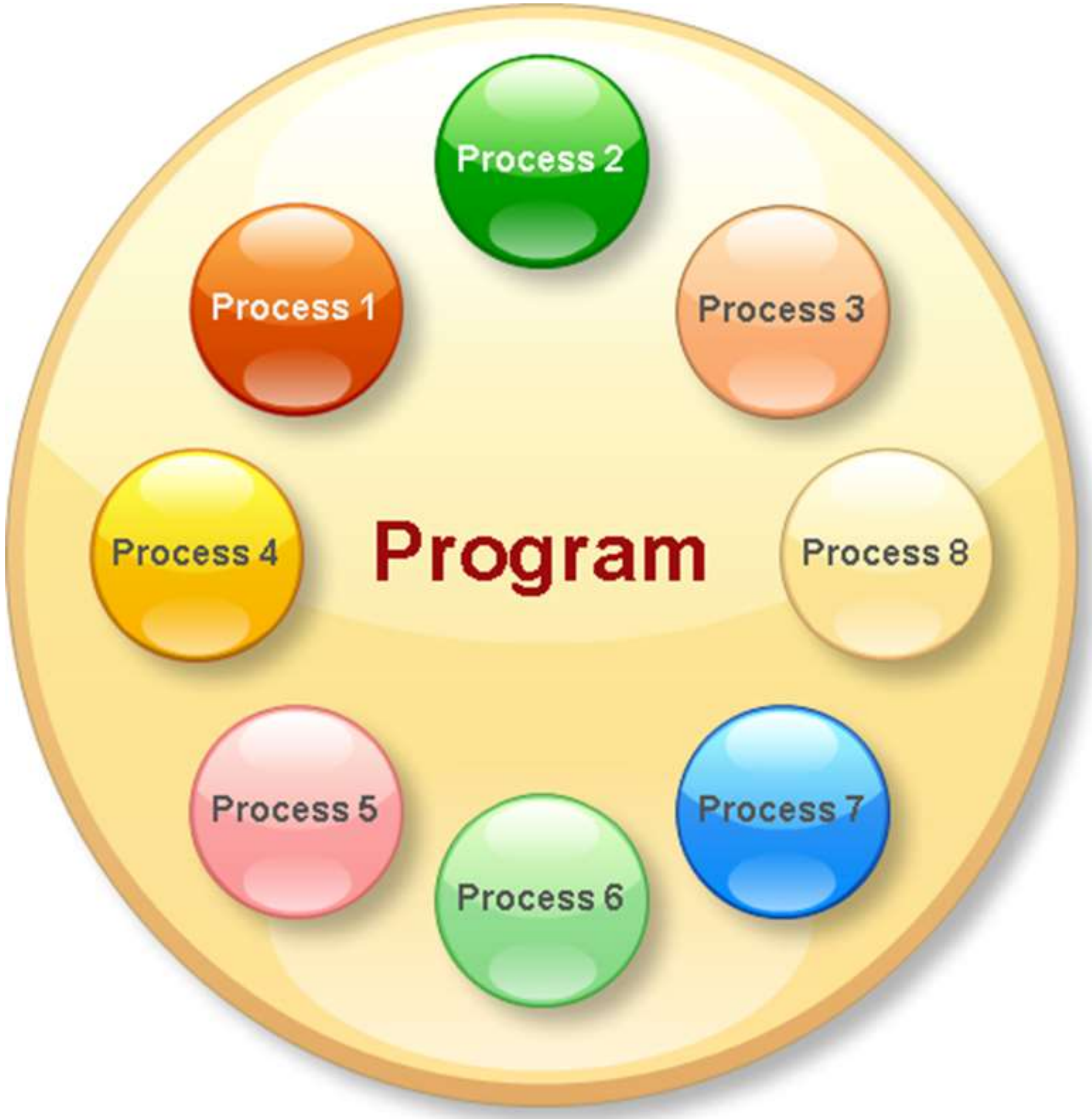
Process And Concurrency Management

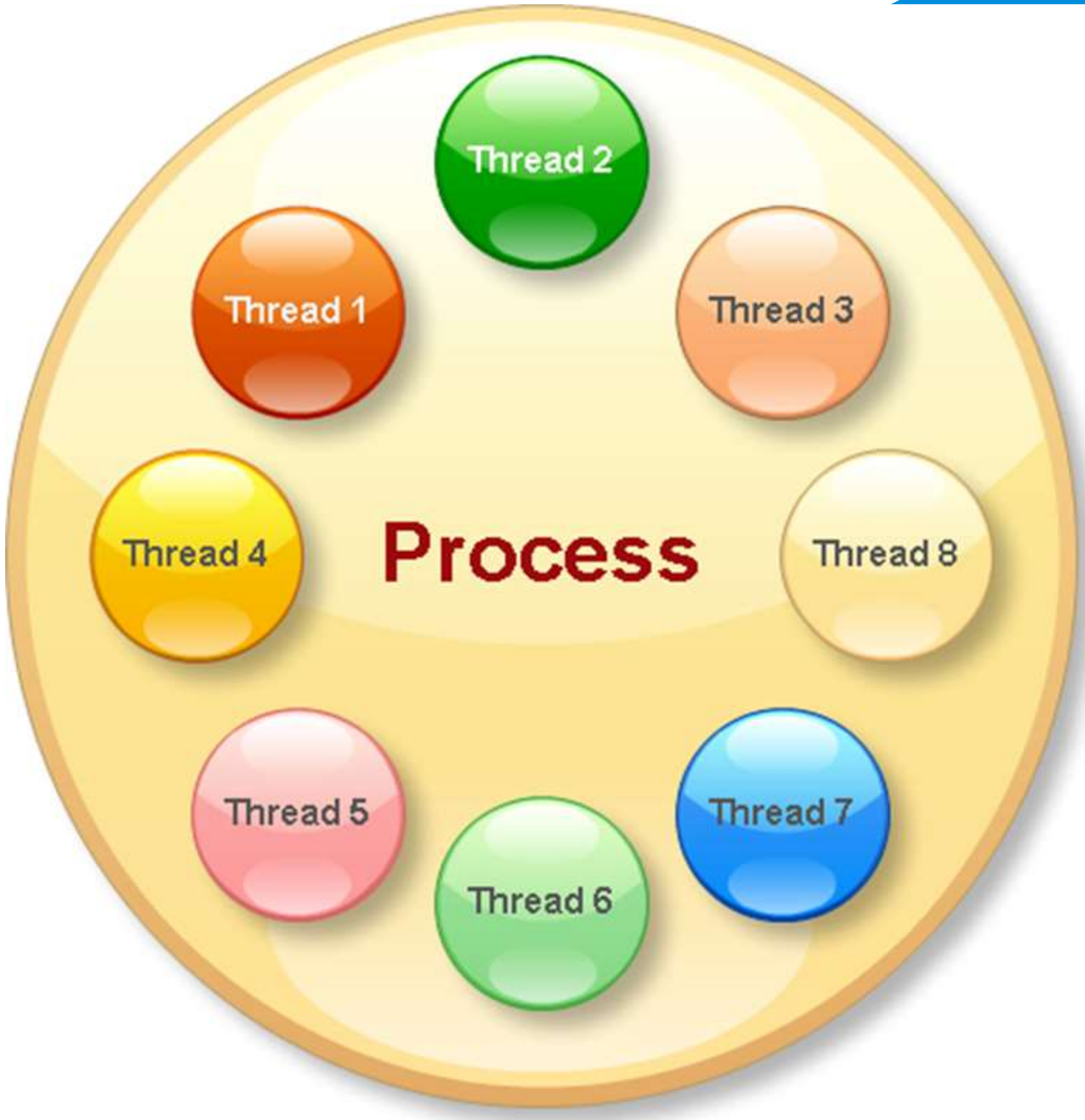
Module 3.1

PROCESS

- A process is basically a program in execution or instance of the program execution.
- The execution of a process must progress in a sequential fashion.
- Process is not as same as program code but a lot more than it.
- A process is an 'active' entity as opposed to program which is considered to be a 'passive' entity.

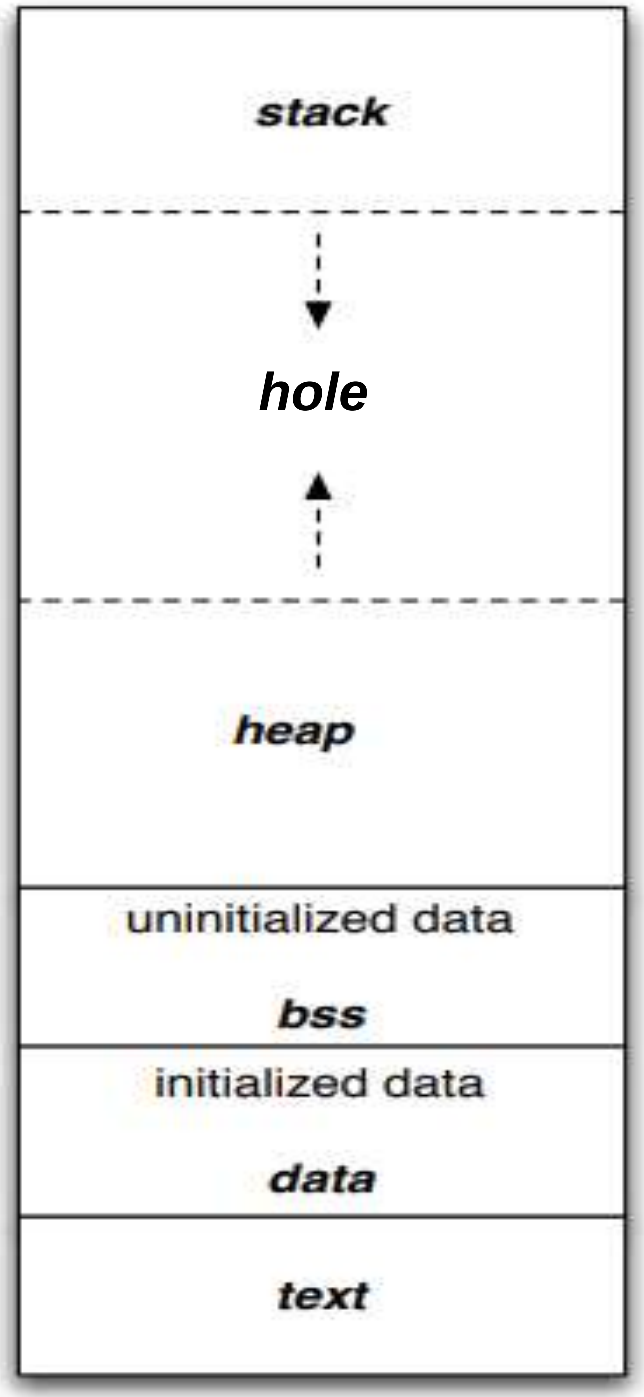
Program, Process and Threads





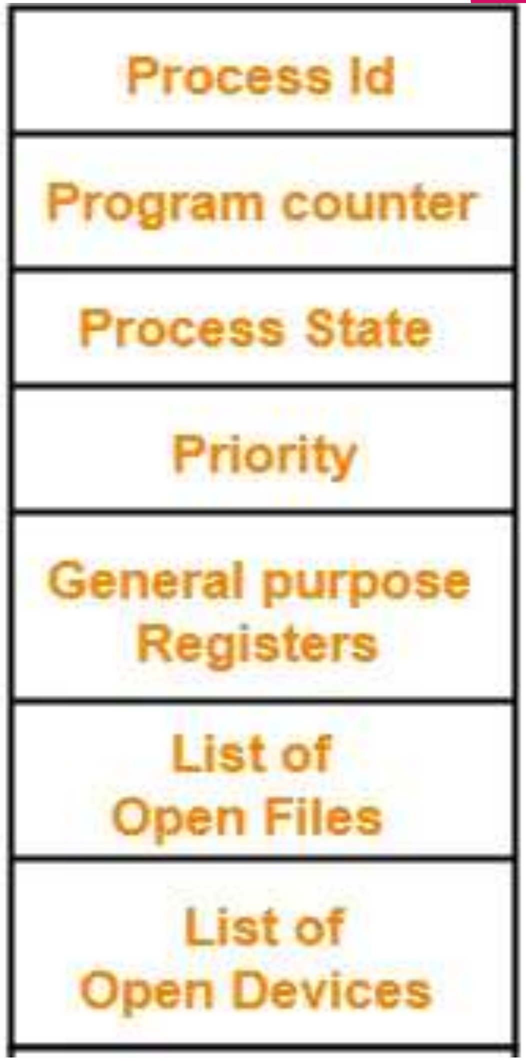
Process Memory

- Process Memory is divided into different regions, which may include:
- **Text/Code Segment:** This is where the executable code of the process is stored.
- **Data Segment:** It holds initialized data and global variables.
- **BSS(Block Start by Symbol) Segment:** This contains uninitialized data and is set to zero.
- **Heap:** Dynamic memory allocation area, used for data structures like arrays and objects.
- **Stack:** Used for function call management, including local variables and function call frames.



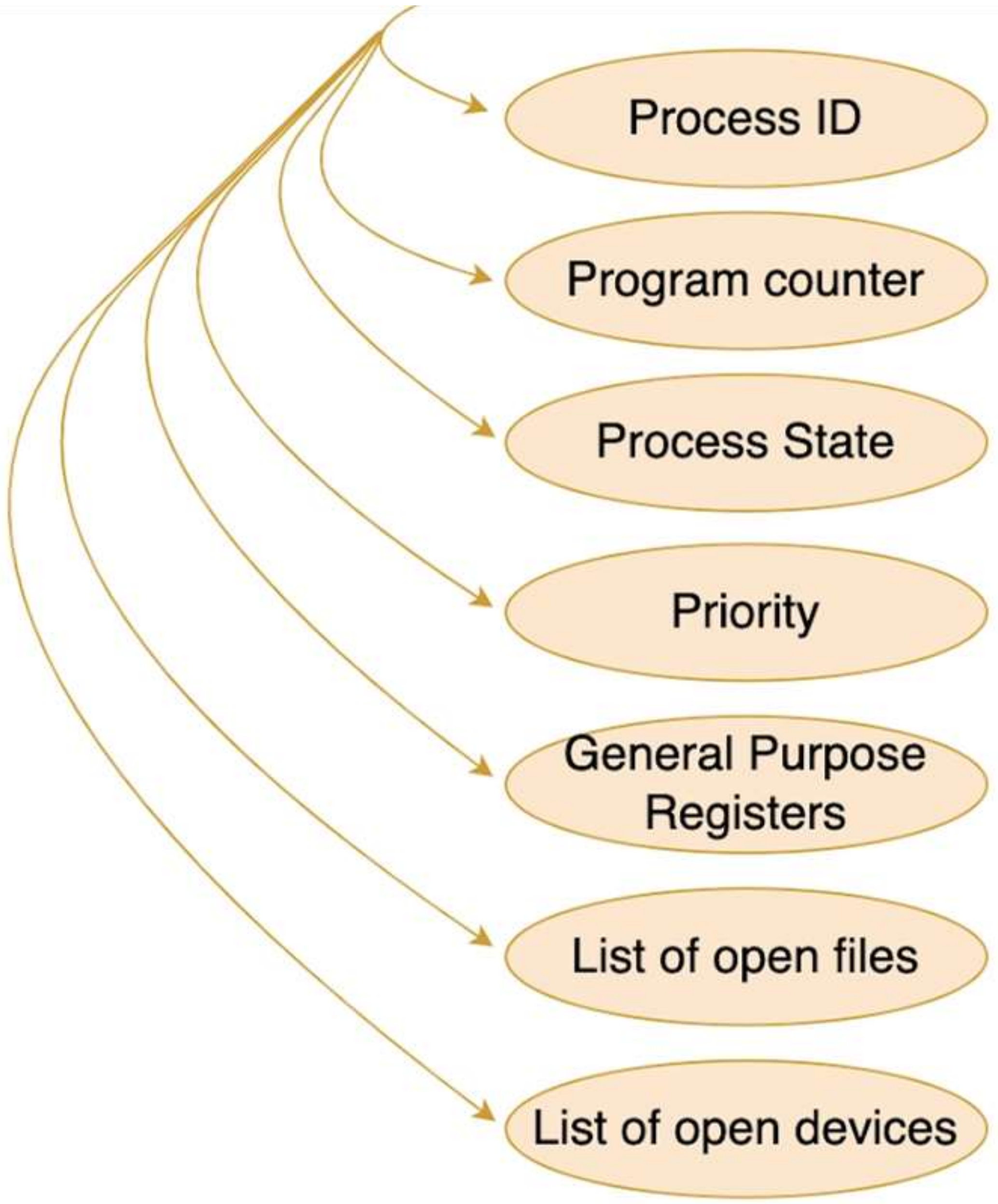
Process Control Block(PCB)

- Process Control Block (PCB) is a data structure that stores information about a particular process.
- This information is required by the CPU while executing the process.
- Each process is identified by its own process control block (PCB).
- It is also called as context of the process.



Process Control Block (PCB)

Process Attributes



Process Attributes

Process ID

- Each process is given a unique id when it is created, and this id is used to identify it across the system.

Program Counter (PC)

- Also known as the instruction pointer.
- Indicates the address of the next instruction to be executed within the process's code.

Process State

- Represents the current condition or status of the process.
- Common process states include "Running," "Ready," "Blocked," "Terminated," and "Suspended."

Process Attributes

Priority

- A numerical or priority level assigned to the process, indicating its relative importance or scheduling priority.
- Higher-priority processes may receive preferential treatment in terms of CPU time.

General Purpose Registers

- Each process has a unique collection of registers used to store data created during the process execution.
- General-purpose registers are used to store data created during the execution of a task.
- Each process has its own set of registers, which its PCB keeps track.

Process Attributes

List of open files

- Every process utilizes files that must be present in the main memory during execution. The OS also keeps track of the open files on the PCB.

List of open devices

- The operating system also keeps track of all available devices used throughout the process execution.

PCB Cont..

- Process Control Block in OS (PCB) plays a critical role in managing process execution and resource allocation in an operating system. PCB acts as a bridge between the process and the operating system.
- It provides the operating system with the necessary information to manage and control the process’s execution.
- PCB manages the process execution by storing information about the process, including process ID, state, priority, CPU registers, program counter, memory information, and other essential information.

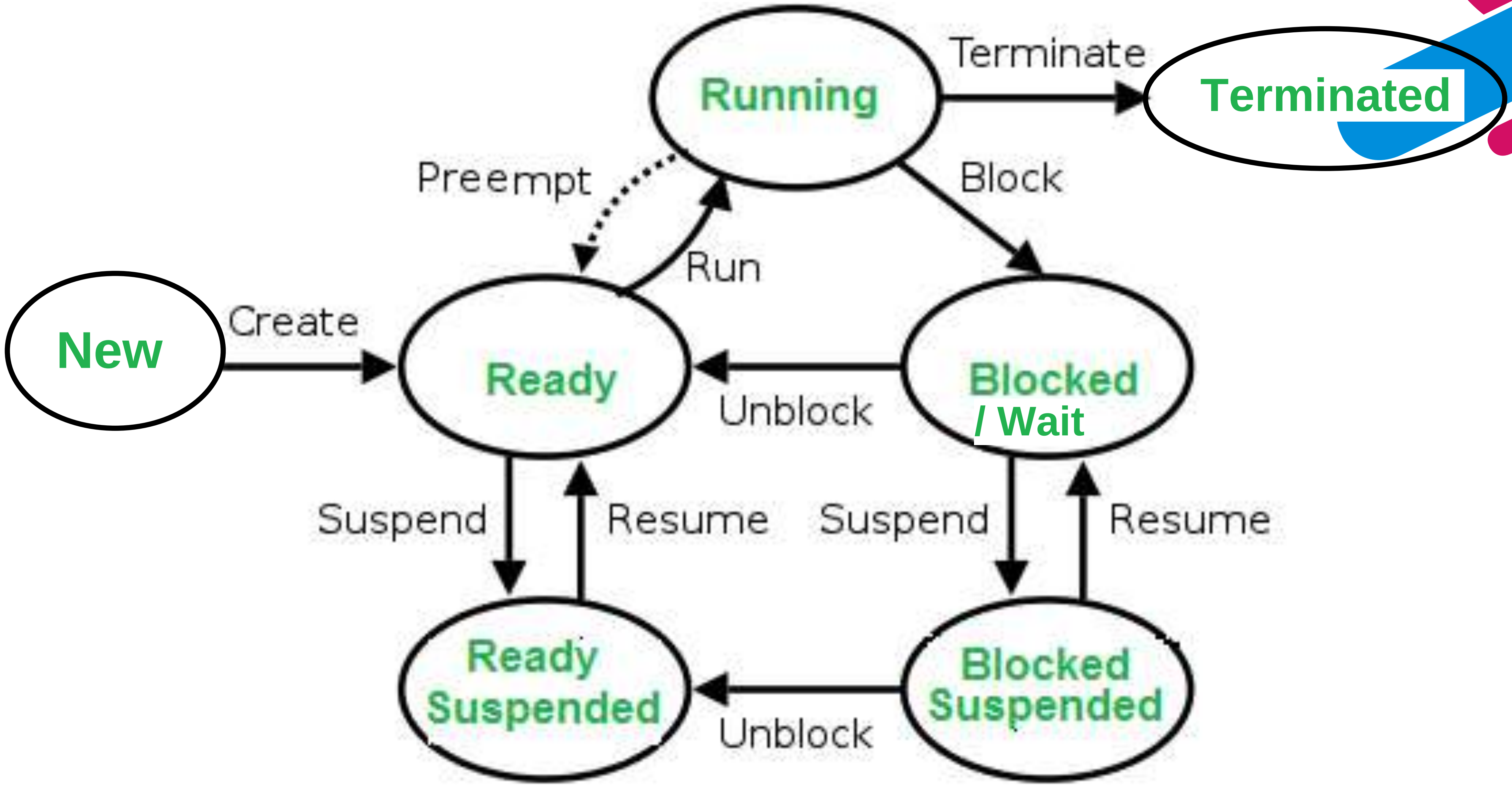
PCB Cont..

- When the process is executing, the operating system refers to the PCB to get information about the process's state and progress.
- If the process is waiting for an event or resource, the operating system updates the process state in the PCB accordingly.
- PCB also manages resource allocation for the process.
- It maintains information about the resources assigned to the process, such as memory, I/O devices, and CPU time.
- PCB keeps track of the resources used by the process and the resources available to the system, ensuring that resources are allocated efficiently and fairly among the processes.

PCB Cont..

- PCB plays a crucial role in preemptive and non-preemptive scheduling. When the operating system decides to switch from one process to another, it saves the current process's context in the PCB and loads the next process's context from its PCB. PCB ensures that the process can resume from where it left off, without losing any critical information.
- In summary, PCB manages process execution and resource allocation by storing essential process information, managing process state and progress, and ensuring efficient and fair resource allocation.

Process States



Process States

New:

- The process is being created, and the operating system is setting up its initial data structures and resources.
- Typically, a process spends a very short amount of time in this state before transitioning to the "Ready" state.

Ready:

- The process has been created and is ready to execute but is waiting for the CPU to be allocated to it.
- Processes in this state are typically waiting in a queue to be scheduled for execution.

Running:

- The process is currently executing on the CPU.
- In a multi-core or multi-processor system, multiple processes can be in the "Running" state simultaneously on different CPUs.

Process States

Blocked/Waiting:

- The process is unable to proceed because it's waiting for a specific event or condition to occur, such as user input, I/O operation completion, or the availability of a resource.
- While blocked, the process is not using CPU time and is waiting for the event to signal it to transition to the "Ready" state.

Terminated:

- The process has finished its execution, either normally or due to an error.
- When a process terminates, its resources are released, and it is removed from the list of active processes.

Process States

Suspended:

- A process can be suspended, which means it is temporarily inactive and not eligible for execution.
- There are typically two types of suspension: "Blocked Suspension" and "Ready Suspension."
  - **Blocked Suspension:** The process is blocked and waiting for an event, similar to the "Blocked" state, but it can be explicitly suspended by the operating system.
  - **Ready Suspension:** The process is in the "Ready" state but has been temporarily removed from the list of processes eligible for execution.

Process States

Zombie (or Defunct)

- A process that has terminated but still has an entry in the process table.
- It exists in this state until its parent process acknowledges its termination, at which point it is removed from the process table entirely.

Ghost (in some systems)

- A process that has been terminated but still has some residual presence in the system, such as open file descriptors or resources that haven't been properly released.

Context Switching

- The process of saving the context of one process and loading the context of another process is known as Context Switching.
- In simple terms, it is like loading and unloading the process from the running state to the ready state.

When Does Context Switching Happen?

- When a high-priority process comes to a ready state (i.e. with higher priority than the running process)
- An Interrupt occurs
- User and kernel-mode switch (It is not necessary though)
- Preemptive CPU scheduling is used.

CPU-Bound vs I/O-Bound Processes

- A **CPU-bound process** requires more CPU time or spends more time in the running state.
- An **I/O-bound process** requires more I/O time and less CPU time. An I/O-bound process spends more time in the waiting state.
- Process planning is an integral part of the process management operating system. It refers to the mechanism used by the operating system to determine which process to run next. The goal of process scheduling is to improve overall system performance by maximizing CPU utilization, minimizing execution time, and improving system response time.

Types of Schedulers

- **Long-term Scheduler – performance:** Decides how many processes should be made to stay in the ready state. This decides the degree of multiprogramming. Once a decision is taken it lasts for a long time which also indicates that it runs infrequently. Hence it is called a long-term scheduler.
- **Short-term Scheduler – Context switching time:** Short-term scheduler will decide which process is to be executed next and then it will call the dispatcher. A dispatcher is a software that moves the process from ready to run and vice versa. In other words, it is context switching. It runs frequently. Short-term scheduler is also called CPU scheduler.

Types of Schedulers

- **Medium-term Scheduler – Swapping time:** Suspension decision is taken by the medium-term scheduler. The medium-term scheduler is used for swapping which is moving the process from main memory to secondary and vice versa. The swapping is done to reduce degree of multiprogramming.

Preemption – Process is forcefully removed from CPU. Pre-emption is also called time sharing or multitasking.

Non-preemption – Processes are not removed until they complete the execution. Once control is given to the CPU for a process execution, till the CPU releases the control by itself, control cannot be taken back forcibly from the CPU.

Operations on the Process

Creation: The process will be ready once it has been created, enter the ready queue (main memory), and be prepared for execution.

Planning: The operating system picks one process to begin executing from among the numerous processes that are currently in the ready queue. Scheduling is the process of choosing the next process to run.

Application: The processor begins running the process as soon as it is scheduled to run. During execution, a process may become blocked or wait, at which point the processor switches to executing the other processes.

Operations on the Process

Killing or Deletion: The OS will terminate the process once its purpose has been fulfilled. The process's context will be over there.

Blocking: When a process is waiting for an event or resource, it is blocked. The operating system will place it in a blocked state, and it will not be able to execute until the event or resource becomes available.

Resumption: When the event or resource that caused a process to block becomes available, the process is removed from the blocked state and added back to the ready queue.

Operations on the Process

Context Switching: When the operating system switches from executing one process to another, it must save the current process's context and load the context of the next process to execute. This is known as context switching.

Inter-Process Communication: Processes may need to communicate with each other to share data or coordinate actions. The operating system provides mechanisms for inter-process communication, such as shared memory, message passing, and synchronization primitives.

Operations on the Process

Process Synchronization: Multiple processes may need to access a shared resource or critical section of code simultaneously. The operating system provides synchronization mechanisms to ensure that only one process can access the resource or critical section at a time.

Process States: Processes may be in one of several states, including ready, running, waiting, and terminated. The operating system manages the process states and transitions between them.

Process 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.

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

Process Queues

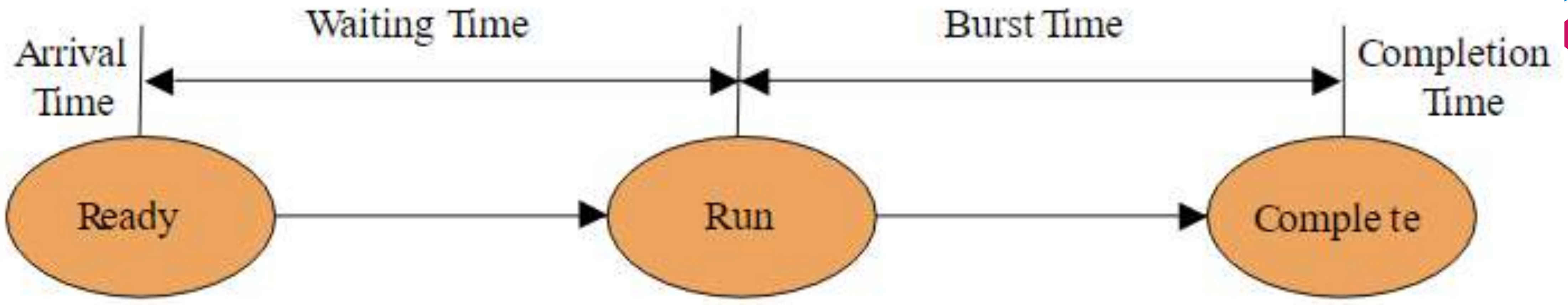
Ready Queue

- 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.

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.

Various Times related to the Process



$$CT - AT = WT + BT$$

$$TAT = CT - AT$$

$$\text{Waiting Time} = TAT - BT$$

TAT	$\longrightarrow$	Turn around tim
BT	$\longrightarrow$	Burst time
AT	$\longrightarrow$	Arrival time

Various Times related to the Process

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.

Various Times related to the Process

Response Time

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

Waiting Time

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

Various Times related to the Process

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.

CPU Scheduling

CPU Scheduling

- In Multiprogramming systems, the Operating system schedules the processes on the CPU to have the maximum utilization of it and this procedure is called CPU scheduling.
- **The Operating System uses various scheduling algorithm to schedule the processes.**

The Purpose of a Scheduling algorithm

- Maximum CPU utilization
- Fare allocation of CPU
- Maximum throughput
- Minimum turnaround time
- Minimum waiting time
- Minimum response time

CPU Scheduling Algorithms

- First-Come, First-Served (FCFS)
- Shortest Job First (SJF) or Shortest Job Next (SJN)
- Shortest Remaining Time (SRT) or Shortest Remaining Time First (SRTF)
- Round Robin (RR)
- Priority Scheduling
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling

First-Come, First-Served (FCFS) Algorithm

- First-Come, First-Served (FCFS)
- This is a non-preemptive scheduling algorithm.
- Processes are executed in the order they arrive in the ready queue.
- It suffers from the "**convoy effect,**" where short processes get delayed behind long processes.

FCFS Algorithm

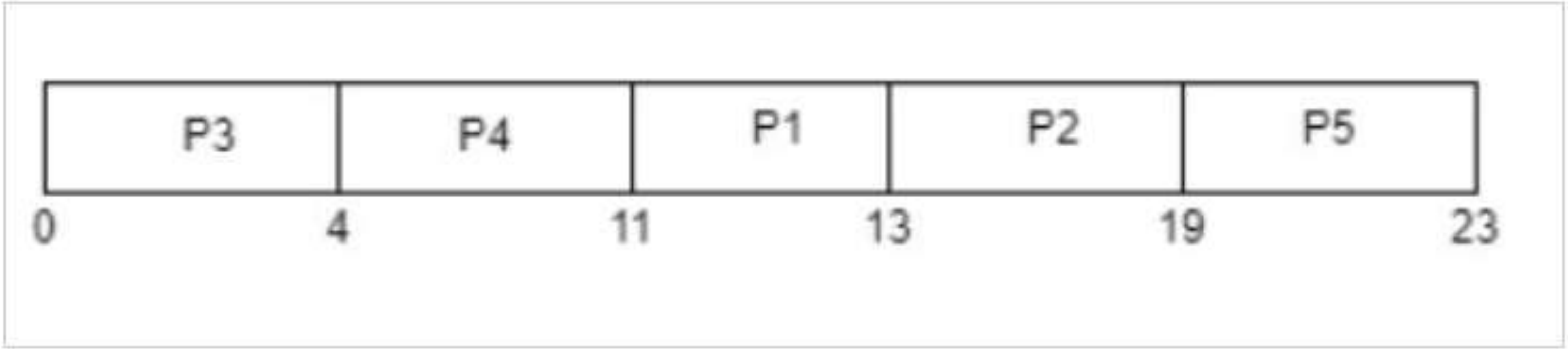
Problem 1:

Consider the set of 5 processes whose arrival time and burst time are given below. If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

Process ID	**Arrival time**	**Burst time**
P1	2	2
P2	5	6
P3	0	4
P4	0	7
P5	7	4

Solution:

Gantt chart



Solution:

For this problem CT, TAT, WT, RT is shown in the given table –

Process ID	Arrival time	Burst time	CT	TAT=CT-AT	WT=TAT-BT	RT
P1	2	2	13	13-2= 11	11-2= 9	9
P2	5	6	19	19-5= 14	14-6= 8	8
P3	0	4	4	4-0= 4	4-4= 0	0
P4	0	7	11	11-0= 11	11-7= 4	4
P5	7	4	23	23-7= 16	16-4= 12	12

Solution:

Average Waiting time = (9+8+0+4+12)/5 = 33/5 = 6.6 time unit (time unit can be considered as milliseconds)

Average Turn-around time = (11+14+4+11+16)/5 = 56/5 = 11.2 time unit (time unit can be considered as milliseconds)

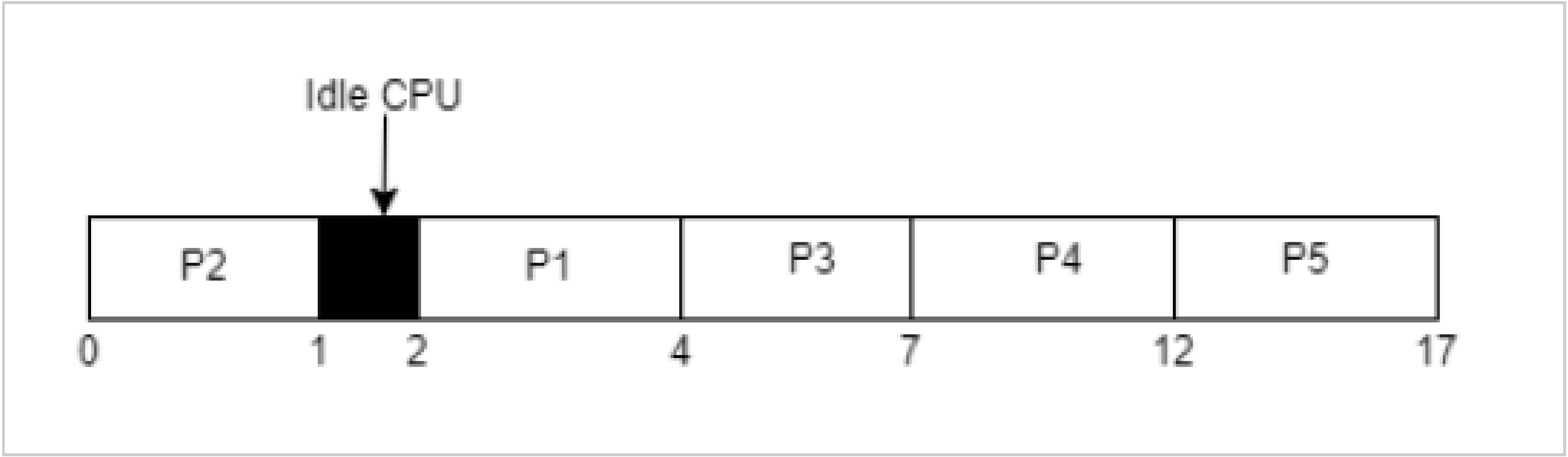
Problem 2

Consider the given table below and find Completion time (CT), Turn-around time (TAT), Waiting time (WT), Response time (RT), Average Turn-around time and Average Waiting time.

Process ID	**Arrival time**	**Burst time**
P1	2	2
P2	0	1
P3	2	3
P4	3	5
P5	4	5

Solution

Gantt chart –



In idle (not-active) CPU period, no process is scheduled to be terminated so in this time it remains void for a little time.

Process ID	Arrival time	Burst time	CT	TAT=CT-AT	WT=TAT-BT	RT
P1	2	2	4	4-2= 2	2-2= 0	0
P2	0	1	1	1-0= 1	1-1= 0	0
P3	2	3	7	7-2= 5	5-3= 2	2
P4	3	5	12	12-3= 9	9-5= 4	4
P5	4	5	17	17-4= 13	13-5= 8	8

- Average Waiting time = (0+0+2+4+8)/5 = 14/5 = 2.8 time unit (time unit can be considered as milliseconds)
- Average Turn-Around time = (2+1+5+9+13)/5 = 30/5 = 6 time unit (time unit can be considered as milliseconds)

Shortest Job First (SJF) or Shortest Job Next (SJN)

- SJF is an algorithm in which the process having the smallest execution time is chosen for the next execution.
- This scheduling method can be preemptive or non-preemptive.
- It significantly reduces the average waiting time for other processes awaiting execution.

There are basically two types of SJF methods:

- Non-Preemptive SJF
- Preemptive SJF

Non-Preemptive SJF

- **In non-preemptive scheduling, once the CPU cycle is allocated to process, the process holds it till it reaches a waiting state or terminated.**
- Consider the following five processes each having its own unique burst time and arrival time.

Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

- At time=0, P4 arrives and starts execution.
- At time= 1, Process P3 arrives. But, P4 still needs 2 execution units to complete. It will continue execution.
- At time =2, process P1 arrives and is added to the waiting queue. P4 will continue execution.
- At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is less compared to P3.
- At time = 4, process P5 arrives and is added to the waiting queue. P1 will continue execution.
- At time = 5, process P2 arrives and is added to the waiting queue. P1 will continue execution.
- At time = 9, process P1 will finish its execution. The burst time of P3, P5, and P2 is compared. Process P2 is executed because its burst time is the lowest.

- At time=10, P2 is executing and P3 and P5 are in the waiting queue.
- At time = 11, process P2 will finish its execution.
- The burst time of P3 and P5 is compared.
- Process P5 is executed because its burst time is lower.
- At time = 15, process P5 will finish its execution.
- At time = 23, process P3 will finish its execution.

Let's calculate the average waiting time for above example.

Wait time

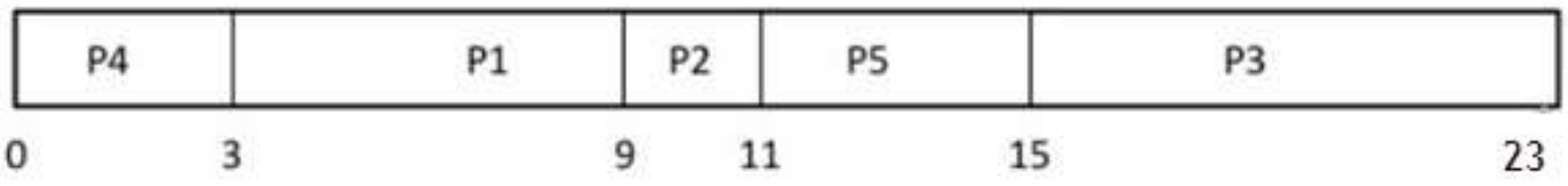
P4= 0-0=0

P1= 3-2=1

P2= 9-5=4

P5= 11-4=7

P3= 15-1=14



Average Waiting Time= 0+1+4+7+14/5 = 26/5 = 5.2

Pre-Emptive SJF

- In Preemptive SJF Scheduling, jobs are put into the ready queue as they come.
- A process with shortest burst time begins execution.
- If a process with even a shorter burst time arrives, the current process is removed or preempted from execution, and the shorter job is allocated CPU cycle.

Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Solution:

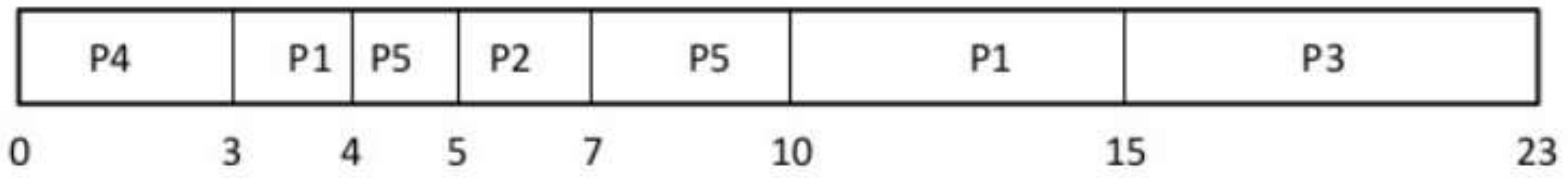
- At time=0, P4 arrives and starts execution.
- At time= 1, Process P3 arrives. But, P4 has a shorter burst time. It will continue execution.
- At time = 2, process P1 arrives with burst time = 6. The burst time is more than that of P4. Hence, P4 will continue execution.
- At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is lower.
- At time = 4, process P5 will arrive. The burst time of P3, P5, and P1 is compared. Process P5 is executed because its burst time is lowest. Process P1 is preempted.
- At time = 5, process P2 will arrive. The burst time of P1, P2, P3, and P5 is compared. Process P2 is executed because its burst time is least. Process P5 is preempted.
- At time =6, P2 is executing.

Solution:

- At time =7, P2 finishes its execution. The burst time of P1, P3, and P5 is compared. Process P5 is executed because its burst time is lesser.
- At time =10, P5 will finish its execution. The burst time of P1 and P3 is compared. Process P1 is executed because its burst time is less.
- At time =15, P1 finishes its execution. P3 is the only process left. It will start execution.
- At time =23, P3 finishes its execution.

Let's calculate the average waiting time for above example: Wait time

- P4= 0-0=0
- P1= (3-2) + 6 =7
- P2= 5-5 = 0
- P5= 4-4+2 =2
- P3= 15-1 = 14



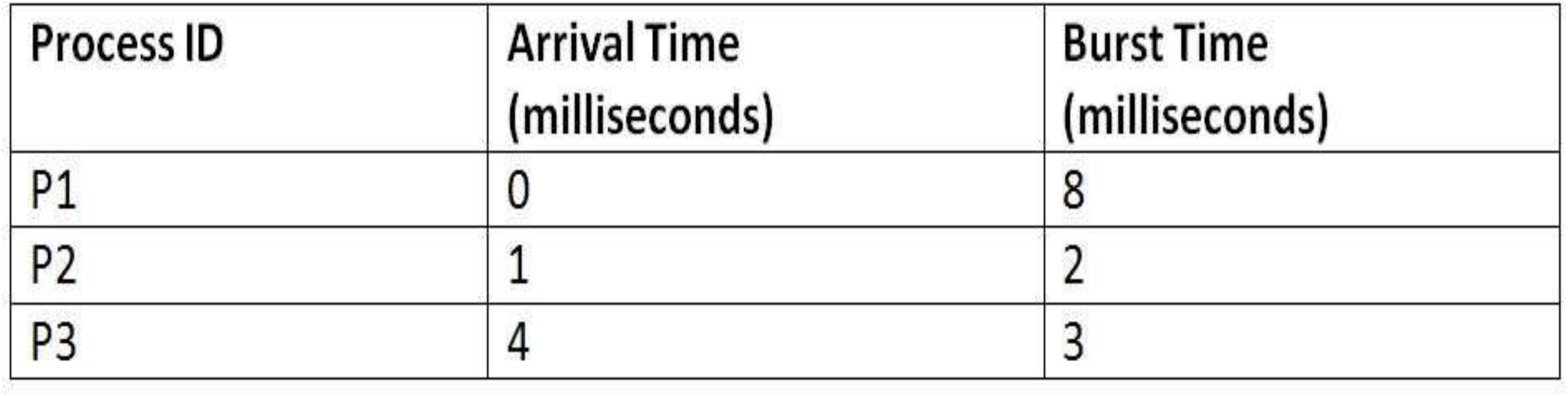
- Average Waiting Time = 0+7+0+2+14/5 = 23/5 =4.6

Shortest Remaining Time (SRT) or Shortest Remaining Time First (SRTF)

- SRTF, Which Stands for Shortest Remaining Time First is a scheduling algorithm used in Operating Systems, which can **also be called as the preemptive version of the SJF scheduling algorithm.**
- The process which has the least processing time remaining is executed first.
- As it is a preemptive type of schedule, it is claimed to be better than SJF scheduling Algorithm.

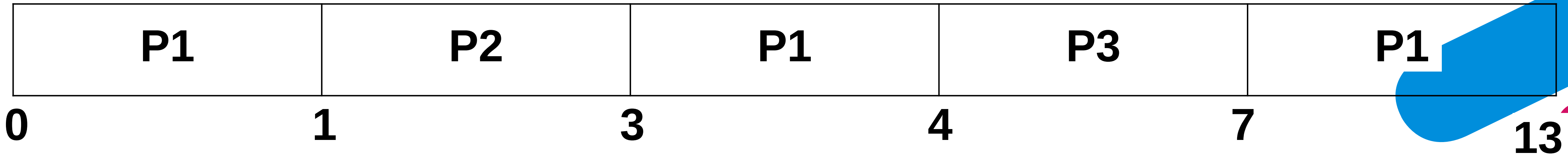
Example:

- Let's understand this with the help of an example.
- Suppose we have the following 3 processes with process ID's P1, P2, and P3 and they arrive into the CPU in the following manner:



Process ID	Arrival Time (milliseconds)	Burst Time (milliseconds)
P1	0	8
P2	1	2
P3	4	3

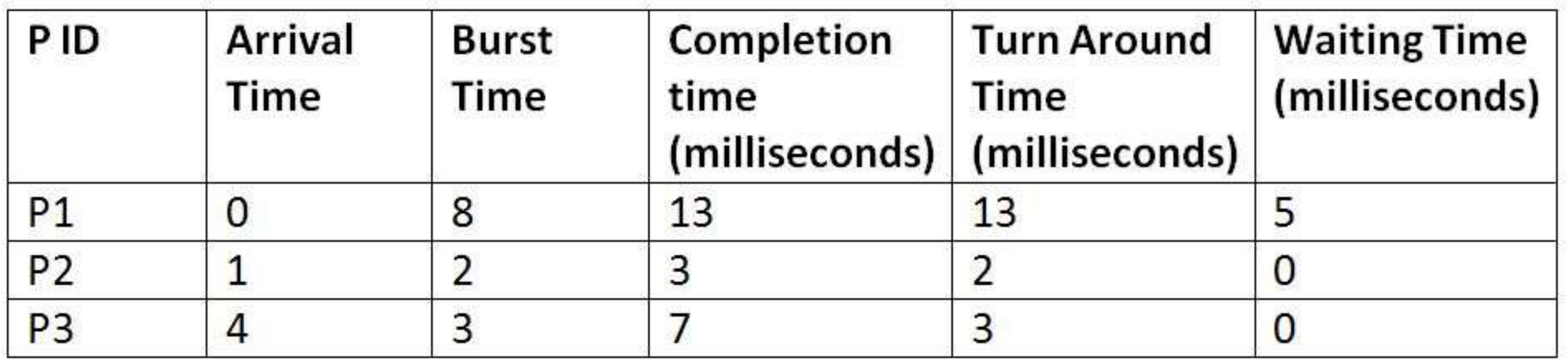
Gantt Chart



Explanation

- At the 0th unit of the CPU, we have only process **P1**, so it gets executed for the 1-time unit.
- At the 1st unit of the CPU, the Process **P2** also arrives. Now, the **P1** needs 7 more units more to be executed, and **P2** needs only 2 units. So, **P2** is executed by preempting **P1**.
- **P2** gets completed at time unit 3, and unit now no new process has arrived. So, after the completion of **P2**, again **P1** is sent for execution.
- Now, **P1** has been executed for one unit only, and we have an arrival of new process **P3** at time unit 4. Now, the **P1** needs 6-time units more and **P3** needs only 3-time units. So, **P3** is executed by preempting **P1**.

- **P1** gets completed at time unit 7, and after that, we have the arrival of no other process. So again, **P1** is sent for execution, and it gets completed at 13th unit.



P ID	Arrival Time	Burst Time	Completion time (milliseconds)	Turn Around Time (milliseconds)	Waiting Time (milliseconds)
P1	0	8	13	13	5
P2	1	2	3	2	0
P3	4	3	7	3	0

Total Turn Around Time = 13 + 2 + 3

= 18 milliseconds

Average Turn Around Time= Total Turn Around Time / Total No. of Processes

= 18 / 3

= 6 milliseconds

Total Waiting Time = 5 + 0 + 0

= 5 milliseconds

Average Waiting Time = Total Waiting Time / Total No. of Processes

= 5 / 3

= 1.67 milliseconds

Round Robin Algorithm

- The Round robin scheduling algorithm is one of the CPU scheduling algorithms in which every process gets a fixed amount of time quantum to execute the process.
- In this algorithm, every process gets executed cyclically.
- This means that processes that have their burst time remaining after the expiration of the time quantum are sent back to the ready state and wait for their next turn to complete the execution until it terminates.
- This processing is done in FIFO order which suggests that processes are executed on a first-come, first-serve basis.

Example

Consider the following 6 processes: P1, P2, P3, P4, P5, and P6 with their arrival time and burst time as given below:

Q. What are the average waiting and turnaround times for the round-robin scheduling algorithm (RR) with a time quantum of 4 units?

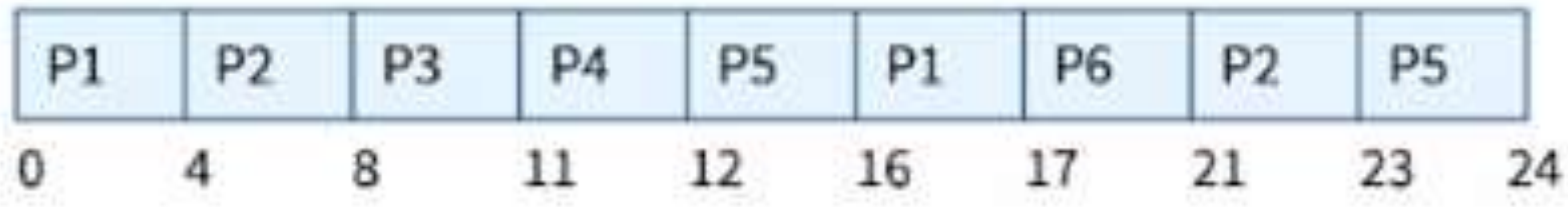
Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	6
P3	2	3
P4	3	1
P5	4	5
P6	6	4

Solution:

- At first, In the ready queue, process P1 will be executed for a time slice of 4 units. Since there are no processes initially, Process P1, with a burst time of 5 units, will be the only process in the ready queue.
- Along with the execution of P1, four more processes, P2, P3, P4, and P5, arrive in the ready queue. P1 will be added to the ready queue due to the remaining 1 unit.
- During the execution of P2, P6 arrived in the ready queue. Since P2 has not been completed, P2 will be added to the ready queue.
- Similarly, P3 and P4 have been completed, but P5 has a remaining burst time of 1 unit. Hence it will be added back to the queue.
- The next processes, P6 and P2, will be executed. Only P5 will be left with 1 unit of burst time.

GANTT Chart:

The Gantt chart will look like this:



Processes	Arrival Time(AT)	Burst Time(BT)	Turn Around Time(TAT)	Waiting Time(WT)
P1	0	5	17	12
P2	1	6	22	16
P3	2	3	9	6
P4	3	1	9	8
P5	4	5	20	15
P6	6	4	15	11

Priority Scheduling Algorithms

- Priority Scheduling is a method of scheduling processes that is based on priority.
- In this algorithm, the scheduler selects the tasks to work as per the priority.
- The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc.

Types of Priority Scheduling

- **Preemptive Scheduling**
- **Non-Preemptive Scheduling**

Priority Scheduling Algorithms

- Priority Scheduling is a method of scheduling processes that is based on priority.
- In this algorithm, the scheduler selects the tasks to work as per the priority.
- The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc.

Types of Priority Scheduling

- **Preemptive Scheduling**
- **Non-Preemptive Scheduling**

Priority Scheduling Algorithms

- **Preemptive Scheduling**

In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

- **Non-Preemptive Scheduling**

In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy, will release the CPU either by switching context or terminating.

Non Preemptive Priority Scheduling

In the Non Preemptive Priority scheduling, The Processes are scheduled according to the priority number assigned to them.

Once the process gets scheduled, it will run till the completion.

Example

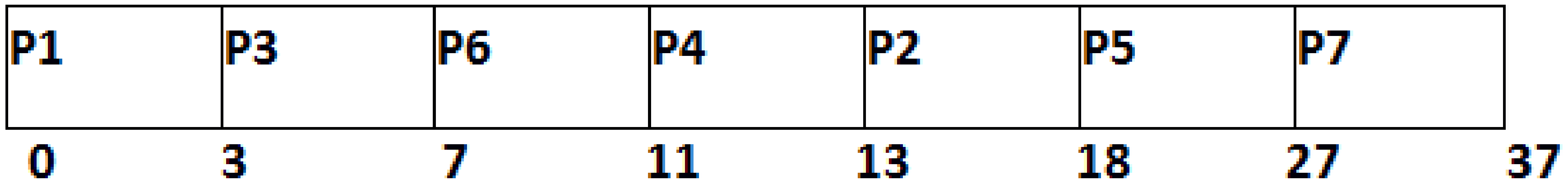
In the Example, there are 7 processes P1, P2, P3, P4, P5, P6 and P7. Their priorities, Arrival Time and burst time are given in the table.

Process ID	Priority	Arrival Time	Burst Time
1	2	0	3
2	6	2	5
3	3	1	4
4	5	4	2
5	7	6	9
6	4	5	4
7	10	7	10

Solution:

We can prepare the Gantt chart according to the Non Preemptive priority scheduling.

The Process P1 arrives at time 0 with the burst time of 3 units and the priority number 2. Since No other process has arrived till now hence the OS will schedule it immediately.



- Meanwhile the execution of P1, two more Processes P2 and P3 are arrived. Since the priority of P3 is 3 hence the CPU will execute P3 over P2.
- Meanwhile the execution of P3, All the processes get available in the ready queue. The Process with the lowest priority number will be given the priority. Since P6 has priority number assigned as 4 hence it will be executed just after P3.
- After P6, P4 has the least priority number among the available processes; it will get executed for the whole burst time.
- Since all the jobs are available in the ready queue hence All the Jobs will get executed according to their priorities. If two jobs have similar priority number assigned to them, the one with the least arrival time will be executed.

Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
1	2	0	3	3	3	0	0
2	6	2	5	18	16	11	13
3	3	1	4	7	6	2	3
4	5	4	2	13	9	7	11
5	7	6	9	27	21	12	18
6	4	5	4	11	6	2	7
7	10	7	10	37	30	18	27

Preemptive Priority Scheduling

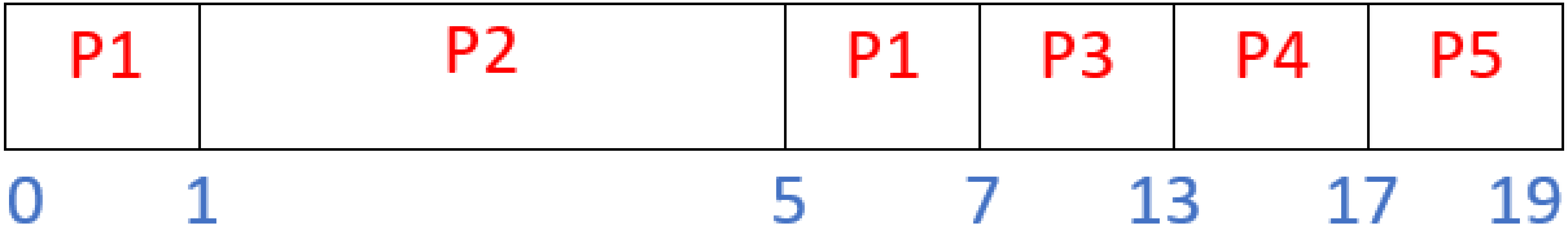
- In Preemptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The One with the highest priority among all the available processes will be given the CPU next.
- The difference between preemptive priority scheduling and non preemptive priority scheduling is that, in the preemptive priority scheduling, the job which is being executed can be stopped at the arrival of a higher priority job.

Example-1: Consider the following table of arrival time, Priority, and burst time for five processes P1, P2, P3, P4, and P5.

Process	Arrival Time	Priority	Burst Time
P1	0 ms	3	3 ms
P2	1 ms	2	4 ms
P3	2 ms	4	6 ms
P4	3 ms	6	4 ms
P5	5 ms	10	2 ms

Solution:

Gantt Chart



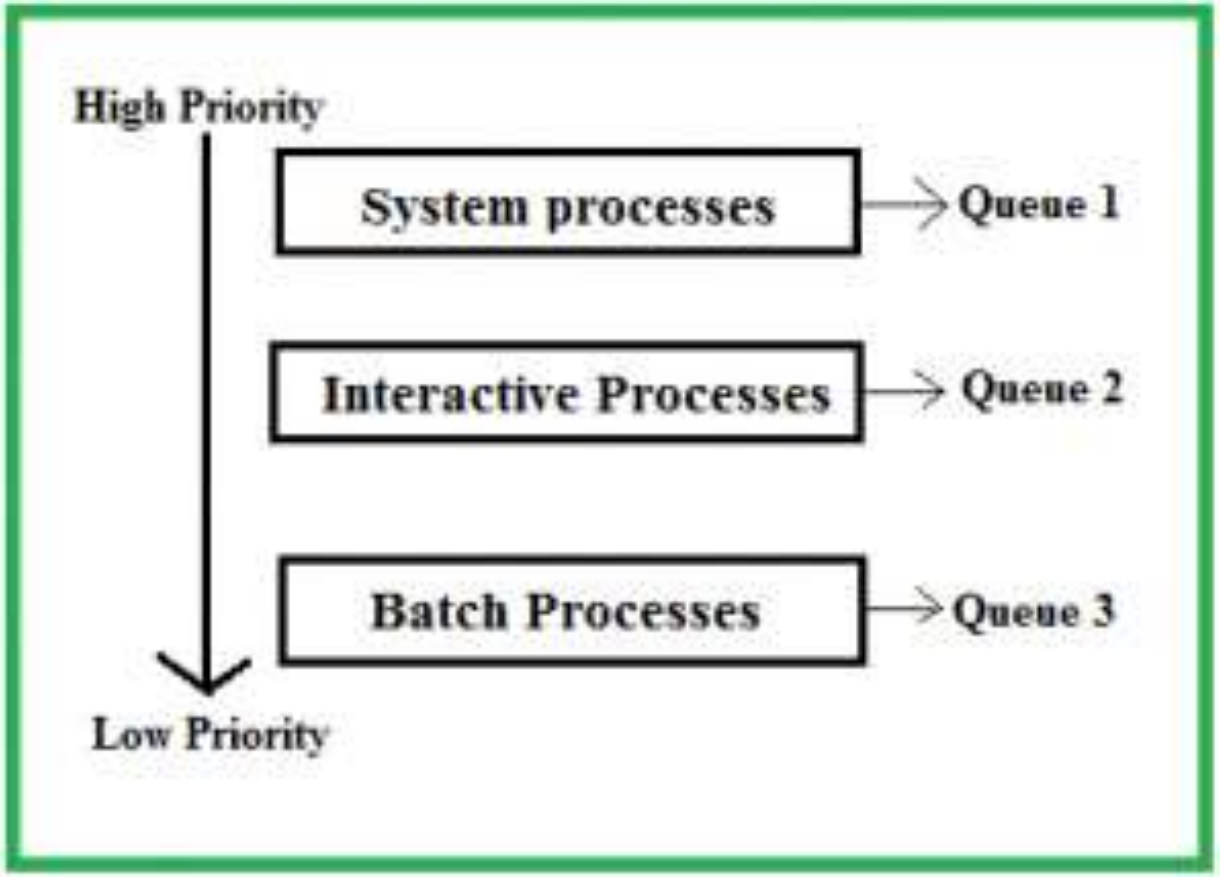
Process	A.T	Priority	B.T	C.T	T.A.T	W.T	R.T
P1	0	3	3	7	7	4	0
P2	1	2(H)	4	5	4	0	0
P3	2	4	6	13	11	5	5
P4	3	6	4	17	14	10	10
P5	5	10(L)	2	19	14	12	12

Multilevel Queue Scheduling Algorithm(MLQ)

- Multilevel queue scheduling is a type of process scheduling algorithm used in operating systems to manage and prioritize the execution of processes.
- In a multilevel queue scheduling system, the ready queue is divided into multiple priority levels, and each priority level has its own queue.
- Each queue can have its own scheduling algorithm and time quantum, allowing processes to be classified and managed

Multilevel Queue Scheduling Algorithm(MLQ)

- **Ready Queue** is divided into separate queues for each class of processes. For example, let us take three different types of processes System processes, Interactive processes, and Batch Processes. All three processes have their own queue. Now, look at the below figure.



Key characteristics of multilevel queue scheduling

- **Multiple Priority Levels:** Processes are classified into different priority levels, often based on their importance or resource requirements. Higher-priority processes are typically executed before lower-priority processes.
- **Separate Queues:** Each priority level has its own queue, and each queue may use a different scheduling algorithm. For example, a high-priority queue might use a round-robin scheduling algorithm with a short time quantum, while a low-priority queue might use a first-come, first-served (FCFS) algorithm.
- **Aging:** To prevent starvation of lower-priority processes, some multilevel queue scheduling systems implement aging. Aging means that processes in lower-priority queues gradually move up to higher-priority queues if they spend too much time waiting for execution.

Key characteristics of multilevel queue scheduling

- **Preemption:** In some multilevel queue systems, preemption is allowed, meaning that a higher-priority process can interrupt the execution of a lower-priority process to run. This ensures that important tasks get executed promptly.
- **Queue Assignment:** Processes are initially assigned to a specific queue based on their attributes, such as priority, process type, or resource requirements. As a process's priority changes or its behavior evolves, it may be moved to a different queue.
- **Scheduling within Queues:** Each queue may employ a different scheduling algorithm, such as round-robin, priority-based, or shortest job first (SJF), depending on the requirements of the processes within that queue.

Advantages of Multilevel Queue CPU Scheduling:

- **Low scheduling overhead:** Since processes are permanently assigned to their respective queues, the overhead of scheduling is low, as the scheduler only needs to select the appropriate queue for execution.
- **Efficient allocation of CPU time:** The scheduling algorithm ensures that processes with higher priority levels are executed in a timely manner, while still allowing lower priority processes to execute when the CPU is idle. This ensures optimal utilization of CPU time.
- **Fairness:** The scheduling algorithm provides a fair allocation of CPU time to different types of processes, based on their priority and requirements.

Advantages of Multilevel Queue CPU Scheduling:

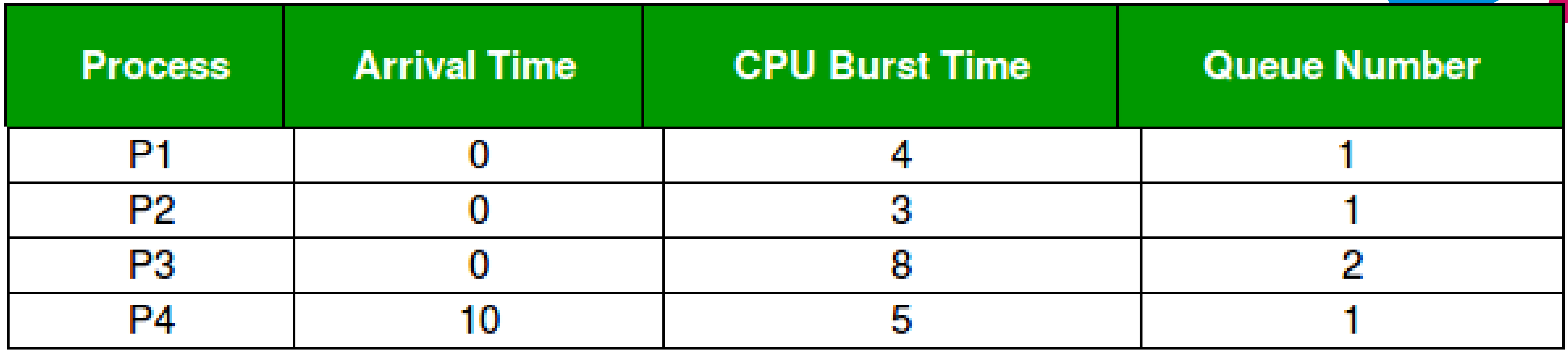
- **Customizable:** The scheduling algorithm can be customized to meet the specific requirements of different types of processes. Different scheduling algorithms can be used for each queue, depending on the requirements of the processes in that queue.
- **Prioritization:** Priorities are assigned to processes based on their type, characteristics, and importance, which ensures that important processes are executed in a timely manner.
- **Preemption:** Preemption is allowed in Multilevel Queue Scheduling, which means that higher-priority processes can preempt lower-priority processes, and the CPU is allocated to the higher-priority process. This helps ensure that high-priority processes are executed in a timely manner.

Disadvantages of Multilevel Queue CPU Scheduling

- Some processes may starve for CPU if some higher priority queues are never becoming empty.
- It is inflexible in nature.
- There may be added complexity in implementing and maintaining multiple queues and scheduling algorithms.

Example :

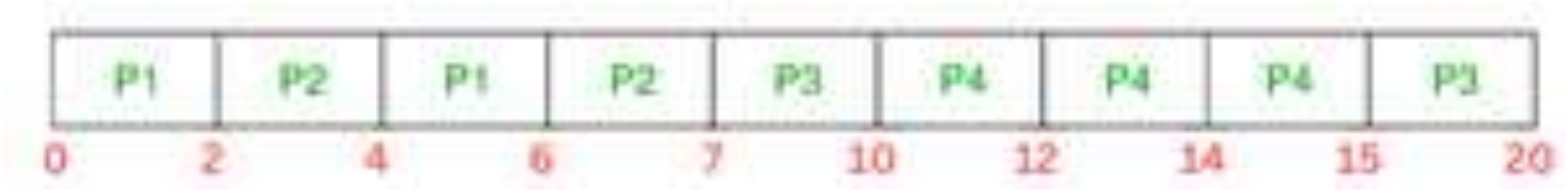
- Consider the below table of four processes under Multilevel queue scheduling. Queue number denotes the queue of the process.



Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

Priority of queue 1 is greater than queue 2. queue 1 uses Round Robin (Time Quantum = 2) and queue 2 uses FCFS.

Solution:



Working:

- At starting, both queues have process so process in queue 1 (P1, P2) runs first (because of higher priority) in the round-robin fashion and completes after 7 units
- Then process in queue 2 (P3) starts running (as there is no process in queue 1) but while it is running P4 comes in queue 1 and interrupts P3 and start running for 5 seconds and
- After its completion P3 takes the CPU and completes its execution.

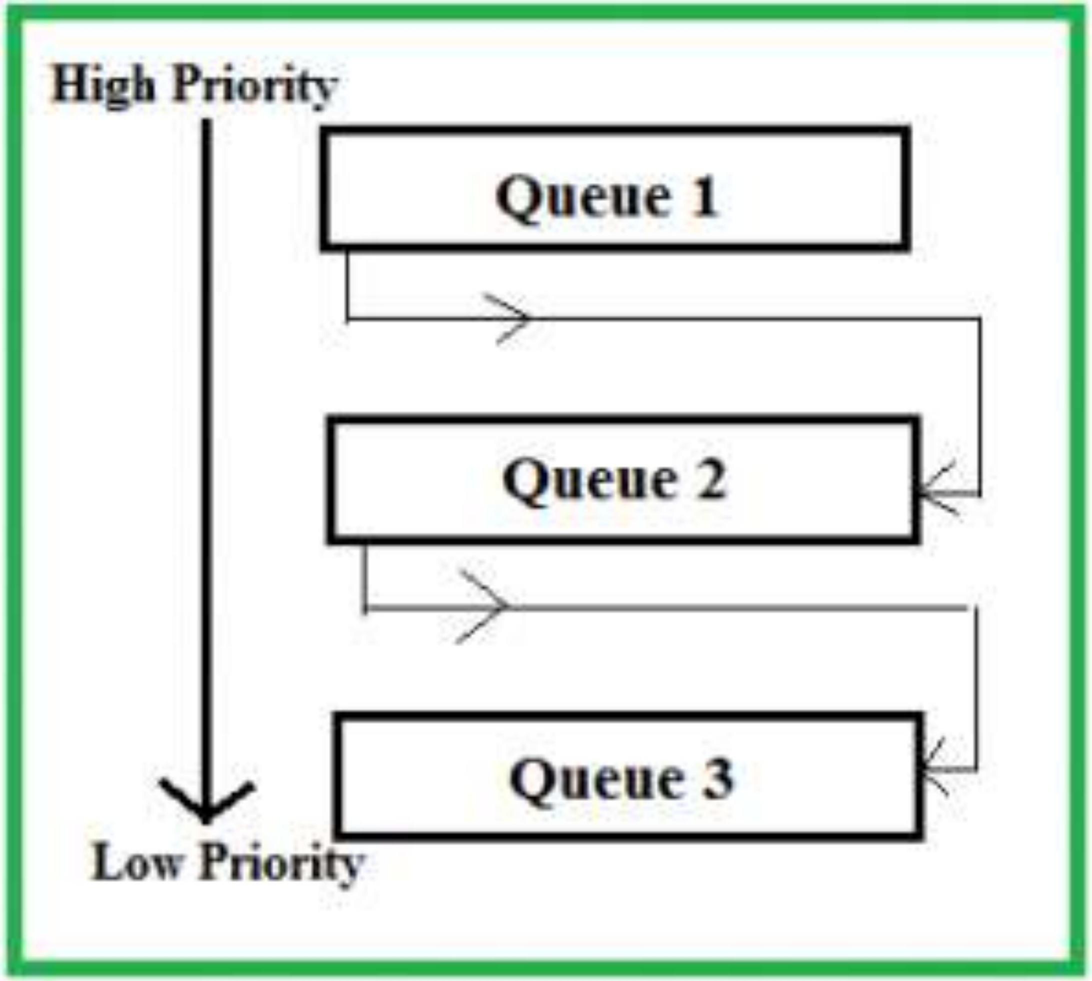
Multilevel Feedback Queue Scheduling (MLFQ)

MLFQ Scheduling is like Multilevel Queue(MLQ) Scheduling but in this process can move between the queues. And thus, much more efficient than multilevel queue scheduling.

Characteristics of Multilevel Feedback Queue Scheduling:

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system, and processes are allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,

- Multilevel feedback queue scheduling, however, allows a process to move between queues. Multilevel Feedback Queue Scheduling (MLFQ) keeps analyzing the behavior (time of execution) of processes and according to which it changes its priority.



Features of Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling:

Multiple queues: Similar to MLQ scheduling, MLFQ scheduling divides processes into multiple queues based on their priority levels. However, unlike MLQ scheduling, processes can move between queues based on their behavior and needs.

Priorities adjusted dynamically: The priority of a process can be adjusted dynamically based on its behavior, such as how much CPU time it has used or how often it has been blocked. Higher-priority processes are given more CPU time and lower-priority processes are given less.

Time-slicing: Each queue is assigned a time quantum or time slice, which determines how much CPU time a process in that queue is allowed to use before it is preempted and moved to a lower priority queue.

Features of Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling:

Feedback mechanism: MLFQ scheduling uses a feedback mechanism to adjust the priority of a process based on its behavior over time. For example, if a process in a lower-priority queue uses up its time slice, it may be moved to a higher-priority queue to ensure it gets more CPU time.

Preemption: Preemption is allowed in MLFQ scheduling, meaning that a higher-priority process can preempt a lower-priority process to ensure it gets the

Multilevel feedback queue scheduling, however, allows a process to move between queues, it also keeps analyzing the behavior (time of execution) of processes and according to which it changes its priority.

Advantages of Multilevel Feedback Queue Scheduling:

- It is more flexible.
- It allows different processes to move between different queues.
- **It prevents starvation by moving a process that waits too long for the lower priority queue to the higher priority queue.**

Disadvantages of Multilevel Feedback Queue Scheduling:

- The selection of the best scheduler, it requires some other means to select the values.
- It produces more CPU overheads.
- **It is the most complex algorithm.**

Example: Consider a system that has a CPU-bound process, which requires a burst time of 40 seconds. The multilevel Feed Back Queue scheduling algorithm is used and the queue time quantum ‘2’ seconds and in each level it is incremented by ‘5’ seconds. Then how many times the process will be interrupted and in which queue the process will terminate the execution?

Solution:

- Process P needs 40 Seconds for total execution.
- At Queue 1 it is executed for 2 seconds and then interrupted and shifted to queue 2.
- At Queue 2 it is executed for 7 seconds and then interrupted and shifted to queue 3.
- At Queue 3 it is executed for 12 seconds and then interrupted and shifted to queue 4.
- At Queue 4 it is executed for 17 seconds and then interrupted and shifted to queue 5.
- At Queue 5 it executes for 2 seconds and then it completes.
- Hence the process is interrupted 4 times and completed on queue 5.