Module II

Process Management: Process Concept The Processes, Process States, PCB, Process Scheduling Scheduling Queues, Schedulers,Context Switch, Operations on Process, Inter-Process CommunicationShared-Memory System, Message Passing System.

CPU Scheduling: Basics, CPU-I/O Burst Cycle, CPU Scheduler Preemptive Scheduling, Dispatcher, Scheduling Criteria, Scheduling Algorithms FCFS, SJF, Round-Robin, Priority

==========

Process Management

Process

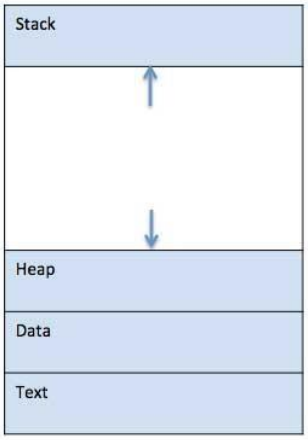
Definition:

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.
- Attributes held by process include hardware state, memory, CPU etc.

Process memory is divided into four sections for efficient working :

- The ***Text section*** is made up of the compiled program code, read in from non-volatile storage when the program is launched.
- The ***Data section*** is made up the global and static variables, allocated and initialized prior to executing the main.
- The ***Heap*** is used for the dynamic memory allocation, and is managed via calls to new, delete, malloc, free, etc.
- The ***Stack*** is used for local variables. Space on the stack is reserved for local variables when they are declared.

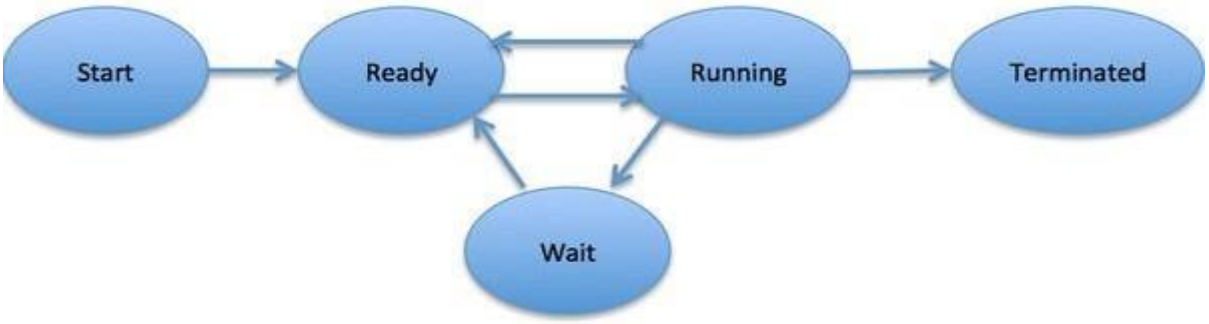


Process States:

When a process executes, it passes through different states. These stages may differ in different operating systems.

In general, a process can have one of the following five states at a time.

S.No	State & Description
1	**Start:** This is the initial state when a process is first started/created.
2	**Ready:** The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that theycan run. Process may come into this state after **Start** state or while running it by butinterrupted by the scheduler to assign CPU to some other process.
3	**Running:** Once the process has been assigned to a processor by the OS scheduler, theprocess state is set to running and the processor executes its instructions.
4	**Waiting:** Process moves into the waiting state if it needs to wait for a resource, such aswaiting for user input, or waiting for a file to become available.
5	**Terminated or Exit:** Once the process finishes its execution, or it is terminated by theoperating system, it is moved to the terminated state where it waits to be removed frommain memory.



Process State Diagram

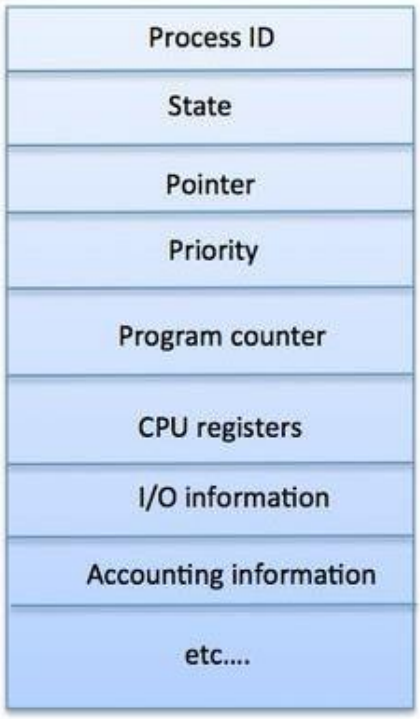
Process Control Block (PCB):

- A Process Control Block is a data structure maintained by the Operating System for every process.
- The PCB is identified by an integer process ID (PID).
- A PCB keeps all the information needed to keep track of a process as listed below in the table –

S.No.	Information & Description
1	**Process State:** The current state of the process i.e., whether it is ready, running,waiting, or whatever.
2	**Process privileges:** This is required to allow/disallow access to system resources.
3	**Process ID:** Unique identification for each of the process in the operating system.
4	**Pointer:** A pointer to parent process.
5	**Program Counter: P**rogram Counter is a pointer to the address of the next instructionto be executed for this process.
6	**CPU registers:** Various CPU registers where process need to be stored for executionfor running state.

7	**CPU Scheduling Information:** Process priority and other scheduling information which is required to schedule the process.
8	**Memory management information:** This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
9	**Accounting information:** This includes the amount of CPU used for process execution, time limits, execution ID etc.
10	**IO status information:** This includes a list of I/O devices allocated to the process.

The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems. Here is a simplified diagram of a PCB −



Process Control Block (PCB) Diagram

The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

Process Scheduling

Definition

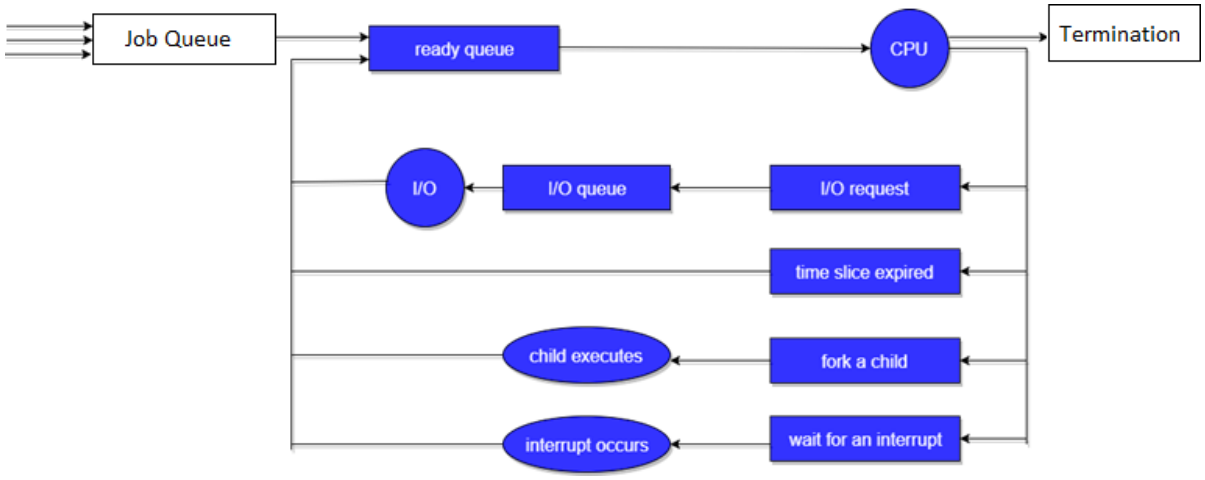
- The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
- Process scheduling is an essential part of a Multiprogramming operating systems.
- Such operating 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.

What are Scheduling Queues?

- All processes, upon entering into the system, are stored in the **Job Queue**.
- Processes in the Ready state are placed in the **Ready Queue**.
- Processes waiting for a device to become available are placed in **Device Queues**. There are unique device queues available for each I/O device.

A new process is initially put in the **Ready queue**. It waits in the ready queue until it is selected for execution (or dispatched). Once the process is assigned to the CPU and is executing, one of the following several events can occur:

- The process could issue an I/O request, and then be placed in the **I/O queue**.
- The process could create a new sub-process and wait for its termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.



In the first two cases, the process eventually switches from the waiting state to the ready state, and is then put back in the ready queue. A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.

Schedulers:

- Schedulers are special system software which handle process scheduling in various ways.
- Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −
  - Long-Term Scheduler
  - Short-Term Scheduler
  - Medium-Term Scheduler

Long Term Scheduler

- It is also called a job scheduler.
- A long-term scheduler determines which programs are admitted to the system for processing.
- It selects processes from the queue and loads them into memory for execution.
- Process loads into the memory for CPU scheduling.
- The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound.

- It also controls the degree of multiprogramming.
- If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.
- On some systems, the long-term scheduler may not be available or minimal.
- Time-sharing operating systems have no long term scheduler.
- When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler:

- It is also called as CPU scheduler.
- Its main objective is to increase system performance in accordance with the chosen set of criteria.
- It is the change of ready state to running state of the process.
- CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.
- Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

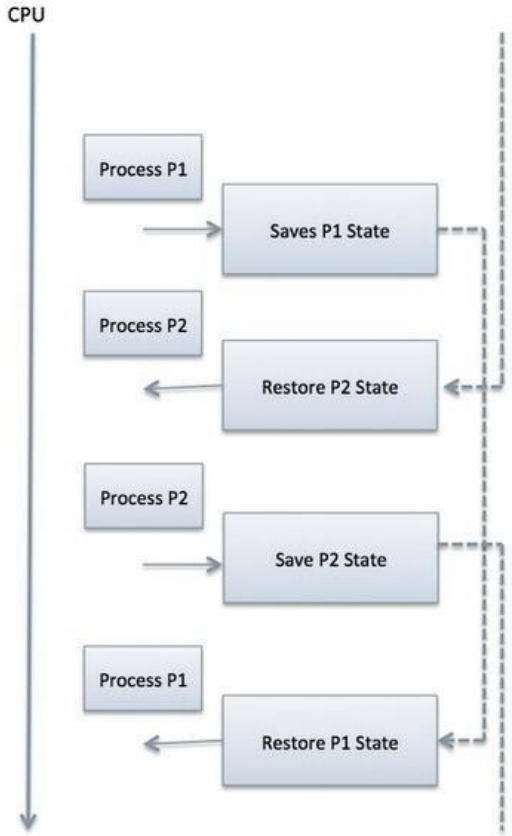
- Medium-term scheduling is a part of swapping.
- It removes the processes from the memory.
- It reduces the degree of multiprogramming.
- The medium-term scheduler is in-charge of handling the swapped out-processes.
- A running process may become suspended if it makes an I/O request.
- A suspended process cannot make any progress towards completion.
- In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage.
- This process is called swapping, and the process is said to be swapped out or rolled out.
- Swapping may be necessary to improve the process mix.

Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch:

- A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time.
- Using this technique, a context switcher enables multiple processes to share a single CPU.
- Context switching is an essential part of a multitasking operating system features.
- When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block.
- After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc.
- At that point, the second process can start executing.



- Context switches are computationally intensive since register and memory state must be saved and restored.
- To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers.
- When the process is switched, the following information is stored for later use.
  - Program Counter
  - Scheduling information
  - Base and limit register value
  - Currently used register
  - Changed State
  - I/O State information
  - Accounting information

Inter-process Communication

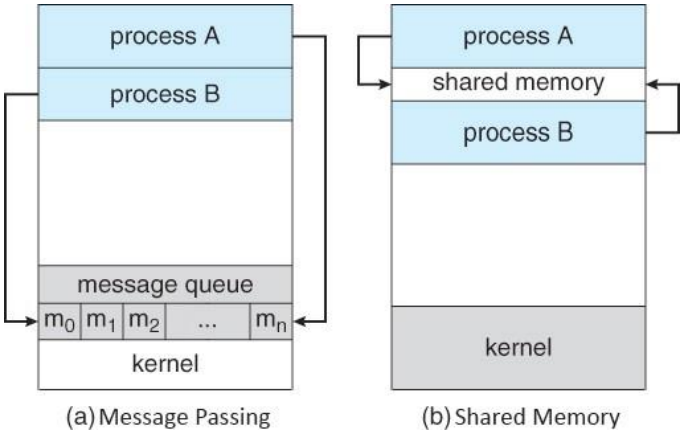
- Processes executing concurrently in the operating system might be either independent processes or cooperating processes.
- A process is independent if it cannot be affected by the other processes executing in the system**.**
- ***Inter Process Communication (IPC)*** is a mechanism that involves communication of one process with another process. This usually occurs only in one system.
- Communication can be of two types −
  - Between related processes initiating from only one process, such as parent and child processes.
  - Between unrelated processes, or two or more different processes.
- Processes can communicate with each other using these two ways:
  - Shared Memory
  - Message passing

Shared Memory

- Shared memory is the memory that can be simultaneously accessed by multiple processes.
- This is done so that the processes can communicate with each other.
- All POSIX systems, as well as Windows operating systems use shared memory.

Message Queue

- Multiple processes can read and write data to the message queue without being connected to each other.
- Messages are stored in the queue until their recipient retrieves them.
- Message queues are quite useful for inter-process communication and are used by most operating systems.
- If two processes p1 and p2 want to communicate with each other, they proceed as follow:
  - Establish a communication link (if a link already exists, no need to establish it again.)
  - Start exchanging messages using basic primitives.
  - We need at least two primitives:
    send(message, destinaion) or send(message)
    receive(message, host) or receive(message)



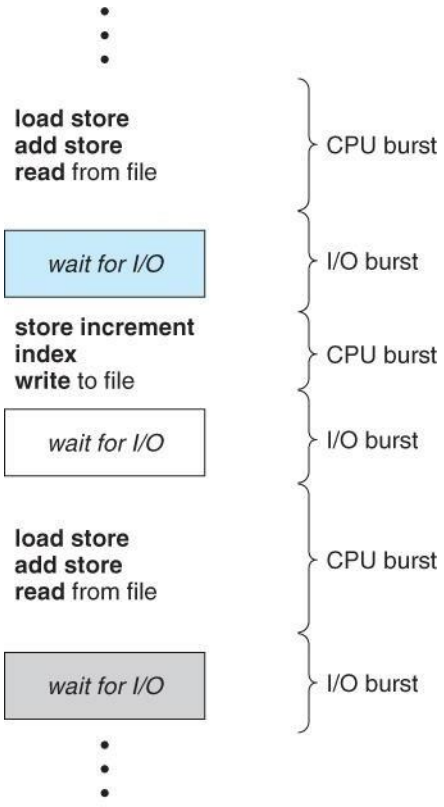
CPU Scheduling

Scheduling

- Maximize throughput of the system
- Maximize number of users receiving acceptable response times.
- Be predictable
- Balance resource use
- Avoid indefinite postponement
- Enforce Priorities
- Give preference to processes holding key resources
- Give better service to processes that have desirable behaviour patterns

CPU and I/O Burst Cycle:

- Process execution consists of a cycle of CPU execution and I/O wait.
- Processes alternate between these two states.
- Process execution begins with a CPU burst, followed by an I/O burst, then another CPU burst ... etc
- The last CPU burst will end with a system request to terminate execution rather than with another I/O burst.
- The duration of these CPU burst have been measured.
- An I/O-bound program would typically have many short CPU bursts, A CPU-bound program might have a few very long CPU bursts.
- This can help to select an appropriate CPU-scheduling algorithm.



Preemptive Scheduling:

- Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state.
- The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining.
- That process stays in ready queue till it gets next chance to execute.

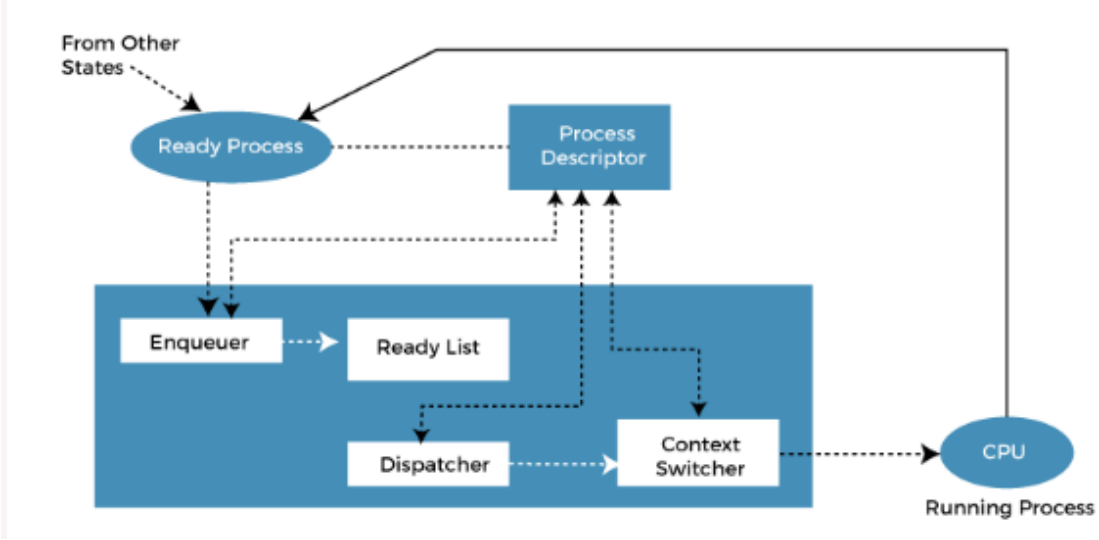
Non-Preemptive Scheduling:

- Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state.
- In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state.
- In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution.
- Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.

Basis for Comparison	**Preemptive Scheduling**	**Non Preemptive Scheduling**
Basic	The resources are allocated to a process for a limited time.	Once resources are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Interrupt	Process can be interrupted in between.	Process can not be interrupted till it terminates or switches to waiting state.
Starvation	If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
Overhead	Preemptive scheduling has overheads of scheduling the processes.	Non-preemptive scheduling does not have overheads.
Flexibility	Preemptive scheduling is flexible.	Non-preemptive scheduling is rigid.
Cost	Preemptive scheduling is cost associated.	Non-preemptive scheduling is not cost associative.

Dispatcher:

A dispatcher is a special program that comes into play after the scheduler. When the short term scheduler selects from the ready queue, the Dispatcher performs the task of allocating the selected process to the CPU. A running process goes to the waiting state for IO operation etc., and then the CPU is allocated to some other process. This switching of CPU from one process to the other is called **context switching**.



A dispatcher performs various tasks, including context switching, setting up user registers and memory mapping. These are necessary for the process to execute and transfer CPU control to that process. When dispatching, the process changes from the ready state to the running state.

The Dispatcher needs to be as fast as possible, as it is run on every context switch. The time consumed by the Dispatcher is known as *dispatch latency*.

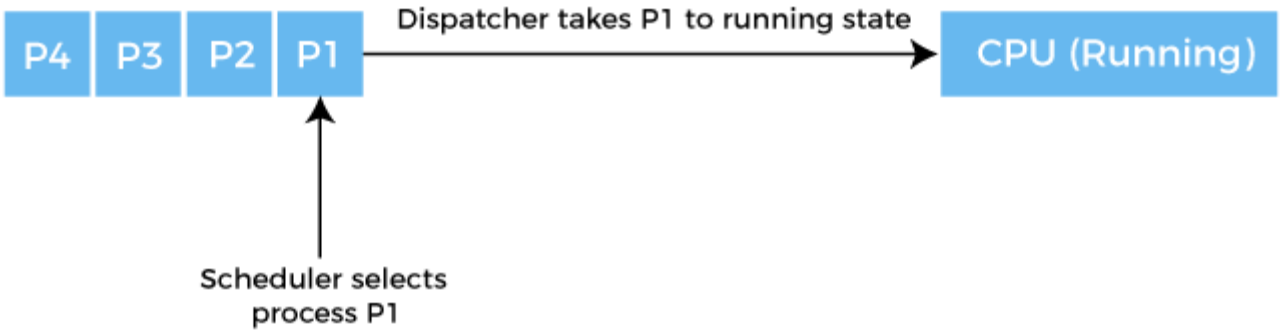
Sometimes, the Dispatcher is considered part of the short-term scheduler, so the whole unit is called the *short-term scheduler*. In this scenario, the task of the short term scheduler is to select a process from the ready queue and allocate the CPU for that process. In the operating system, a dispatcher has the following responsibilities:

- **Switching to user mode:** All of the low-level operating system processes run on the kernel level security access, but all application code and user issued processes run in the application space or the user permission mode. The Dispatcher switches the processes to the user mode.
- **Addressing:** The program counter (PC) register points towards the next process to be executed. The Dispatcher is responsible for addressing that address.
- **Initiation of context switch:** A context switch is when a currently running process is halted, and all of its data and its process control block (PCB) are stored in the main memory, and another process is loaded in its place for execution.
- **Managing dispatch latency:** Dispatch latency is calculated as the time it takes to stop one process and start another. The lower the dispatch latency, the more efficient the software for the same hardware configuration.

NOTE: A dispatcher is NOT a thread. The Dispatcher runs on each core, runs a thread for a while, saves its state, loads the state of another thread and runs it.

Example of Scheduler and Dispatcher

Suppose four processes, P1, P2, P3, and P4, are in the ready queue. Their arrival times are T1, T2, T3, and T4, respectively. And a First in, First out (FIFO) scheduling algorithm is used in this whole process or task.



The process P1 arrived first, so the scheduler will decide it is the first process to be executed, and the Dispatcher will remove P1 from the ready queue and give it to the CPU.

Then the scheduler will determine process P2 to be the next process that should be executed, so when the Dispatcher returns to the queue for a new process, it will take process P2 and give it to the CPU. This continues in the same way for process P3 and then P4.

Difference between Scheduler and Dispatcher in Operating System

The CPU cannot execute all processes residing in the ready queue and waiting for execution simultaneously. So the operating system has to choose a particular process based on the scheduling algorithm, and the scheduler does this procedure of selecting a process among various processes.

Once the scheduler has selected a process from the queue, the Dispatcher takes it from the ready queue and moves it into the running state. Therefore, the scheduler gives the Dispatcher an ordered list of processes which the Dispatcher moves to the CPU over time.

Scheduler and Dispatcher are used in the process scheduling of an operating system, and they both complete the same process or task. Still, the difference between scheduler and Dispatcher is that the scheduler selects a process out of several processes to be executed. In contrast, the Dispatcher allocates the CPU for the selected process by the scheduler. There are some more differences between the scheduler and the Dispatcher in the operating system, such as:

Properties	**Scheduler**	**Dispatcher**
Definition	The scheduler is special system software that handles process scheduling by selecting the process to execute.	The Dispatcher is a module that controls the CPU to the process selected by the short term scheduler.

Algorithm	The scheduler works on various algorithms such as FCFS, SJF, RR etc.	The Dispatcher has no specific algorithm for its implementation.
Types	There are three types of schedulers, such as Long-term, Short-term, Medium-term.	There are no different types in Dispatcher, and it is just a code segment.
Dependency	The scheduler works independently, and it works immediately when needed.	The working of a Dispatcher depends on a scheduler, which means Dispatcher has to wait until the scheduler selects a process.
Time Taken	Time taken by the scheduler is usually negligible.	The time taken by the Dispatcher is called dispatch latency.
Functions	The only work of the scheduler is a selection of processes.	A dispatcher is also responsible for Context Switching, Switch to user mode, and Jumping to the proper location when the process restarted again.
Tasks	The scheduler performs the task in three stages, such as: o The **long-term scheduler**selects the process from the job queue and brings it to the ready queue. o The **short term scheduler**selects a process in the ready queue. o The **medium scheduler**carries out the swap in, swap out of the process.	The Dispatcher allocates the CPU to the process selected by the short-term scheduler.

Scheduling Criteria

- There are several different criteria to consider when trying to select the "best" scheduling algorithm for a particular situation and environment, including:
  - **CPU utilization** - Ideally the CPU would be busy 100% of the time, so as to waste 0 CPU cycles. On a real system CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.)
  - **Throughput** - Number of processes completed per unit time. May range from 10 / second to 1 / hour depending on the specific processes.
  - **Turnaround time** - Time required for a particular process to complete, from submission time to completion.
  - **Waiting time** - How much time processes spend in the ready queue waiting their turn to get on the CPU.
  - **Response time** - The time taken in an interactive program from the issuance of a command to the ***commence*** of a response to that command.

In brief:
Arrival Time: Time at which the process arrives in the ready queue.
Completion Time: Time at which process completes its execution.
Burst Time: Time required by a process for CPU execution.
Turn Around Time: Time Difference between completion time and arrival time.
Turn Around Time = Completion Time – Arrival Time
Waiting Time(W.T): Time Difference between turnaround time and burst time.
Waiting Time = Turn Around Time – Burst Time

Types of Scheduling Algorithm

(a) First Come First Serve (FCFS)

In FCFS Scheduling

- The process which arrives first in the ready queue is firstly assigned the CPU.
- In case of a tie, process with smaller process id is executed first.
- It is always non-preemptive in nature.
- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Advantages-

- It is simple and easy to understand.
- It can be easily implemented using queue data structure.
- It does not lead to starvation.

Disadvantages-

- It does not consider the priority or burst time of the processes.
- It suffers from convoy effect i.e. processes with higher burst time arrived before the processes with smaller burst time.

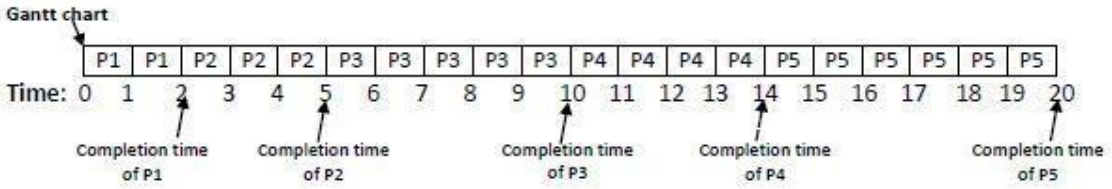
Example 1:

Q. Consider the following processes with burst time (CPU Execution time). Calculate the average waiting time and average turnaround time?

Process id	Arrival time	Burst time/CPU execution time
P1	0	2
P2	1	3
P3	2	5
P4	3	4
P5	4	6

Sol.

Gantt chart



Turnaround time= Completion time – Arrival time

Waiting time= Turnaround time – Burst time

Process id	Arrival time	Burst time	Completion time	Turnaround time	Waiting time
P1	0	2	2	2-0=2	2-2=0
P2	1	3	5	5-1=4	4-3=1
P3	2	5	10	10-2=8	8-5=3
P4	3	4	14	14-3=11	11-4=7
P5	4	6	20	20-4=16	16-6=10

Average turnaround time= $\sum_{i=0}^{n}$ Turnaround time(i)/n where, n= no. of process

Average waiting time= $\sum_{i=0}^{n}$ Wating time(i)/n where, n= no. of process

Average turnaround time= 2+4+8+11+16/5 =41/5 =8.2

Average waiting time= 0+1+3+7+10/5 = 21/5 =4.2

Example 2:

Consider the processes P1, P2, P3 given in the below table, arrives for execution inthe same order, with Arrival Time 0, and given Burst Time,

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	24
P2	0	3
P3	0	3

Gantt chart

P_1	P_2	P_3
0 24	27	30

PROCESS	WAIT TIME	TURN AROUND TIME
P1	0	24
P2	24	27
P3	27	30

Total Wait Time = 0 + 24 + 27 = 51 ms

Average Waiting Time = (Total Wait Time) / (Total number of processes) = 51/3 =

17 msTotal Turn Around Time: 24 + 27 + 30 = 81 ms

Average Turn Around time = (Total Turn Around Time) / (Total number of processes) = 81 / 3 = 27 ms

Throughput = 3 jobs/30 sec = 0.1 jobs/sec

Example 3:

Consider the processes P1, P2, P3, P4 given in the below table, arrives for executionin the same order, with given Arrival Time and Burst Time.

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Gantt chart

P_1	P_2	P_3	P_4

0 8 12 21 26

PROCESS	WAIT TIME	TURN AROUND TIME
P1	0	8 – 0 = 8
P2	8 – 1 = 7	12 – 1 = 11
P3	12 – 2 = 10	21 – 2 = 19
P4	21 – 3 = 18	26 – 3 = 23

Total Wait Time:= 0 + 7 + 10 + 18 = 35 ms

Average Waiting Time = (Total Wait Time) / (Total number of processes)= 35/4 =

8.75 msTotal Turn Around Time: 8 + 11 + 19 + 23 = 61 ms

Average Turn Around time = (Total Turn Around Time) / (Total number of processes)61/4 = 15.25 ms

Throughput: 4 jobs/26 sec = 0.15385 jobs/sec

(b) Shortest Job First (SJF)

- Process which have the shortest burst time are scheduled first.
- If two processes have the same bust time, then FCFS is used to break the tie.
- This is a non-pre-emptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.
- Pre-emptive mode of Shortest Job First is called as Shortest Remaining Time First (SRTF).

Advantages-

- SRTF is optimal and guarantees the minimum average waiting time.
- It provides a standard for other algorithms since no other algorithm performs better than it.

Disadvantages-

- It can not be implemented practically since burst time of the processes can not be known in advance.
- It leads to starvation for processes with larger burst time.
- Priorities can not be set for the processes.
- Processes with larger burst time have poor response time.

Example-01:

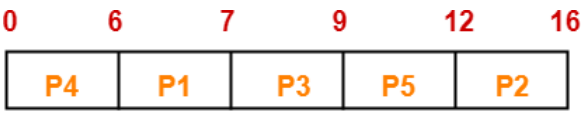
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

Solution-

If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turnaround time.

Gantt Chart-



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 1 = 3
P2	16	16 – 1 = 15	15 – 4 = 11
P3	9	9 – 4 = 5	5 – 2 = 3
P4	6	6 – 0 = 6	6 – 6 = 0
P5	12	12 – 2 = 10	10 – 3 = 7

Now,

- Average Turn Around time = (4 + 15 + 5 + 6 + 10) / 5 = 40 / 5 = 8 unit
- Average waiting time = (3 + 11 + 3 + 0 + 7) / 5 = 24 / 5 = 4.8 unit

Example-02:

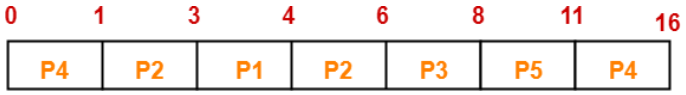
Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

If the CPU scheduling policy is SJF pre-emptive, calculate the average waiting time and average turnaround time.

Solution-

Gantt Chart-



Gantt Chart

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 3 = 1	1 – 1 = 0
P2	6	6 – 1 = 5	5 – 4 = 1
P3	8	8 – 4 = 4	4 – 2 = 2
P4	16	16 – 0 = 16	16 – 6 = 10
P5	11	11 – 2 = 9	9 – 3 = 6

Now,

- Average Turn Around time = (1 + 5 + 4 + 16 + 9) / 5 = 35 / 5 = 7 unit
- Average waiting time = (0 + 1 + 2 + 10 + 6) / 5 = 19 / 5 = 3.8 unit

Example-03:

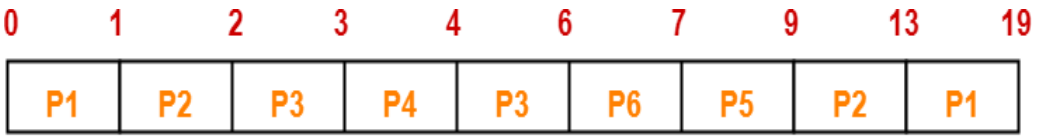
Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	7
P2	1	5
P3	2	3
P4	3	1
P5	4	2
P6	5	1

If the CPU scheduling policy is shortest remaining time first, calculate the average waiting time and average turnaround time.

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	19	19 – 0 = 19	19 – 7 = 12
P2	13	13 – 1 = 12	12 – 5 = 7
P3	6	6 – 2 = 4	4 – 3 = 1
P4	4	4 – 3 = 1	1 – 1 = 0
P5	9	9 – 4 = 5	5 – 2 = 3
P6	7	7 – 5 = 2	2 – 1 = 1

Now,

- Average Turn Around time = (19 + 12 + 4 + 1 + 5 + 2) / 6 = 43 / 6 = 7.17 unit
- Average waiting time = (12 + 7 + 1 + 0 + 3 + 1) / 6 = 24 / 6 = 4 unit

Example -04:

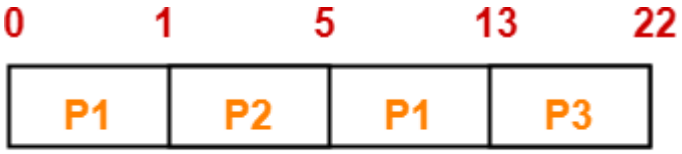
Consider the set of 3 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	9
P2	1	4
P3	2	9

If the CPU scheduling policy is SRTF, calculate the average waiting time and average turn around time.

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	13	13 – 0 = 13	13 – 9 = 4
P2	5	5 – 1 = 4	4 – 4 = 0
P3	22	22- 2 = 20	20 – 9 = 11

Now,

- Average Turn Around time = (13 + 4 + 20) / 3 = 37 / 3 = 12.33 unit
- Average waiting time = (4 + 0 + 11) / 3 = 15 / 3 = 5 unit

Example-05:

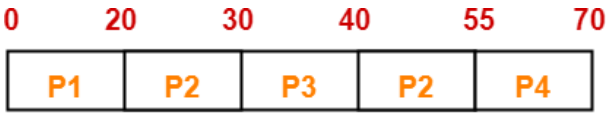
Consider the set of 4 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	20
P2	15	25
P3	30	10
P4	45	15

If the CPU scheduling policy is SRTF, calculate the waiting time of process P2.

Solution-

Gantt Chart-



Gantt Chart

Now, we know-

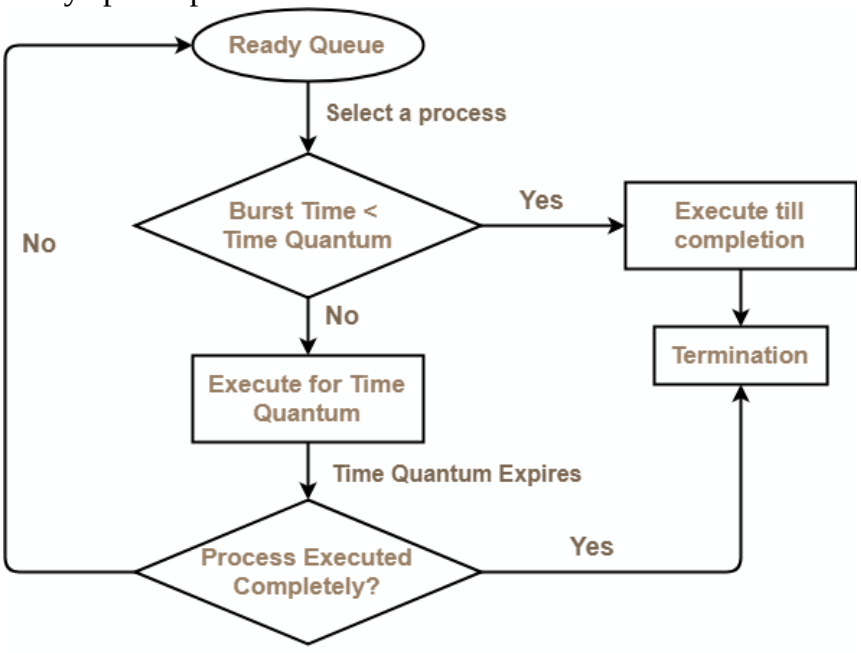
- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Thus,

- Turn Around Time of process P2 = 55 – 15 = 40 unit
- Waiting time of process P2 = 40 – 25 = 15 unit

(c) Round Robin Scheduling

- CPU is assigned to the process on the basis of FCFS for a fixed amount of time.
- This fixed amount of time is called as time quantum or time slice.
- After the time quantum expires, the running process is preempted and sent to the ready queue.
- Then, the processor is assigned to the next arrived process.
- It is always preemptive in nature.



Round Robin Scheduling

Advantages-

- It gives the best performance in terms of average response time.
- It is best suited for time sharing system, client server architecture and interactive system.

Disadvantages-

- It leads to starvation for processes with larger burst time as they have to repeat the cycle many times.
- Its performance heavily depends on time quantum.
- Priorities can not be set for the processes.

With decreasing value of time quantum,

- Number of context switch increases
- Response time decreases
- Chances of starvation decreases

Thus, smaller value of time quantum is better in terms of response time.

With increasing value of time quantum,

- Number of context switch decreases
- Response time increases
- Chances of starvation increases

Thus, higher value of time quantum is better in terms of number of context switch.

- With increasing value of time quantum, Round Robin Scheduling tends to become FCFS Scheduling.
- When time quantum tends to infinity, Round Robin Scheduling becomes FCFS Scheduling.
- The performance of Round Robin scheduling heavily depends on the value of time quantum.
- The value of time quantum should be such that it is neither too big nor too small.

Example-01:

Consider the set of 5 processes whose arrival time and burst time are given below-

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

If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turnaround time.

Solution-

Ready Queue- P5, P1, P2, P5, P4, P1, P3, P2, P1

Gantt Chart-

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

Now,

- Average Turn Around time = (13 + 11 + 3 + 6 + 10) / 5 = 43 / 5 = 8.6 unit
- Average waiting time = (8 + 8 + 2 + 4 + 7) / 5 = 29 / 5 = 5.8 unit

Problem-02:

Consider the set of 6 processes whose arrival time and burst time are given below-

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

If the CPU scheduling policy is Round Robin with time quantum = 2, calculate the *average waiting time* and *average turnaround time.*

Solution-

Ready Queue- P5, P6, P2, P5, P6, P2, P5, P4, P1, P3, P2, P1

Gantt chart-

Now, we know-

- Turn Around time = Exit time – Arrival time

- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	8	8 – 0 = 8	8 – 4 = 4
P2	18	18 – 1 = 17	17 – 5 = 12
P3	6	6 – 2 = 4	4 – 2 = 2
P4	9	9 – 3 = 6	6 – 1 = 5
P5	21	21 – 4 = 17	17 – 6 = 11
P6	19	19 – 6 = 13	13 – 3 = 10

Now,

- Average Turn Around time = (8 + 17 + 4 + 6 + 17 + 13) / 6 = 65 / 6 = 10.84 unit
- Average waiting time = (4 + 12 + 2 + 5 + 11 + 10) / 6 = 44 / 6 = 7.33 unit

Problem-03: Consider the set of 6 processes whose arrival time and burst time are given below-

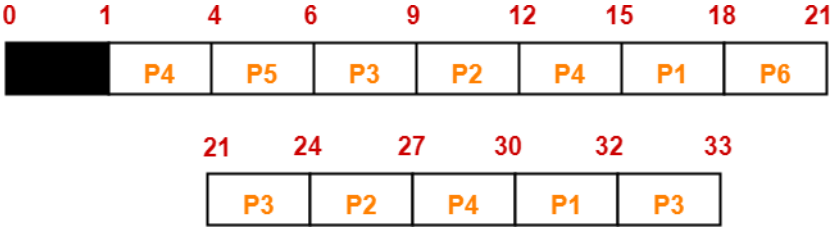
Process Id	Arrival time	Burst time
P1	5	5
P2	4	6
P3	3	7
P4	1	9
P5	2	2
P6	6	3

If the CPU scheduling policy is Round Robin with time quantum = 3, calculate the average waiting time and average turnaround time.

Solution-

Ready Queue- P3, P1, P4, P2, P3, P6, P1, P4, P2, P3, P5, P4

Gantt chart-



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	32	32 – 5 = 27	27 – 5 = 22
P2	27	27 – 4 = 23	23 – 6 = 17
P3	33	33 – 3 = 30	30 – 7 = 23
P4	30	30 – 1 = 29	29 – 9 = 20
P5	6	6 – 2 = 4	4 – 2 = 2
P6	21	21 – 6 = 15	15 – 3 = 12

Now,

- Average Turn Around time = (27 + 23 + 30 + 29 + 4 + 15) / 6 = 128 / 6 = 21.33 unit
- Average waiting time = (22 + 17 + 23 + 20 + 2 + 12) / 6 = 96 / 6 = 16 unit

(d) Priority Scheduling

- Out of all the available processes, CPU is assigned to the process having the highest priority.
- In case of a tie, it is broken by **FCFS Scheduling**.
- Priority Scheduling can be used in both preemptive and non-preemptive mode.

- The waiting time for the process having the highest priority will always be zero in preemptive mode.
- The waiting time for the process having the highest priority may not be zero in non-preemptive mode.

Priority scheduling in preemptive and non-preemptive mode behaves exactly same underfollowing conditions-

- The arrival time of all the processes is same
- All the processes become available

Advantages-

- It considers the priority of the processes and allows the important processes to run first.
- Priority scheduling in pre-emptive mode is best suited for real time operating system.

Disadvantages-

- Processes with lesser priority may starve for CPU.
- There is no idea of response time and waiting time.

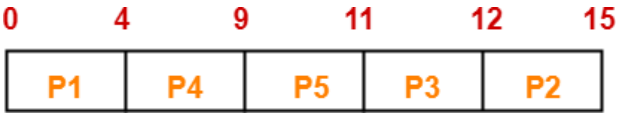
Problem-01:

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turnaround time. (*Higher number represents higher priority*)

Solution-
Gantt Chart-



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 0 = 4	4 – 4 = 0
P2	15	15 – 1 = 14	14 – 3 = 11
P3	12	12 – 2 = 10	10 – 1 = 9
P4	9	9 – 3 = 6	6 – 5 = 1
P5	11	11 – 4 = 7	7 – 2 = 5

Now,

- Average Turn Around time = (4 + 14 + 10 + 6 + 7) / 5 = 41 / 5 = 8.2 unit
- Average waiting time = (0 + 11 + 9 + 1 + 5) / 5 = 26 / 5 = 5.2 unit

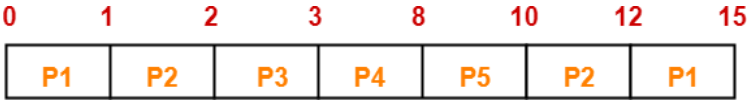
Problem-02: Consider the set of 5 processes whose arrival time and burst time aregiven below-

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

If the CPU scheduling policy is priority preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority).

Solution-

Gantt Chart-



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	15	15 – 0 = 15	15 – 4 = 11
P2	12	12 – 1 = 11	11 – 3 = 8
P3	3	3 – 2 = 1	1 – 1 = 0
P4	8	8 – 3 = 5	5 – 5 = 0
P5	10	10 – 4 = 6	6 – 2 = 4

Now,

- Average Turn Around time = (15 + 11 + 1 + 5 + 6) / 5 = 38 / 5 = 7.6 unit
- Average waiting time = (11 + 8 + 0 + 0 + 4) / 5 = 23 / 5 = 4.6 unit

(e)Multilevel Queue Scheduling

A multi-level queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue hasits own scheduling algorithm.

Let us consider an example of a multilevel queue-scheduling algorithm with five queues:

1. System Processes
2. Interactive Processes
3. Interactive Editing Processes
4. Batch Processes
5. Student Processes

Each queue has absolute priority over lower-priority queues. No process in the batch queue, for example, could run unless the queues for system processes, interactive processes, and interactive editing processes were all empty. If an interactive editing process entered the readyqueue while a batch process was running, the batch process will be pre-empted.