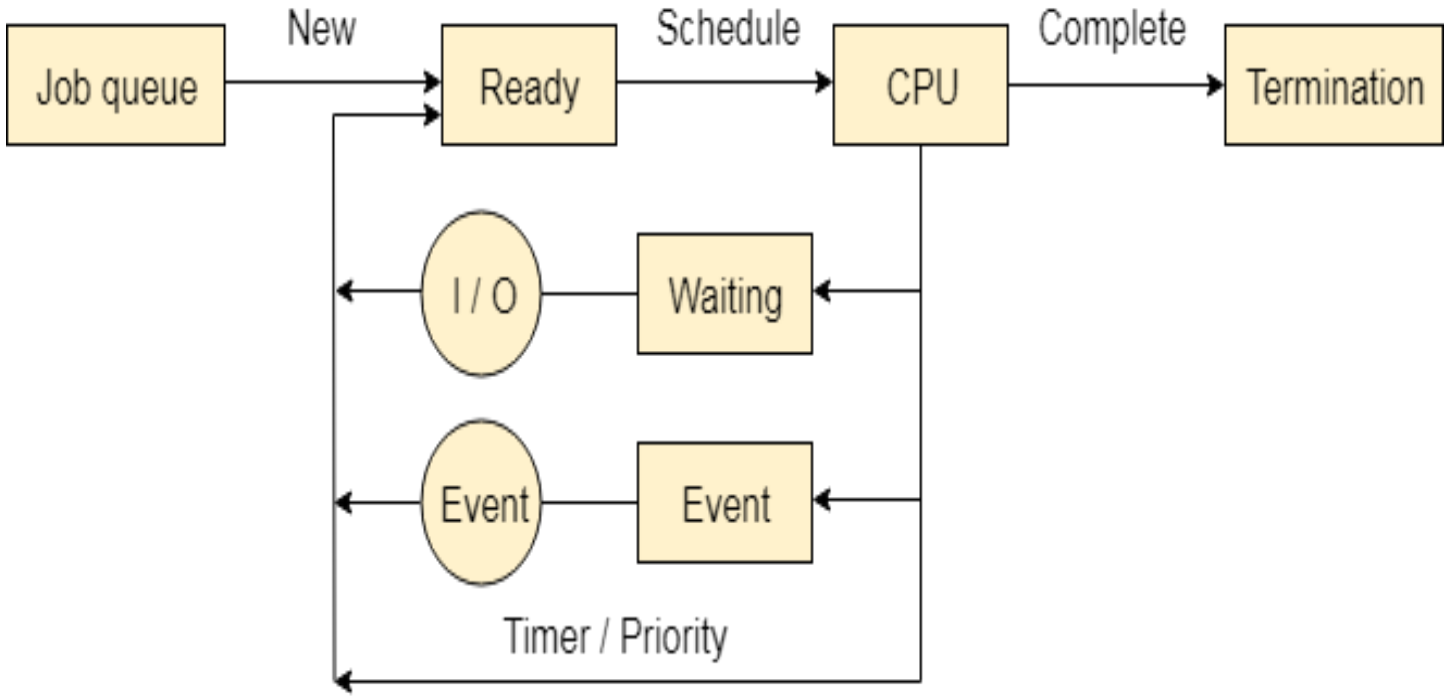
Average Turn Around Time = 42.16667

2.6 Overview of Schedulers

- 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 system. 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.

2.7 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.



Continue…

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

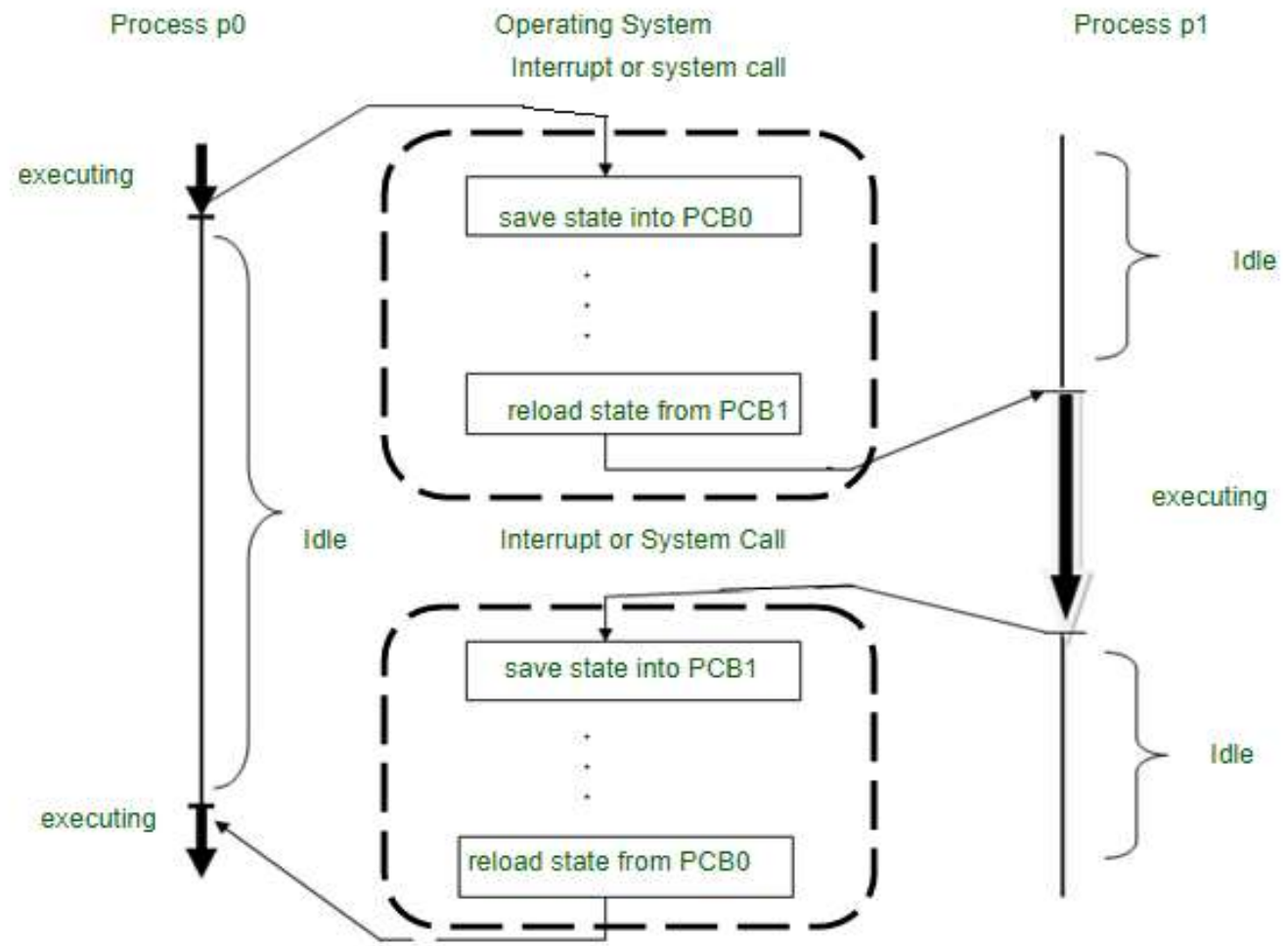
- When the process needs some IO operation in order to complete its execution, OS changes the state of the process from running to waiting. The context (PCB) associated with the process gets stored on the waiting queue which will be used by the Processor when the process finishes the IO.

2.8 Context Switching

- Context switching in an operating system involves saving the context or state of a running process so that it can be restored later, and then loading the context or state of another process and run it.
- Context Switching refers to the process/method used by the system to change the process from one state to another using the CPUs present in the system to perform its job.
- **Example** – Suppose in the OS there (N) numbers of processes are stored in a Process Control Block(PCB). The process is running using the CPU to do its job. While a process is running, other processes with the highest priority queue up to use the CPU to complete their job.

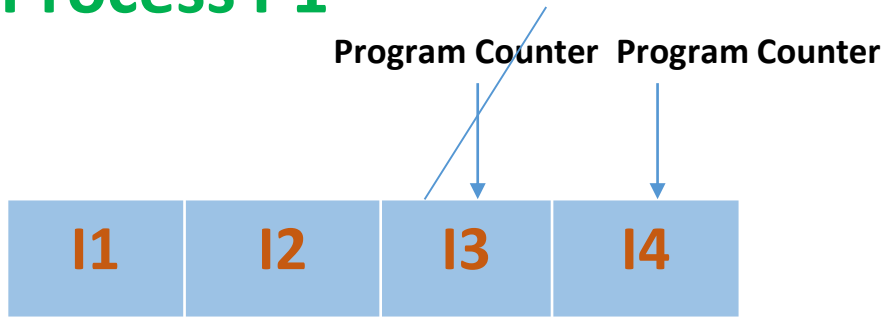
Continue…

State Diagram of Context Switching



Continue…

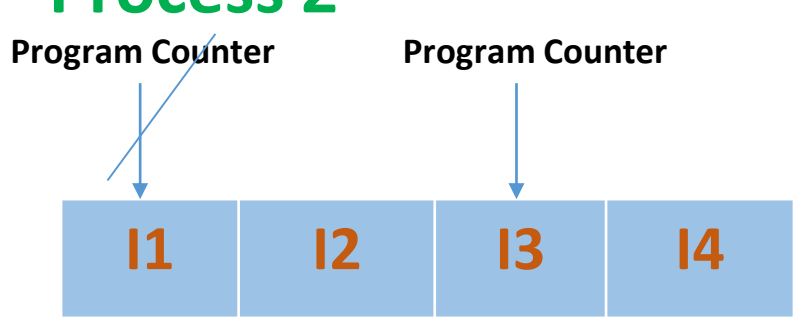
Process P1



I3
3
4
5

I4
6
7
8

Process 2



I1
10
11
12

I3
14
15
16

2.9 Critical Section

- A critical section is a code segment that can be accessed by only one process at a time. The critical section contains shared variables that need to be synchronized to maintain the consistency of data variables.

- In critical section the process may be changing common variable, updating table, writing a file and so on.

- When process 1 is executing in its critical section, no other process is to be allowed to execute in its critical section.

- That means no two processes are executing in their critical section at the same time.

Continue…

```
do {
    entry section
        critical section
    exit section
        remainder section
} while (TRUE);
```

Critical Section :

Each Process must request for permission to critical section.

Entry Section :

In this section, permission of process to enter its critical section is granted.

Exit Section :

In this section, process is exit from its critical section.

Remainder Section :

The remaining code is in this section.

Continue…

Any solution to the critical section problem must satisfy three requirements:

Mutual Exclusion: If a process is executing in its critical section, then no other process is allowed to execute in the critical section.

Progress: If no process is executing in the critical section and other processes are waiting outside the critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter in the critical section next.

Bounded Waiting: A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

2.10 Mutual Exclusion

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.

Mutual exclusion is a property of process synchronization that states that “no two processes can exist in the critical section at any given point of time”. Any process synchronization technique being used must satisfy the property of mutual exclusion.

The requirement of mutual exclusion is that when process P1 is accessing a shared resource R1, another process should be able to access resource R1 until process P1 has finished its operation with resource R1.

Continue…

Condition required for Mutual Exclusion.

Safety : At most one process may execute in the critical section at time.

Liveness : A process requesting enter to the critical section eventually granted It. Liveness implies freedom of deadlock and star vation (If any process not leave it's critical section is called star vation situation)

Fairness : Each process should get a fair chance to execute it's critical section.

Continue…

Performance Matrices :

Synchronization Delay : Time interval between critical section exit and new entry by any process.

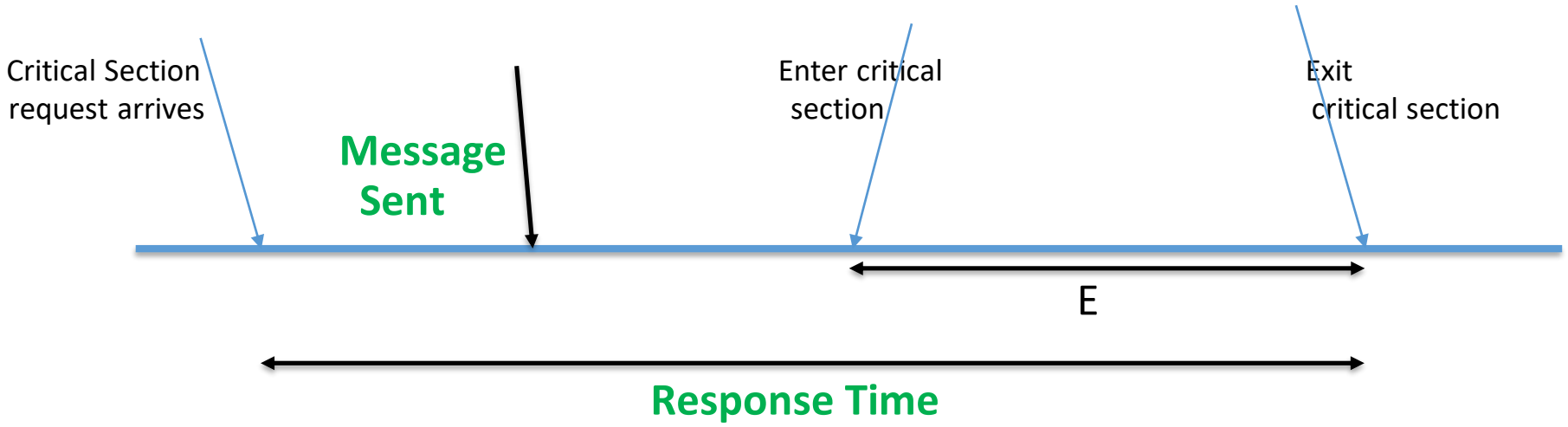
System Throughput : Rate at which request for critical section get executed system throughput = 1/ (Sd + E) where Sd = Synchronization Delay & E = Average C.S. Execute time.

Message Complexity : No. of message that are required / Critical section execution by a process.

Response Time : Time interval from a request send to its critical section execution completed.

Continue…

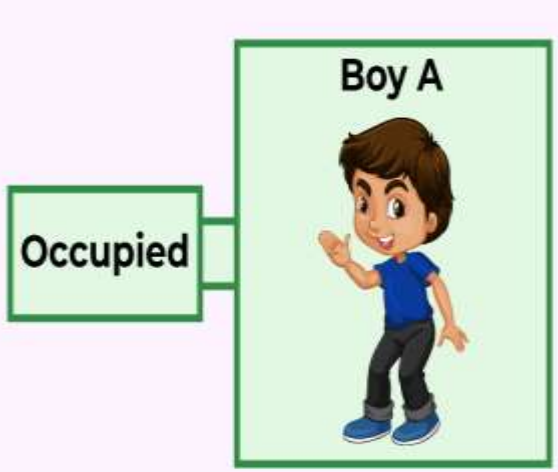

Last site
Exit critical section
Next site
enter critical section
Synchronization Delay


Critical Section
request arrives
Message
Sent
Enter critical
section
Exit
critical section
E
Response Time

Continue…

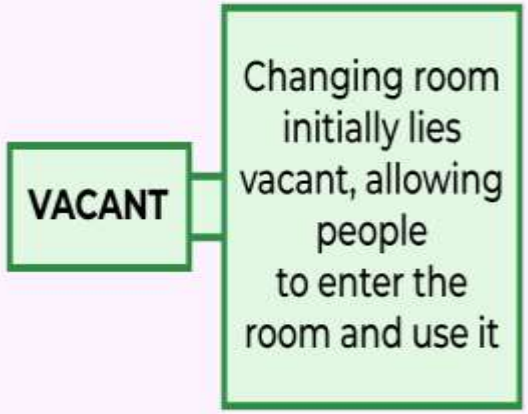
Example :

In the clothes section of a supermarket, two people are shopping for clothes.



Since Boy A is Inside the changing room, the sign on it prevent the other from entering the room.

Boy B has to wait outside the changing room till boy A comes out



Since Boy A is done using the changing room, he vacates it- making the sign outside the room change.

Since the changing room has been vacted by Boy A, Boy B can enter the hanging room.

The changing room is nothing but the critical section, boy A and boy B are two different processes, while the sign outside the changing room indicates the process synchronization mechanism being used.

2.11 DeadLock

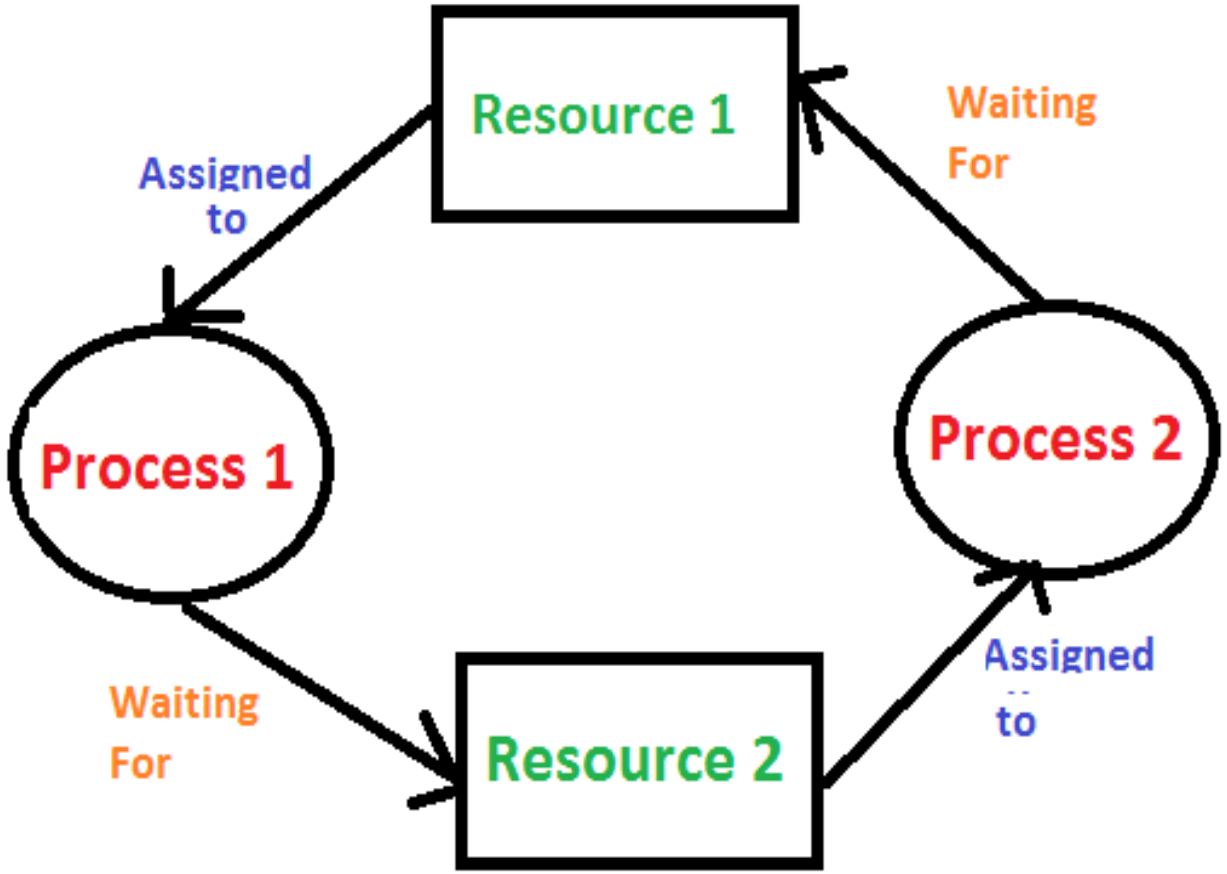
A process in an operating system uses resources in the following way.

- Requests a resource
- Use the resource
- Releases the resource

*A **deadlock*** is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other. A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other.

Continue…



In the this diagram, Process 1 is holding resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.

2.12 Necessary Conditions for DeadLock

A Deadlocked system must satisfy the following 4 conditions.

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

2. 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.

3. 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.

4. 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.

2.13 Resource Allocation Graph(RAG)

- 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 the Resource allocation graph, the process is represented by a Circle while the Resource is represented by a rectangle. Let's see the types of vertices and edges in detail.
- We know that any graph contains vertices and edges. So RAG also contains vertices and edges.

Continue…

In RAG vertices are two type –

1. Process vertex –

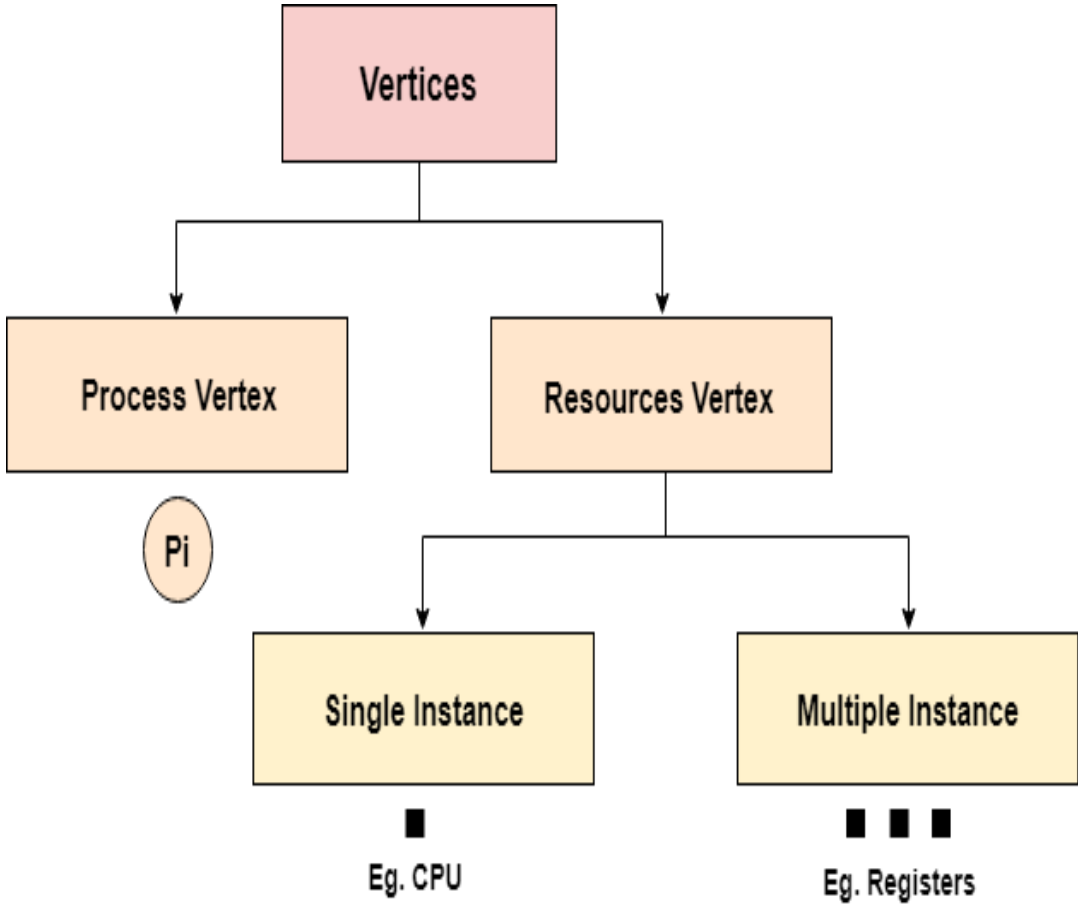
Every process will be represented as a process vertex.

Generally, the process will be represented with a circle.

2. Resource vertex –

Every resource will be represented as a resource vertex.

It is also two type –



i) Single instance type resource –

It represents as a box, inside the box, there will be one dot. So the number of dots indicate how many instances are present of each resource type.

ii) Multi-resource instance type resource –

It also represents as a box, inside the box, there will be many dots present.

Continue…

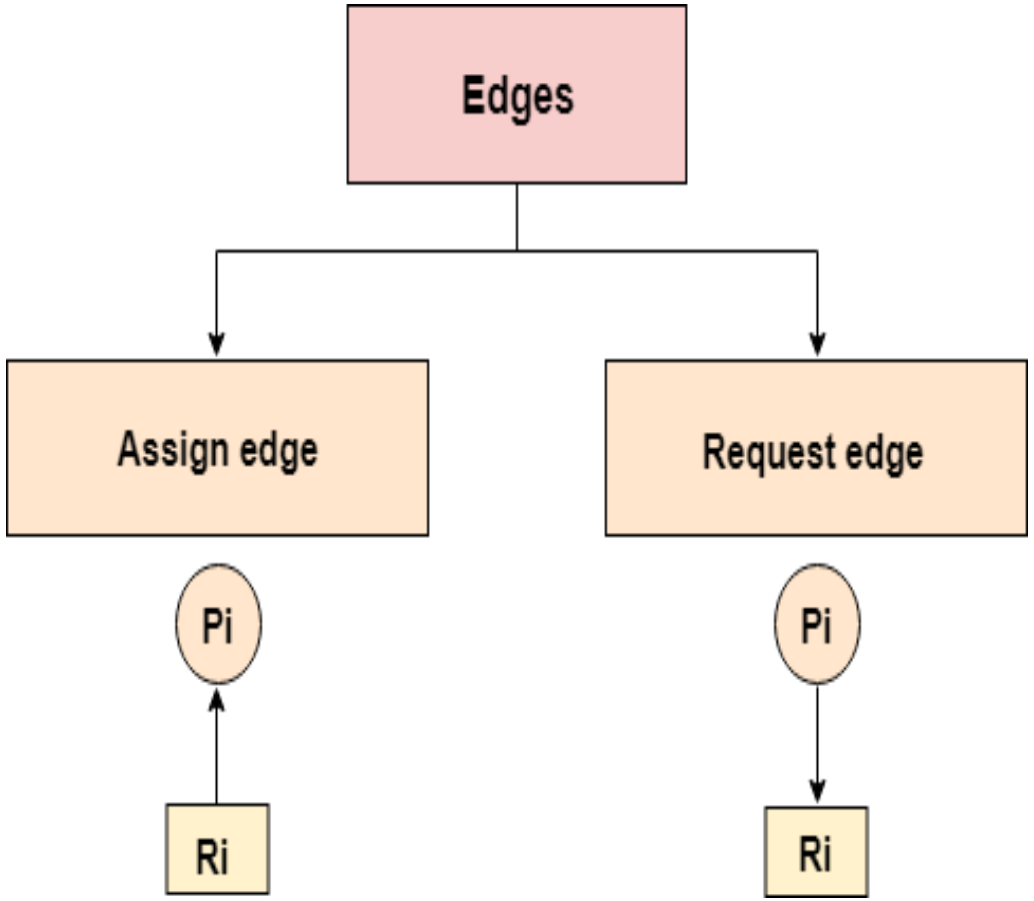
In RAG two types of edge –

1. Assign Edge –

If you already assign a resource to a process then it is called Assign edge.

2. Request Edge –

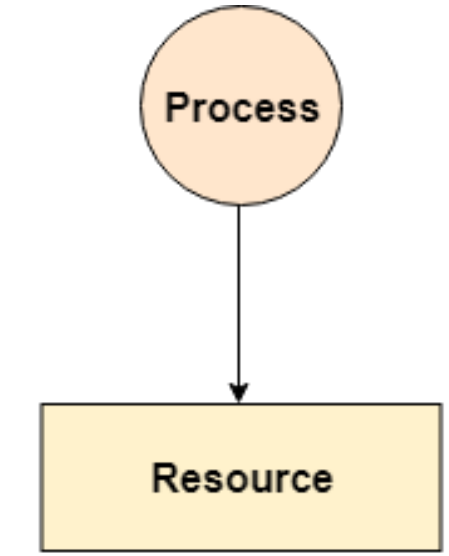
It means in future the process might want some resource to complete the execution, that is called request edge.



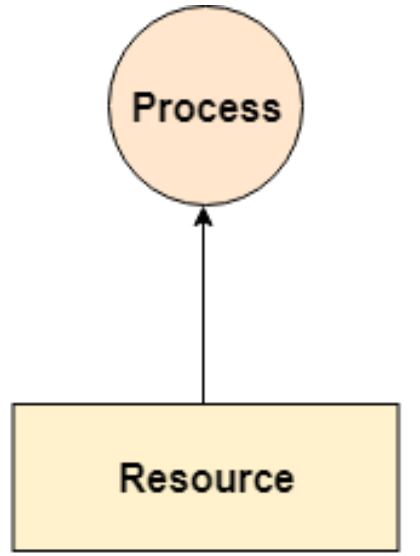
Continue…

- 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.



Process is requesting for a resource

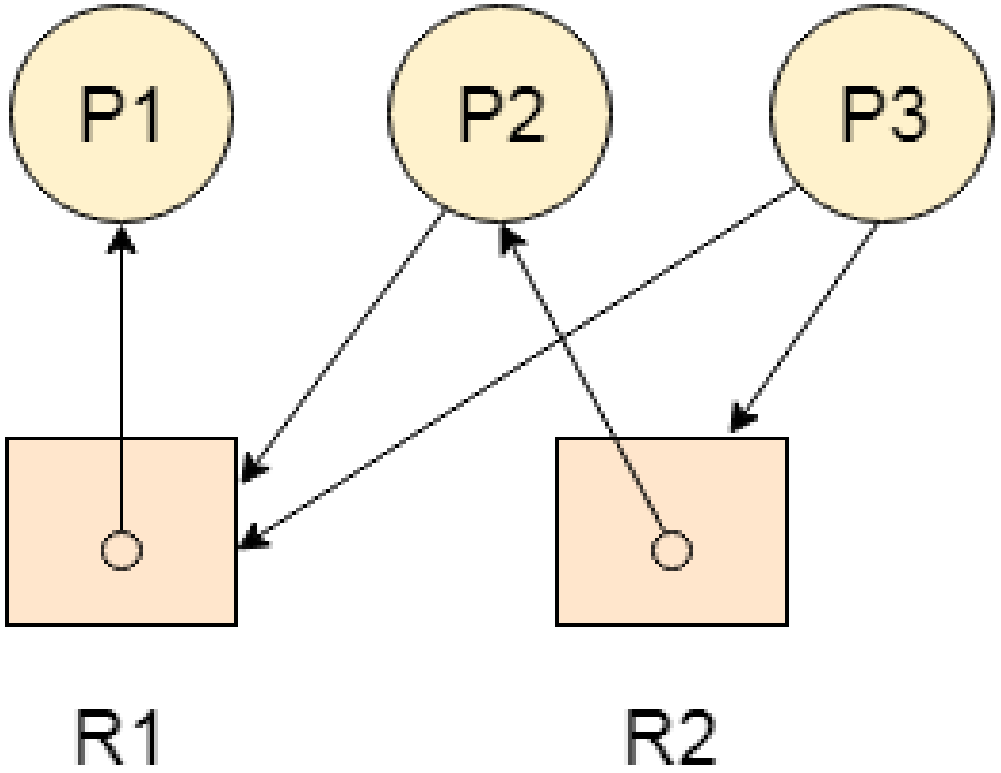


Resource is assigned to process

Continue…

Example:

- Let's consider 3 processes P1, P2, 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.



Questions

Q-1) Explain Process Control Block (PCB) in details.

Q-2) Difference between Process and Thread.

Q-3) Write about Process Lifecycle.

Q-4) Apply all scheduling algorithm(FCFS, SJF[Non-Preemptive & Preemptive], RR) on below example.

Sr.No.	Process ID	Arrival Time	Burst Time
1	P1	0	4
2	P2	1	7
3	P3	2	8
4	P4	4	5
5	P5	6	2

Questions

Q-5) What is scheduling queue? Explain it.

Q-6) Explain Context Switching with details.

Q-7) Explain Critical Section and Mutual Exclusion.

Q-8) Write down Deadlock with its necessary conditions.

Q-9) Explain Resource Allocation Graph with example.

Thank You