4...

CPU Scheduling

Objectives...

- To study about CPU Scheduling Concepts and Criteria.
- To get knowledge of CPU – I/O Burst Cycle.
- To learn about various Scheduling Algorithms such as First-Come First-Served Scheduling (FCFS), Shortest-Job-First Scheduling (SJF), Priority Scheduling, Round Robin (RR) Scheduling, Multilevel Queue Scheduling, Multilevel Feedback Queue Scheduling.

4.1 WHAT IS SHEDULING? [W-18; S-23]

- CPU scheduling is the basis of multiprogrammed operating systems.
- The idea of multiprogramming is relatively simply, if a process (job) is waiting for an I/O request, then the CPU switches from that job to another job, so the CPU is always busy in multiprogramming.
- But in a simple computer system, the CPU sit idle until the I/O request is granted.
- By switching the CPU among processes, the operating system can make the computer more productive.
- Scheduling is a fundamental operating system function, almost all the computer resources are scheduled before use.

CPU Scheduling Algorithm:

- The CPU scheduling algorithm determines how the CPU will be allocated to the process.
- CPU scheduling algorithms are two types one is non-preemptive and second one is preemptive scheduling algorithms.
- In the non-preemptive scheduling once the CPU is assigned to a process, the processor do not **release Switch** until the completion of that process.
- The CPU is assigned to some other job only after the previous job has been finished. But in the preemptive scheduling the CPU can release the processes even in the middle of the execution.
- For example, when the CPU executing the process P1 receives a request signal from process P2 in the middle of the execution, then the operating system compares the priorities of P1 and P2.

- If the priority of P1 is higher than P2, then the CPU continues the execution of process P1. Otherwise the CPU preempts the process P1 and assigns to process P2 [priority (P1 < P2)].
- CPU scheduling is the basis of multiprogrammed operating system. By switching the CPU among different processes, operating system can improve your degree of resource utilization.
- In this chapter, we can study different scheduling policies and how to select particular algorithm, which is best suited for our system.

4.2 SCHEDULING CONCEPTS

- The main goal of multiprogrammed system is to maximize the CPU utilization. For uniprocessor system there will never be more then one running process, all other processes are in state of waiting.
- In multiprogramming environment several processes are kept in memory at one time.
- When one process is in the state of wait, Operating system switches the CPU from this process to another one. This pattern continues among all.
- Scheduling is the one of the most important function of operating system so almost all computers resources are scheduled before use.
- So CPU is one of the primary computer resources for this reason its scheduling is central to operating system design.

4.2.1 CPU-I/O Burst Cycle [S-18]

- CPU-I/O burst cycle contains:
  - **CPU Burst:** A period of uninterrupted CPU activity.
  - **I/O Burst:** A period of uninterrupted input/output activity.

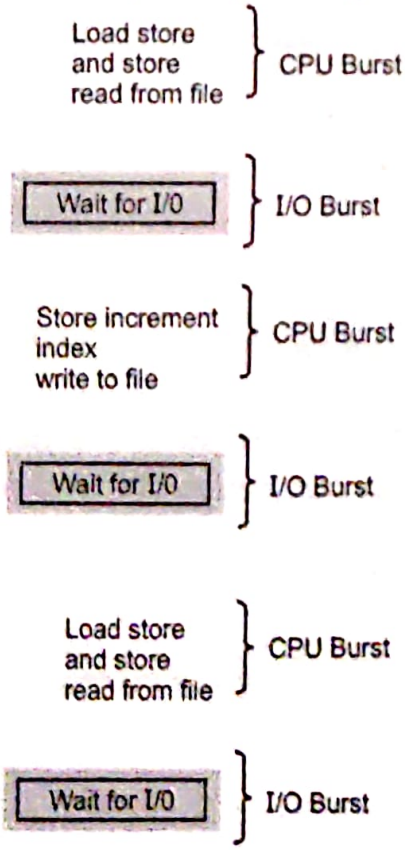


Fig. 4.1: Alternating Sequence of CPU and I/O Bursts

- Process execution consists of a cycle of CPU execution and I/O wait. Processes alternate back and forth between these two states. Process starts with CPU burst followed by I/O burst and so on.
- An I/O bound program would typically have many short CPU bursts where as CPU bound program consist of few long CPU bursts.

4.2.2 CPU Scheduler

- The operating system must select one of the processes in the ready queue to be executed. This is carried out by short term scheduler.
- A ready queue may be FIFO queue, Priority queue, Tree, Linked list.
- Operating system selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

4.2.3 Preemptive Scheduling [W-18; S-22]

- In this section, we will look at several methods of preemptive scheduling of the processor.
- We will assume a system where processes enter the processor scheduling system and remain there sometimes executing and sometimes waiting to execute until they have finished execution.
- By "finished execution", we do not mean that the process terminates rather we mean that the process becomes blocked waiting for an event waiting for a message or an I/O operation to complete.
- At that point, the process can no longer use the processor and so it is considered to be finished as shown in Fig. 4.2.

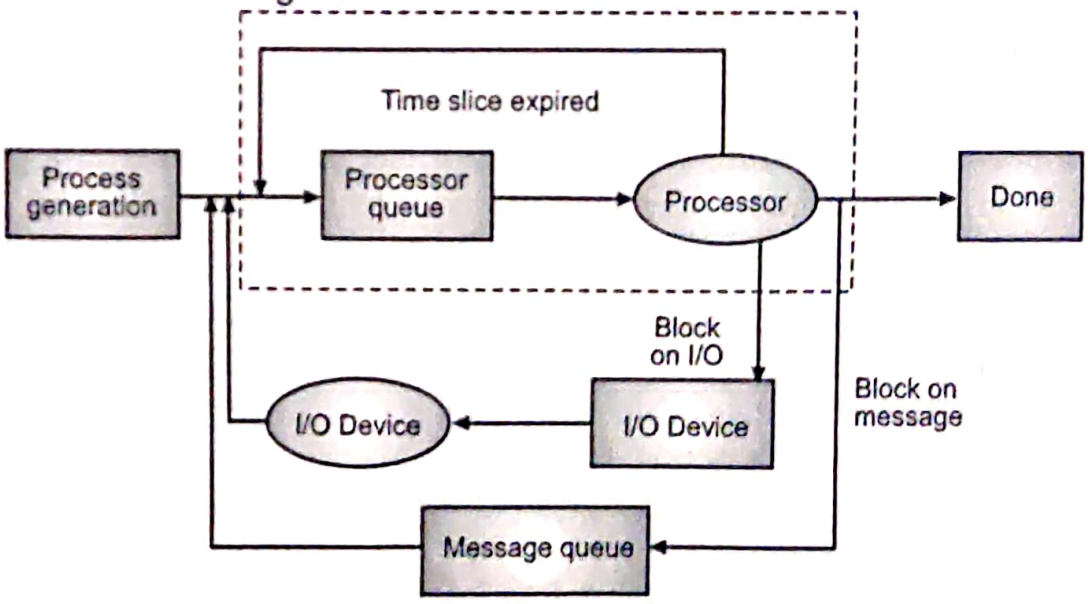


Fig. 4.2: The Flow of Processes in an Operating System

- The currently running process may be interrupted and moved to the ready state by the operating system. This is known as Preempting.
- The decision to preempt may be performed:
  - When a new process arrives.
  - When an interrupt occurs that places a blocked process in the ready state or
  - Periodically based on a clock interrupt.

- A process will enter and leave the processor scheduling system many times during its entire execution, may be hundreds of thousands of times. Still, each one is a separate transaction as far as the processor scheduling system is concerned.
- The processor scheduling algorithm might look at the history of the process, that is, what happened on previous trips through the processor scheduling system. For example, the scheduler might want to estimate how much processor time the process will consume on this trip though the scheduling system.
- One rough estimate would be the same amount of time it used last trip. A better estimate might be weighted average of the last 10 trips.

4.2.4 Non-preemptive Scheduling [W-18]

- In non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the working state.
- This method uses some hardware platforms. Microsoft Windows and Apple Macintosh Operating System use this type of scheduling method. Non-preemptive scheduling is attractive due to its simplicity.
- In non-preemptive scheduling, once a process is in the running state, it continues to execute until it terminates or blocks itself to wait for I/O or by requesting some operating system services.
- In short, we can define the preemptive and non-preemptive scheduling as follows:
  1. **Preemptive scheduling:** CPU allocated to a process may be switched if another process is scheduled which is of higher priority.
  2. **Non-preemptive scheduling:** Once the CPU has been allocated to a process, it keeps the CPU until process terminates or by switching to the wait state.

4.2.5 Dispatcher [S-19]

- A Dispatcher is a module; it connects the CPU to the process selected by the short-term scheduler.
- The main function of the dispatcher is switching, it means switching the CPU from one process to another process.
- The function of the dispatcher is 'jumping to the proper location in the user program and ready to start execution'.
- The dispatcher should be fast, because it is invoked during each and every process switch.
- **Dispatch Latency:** The time it takes by the dispatcher to stop one process and start another running is known as the 'dispatch latency'.
- The degree of multiprogramming is depending on the dispatch latency. If the dispatch latency is increasing then the degree of multiprogramming decreases.
- **Worst Case Latency:** This term is used for the maximum time taken for execution of all the tasks.

4.3 SCHEDULING CRITERIA [W-18, 19, 22]

- There are many different CPU - Scheduling algorithms are available and different CPU scheduling algorithms have different properties and may favour one class of processes over another.
- To choose a particular algorithm for a particular situation, we must consider the properties of the various algorithms.
- For comparing CPU - scheduling algorithms many criteria have been suggested. The characteristics used for comparison can make a substantial difference in the determination of the best algorithm.
- The following scheduling criteria are listed below:
  1. **CPU utilization:** Our main aim is to keep the CPU as busy as possible. The utilization of CPU may range from 0 to 100%. In real time, the CPU range is from 40% (for lightly loaded systems) to 90% (for heavily loaded systems).
  2. **Throughout:** If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes completed per time unit, called *throughput*. For long processes, this rate may be one process per hour, for short transactions, throughput might be 10 processes per second.
  3. **Turnaround time:** For a process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the *turnaround time*. It is the sum of the periods spends waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O.
  4. **Waiting time:** The CPU scheduling algorithm does not affect the amount of time during which a process executes or does I/O. The CPU - scheduling algorithm affects only the amount of time during which a process spends waiting in the ready queue. Waiting time is the addition of the periods spends waiting in the ready queue.
  5. **Response time:** Response time is the time from the submission of a request until the first response is produced or we can say that it is the amount of time it takes to start responding, but not the time that it takes the output that response. To guarantee that all users get good service, we may want to minimize the maximum response time.
  6. **Load average:** It is the average number of processes residing in the ready queue waiting for their turn to get into the CPU.
- In general CPU utilization and throughput are maximized and other factors are reduced for proper optimization.

4.4 SCHEDULING ALGORITHMS [S-18, 19]

- CPU scheduling algorithms deals with the problem of deciding which of the processes in the ready queue is to allocate the resource that is the CPU.
- There are many different algorithms, here we will discuss the below mentioned algorithms only.

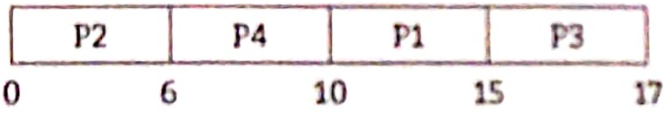
4.4.1 First-Come, First-Serve Scheduling (FCFS) [S-18]

- The simplest among all is the FCFS scheduling algorithm.
- The process that requests the CPU first, is allocated CPU first.
- It can be implemented with FIFO queue.
- When a process enters in a ready queue, get allocated with CPU.
- When CPU is free it is allocated at the head of the queue.
- The running process is then removed from the queue. The average waiting time under FCFS policy, however, is often quite long.

Example 1: Calculate Average Turnaround Time and Average Waiting Time for all set of processes using FCFS: [S-22]

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time	Turnaround Time
P1	5	1	9	15	14
P2	6	0	0	6	6
P3	2	2	13	17	15
P4	4	0	6	10	10

The Gantt chart for the schedule is:



Solution:

(i) Waiting time:

Waiting Time = Start Time – Arrival Time

P1 Wait Time = 9

P2 Wait Time = 0

P3 Wait Time = 13

P4 Wait Time = 6

$\therefore$ Average waiting time = $\frac{9 + 0 + 13 + 6}{4} = \frac{28}{4} = 7$ milliseconds

(ii) Turnaround Time:

Turnaround Time = Finished Time – Arrival Time

P1 Turnaround Time = 15 – 1 = 14 milliseconds

P2 Turnaround Time = 6 – 0 = 6 milliseconds

P3 Turnaround Time = 17 – 2 = 15 milliseconds

P4 Turnaround Time = 10 – 0 = 10 milliseconds

$\therefore$ Average waiting time = $\frac{14 + 6 + 15 + 10}{4} = \frac{45}{4} = 11.25$ milliseconds

- Here, we have the waiting period for process P1 is 9 and for process P2 waiting period is 0 and for process P3 the waiting period is 13 and for process P4 the waiting period is 6.

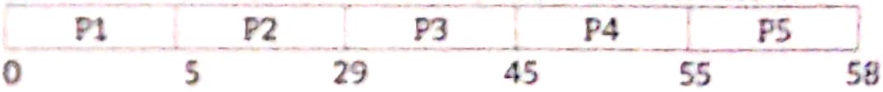
- Then the average waiting period/time is 4, therefore, we can say that the waiting Time is much better than the previous case.
- Now consider the performance of FCFS scheduling in dynamic situation. Assume that we have one CPU bound process and many I/O bound processes.
- As processes flow around the system, we may have the following scenario, the CPU bound process will get the CPU and it will hold it.
- During that time all other processes will finish their I/O and now they are ready to move into the ready queue, and waiting for the CPU. As the processes are waiting in the ready queue mean while the I/O devices are sitting idle.
- The CPU bound process finishes its CPU burst and moves to an I/O device for I/O related work. Now all the I/O bound processes, which have very short CPU burst gets execute quickly and moves back to the I/O queues.
- At this point the CPU remains idle. The CPU-bound process will then move back to the ready queue and be allocated the CPU. Again, all the I/O processes end up waiting in the ready queue until the CPU bound process is done.

Example 2: Consider the following set of processes which all arriving at time 0 and the burst time of each process is given below:

Process	Burst Time (milliseconds)
P1	5
P2	24
P3	16
P4	10
P5	3

The processes arrive in the order given below as P1, P2, P3, P4 and P5.

The Gantt chart for the above scenario would be as shown below:



Calculate:

(i) Average Waiting Time

(ii) Average Response Time

(iii) Average Turnaround time.

Solution:

(i) Average Waiting Time:

Waiting Time = Starting Time – Arrival Time

Process	Starting Time	Arrival Time	Waiting Time
P1	0	0	0
P2	5	0	5
P3	29	0	29
P4	45	0	45
P5	55	0	55

$$\therefore \text{Average waiting time} = \frac{0 + 5 + 29 + 45 + 55}{5}$$

$$= 26.8 \text{ milliseconds}$$

(ii) Average Response Time:

Response Time = First Response – Arrival Time

Process	Starting Time	Arrival Time	Waiting Time
P1	0	0	0
P2	5	0	5
P3	29	0	29
P4	45	0	45
P5	55	0	55

$$\therefore \quad \text{Average Response Time} = \frac{0 + 5 + 29 + 45 + 55}{5}$$

$$= 26.8 \text{ milliseconds}$$

(iii) Average Turnaround Time:

Turnaround Time = Finished time – Arrival Time

Process	Starting Time	Arrival Time	Waiting Time
P1	5	0	5
P2	29	0	29
P3	45	0	45
P4	55	0	55
P5	58	0	58

$$\therefore \quad \text{Average Turnaround Time} = \frac{5 + 29 + 45 + 55 + 58}{5}$$

$$= 38.4 \text{ milliseconds}$$

Since, it is non-preemptive scheduling algorithms the average response time and average waiting time is same in both the cases.

Note: Non-preemptive means once the CPU is allocated to one process it keeps it as long as it needs and release only when it finishes its jobs either by terminating or by requesting an I/O.

Example 3: In Example 2, the arrival time for all the processes was 0. But if we change the arrival time of the processes then the whole calculation changes as shown below.

Process	CPU burst Time	Arrival Time
P1	3	0
P2	6	2
P3	4	4
P4	5	6
P5	2	8

Order of Arrival of processes is P1, P2, P3, P4 and P5.

P1 arrived at Time 0 milliseconds

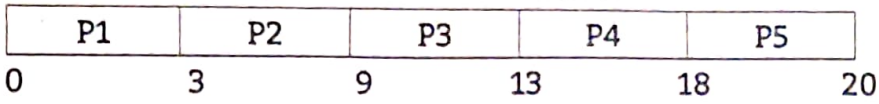
P2 arrived after 2 milliseconds

P3 arrived after 4 milliseconds

P4 arrived after 6 milliseconds

P5 arrived after 8 milliseconds.

The Gantt chart of the above problem is shown below:



Calculate:

(i) Average Relative Delay

(ii) Average Response Time

(iii) Average Turnaround Time.

Solution:

(i) Average Relative Delay:

$$\text{Relative delay} = \frac{\text{Turn around time}}{\text{Burst Time}}$$

Process	Turnaround Time	Burst Time	Relative delay
P1	3	3	1
P2	7	6	1.16
P3	9	4	2.25
P4	12	5	2.4
P5	12	2	6

$$\text{Average relative delay} = \frac{1 + 1.16 + 2.25 + 2.4 + 6}{5}$$

$$= \frac{12.81}{5} = 2.56 \text{ milliseconds}$$

(ii) Average Turnaround Time:

Turnaround Time = Finish Time – Arrival Time

Process	Finish Time	Arrival Time	Turnaround Time
P1	3	0	3
P2	9	2	7
P3	13	4	9
P4	18	6	12
P5	20	8	12

$$\therefore \quad \text{Average Turnaround Time} = \frac{3 + 7 + 9 + 12 + 12}{5}$$

$$= 8.6 \text{ milliseconds}$$

(iii) Average Response Time:

Response Time = First Response Time – Arrival Time

Process	Finish Time	Arrival Time	Response Time
P1	0	0	0
P2	3	2	1
P3	9	4	5
P4	13	6	7
P5	18	8	10

$$\therefore \quad \text{Average Response time} = \frac{0 + 1 + 5 + 7 + 10}{5}$$

$$= 4.6 \text{ milliseconds}$$

Advantages:

1. Easy to implement.
2. It is very simple.

Disadvantages:

1. Difficult with some time sharing systems.
2. Average waiting time is very high with respect to others.
3. Because of this performance is affected or degraded.

Note: To summarize FCFS algorithm

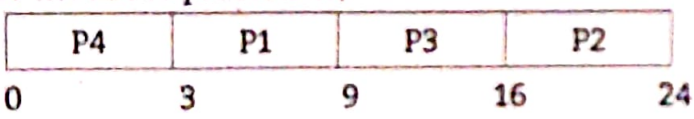
- FCFS scheduling algorithm is non-preemptive.
- For time-sharing system we cannot implement FCFS, because process will hold the CPU until it finishes or changes a state to wait state.
- Average waiting time for FCFS algorithm is not minimal, and it also varies substantially if the process CPU burst time vary greatly.

4.4.2 Shortest-Job-First Scheduling (SJF)

- The Shortest-Job-First (SJF) is the method of CPU scheduling.
- It allocates the CPU to a process having smallest next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
- If the two processes having same CPU burst then they will be scheduled according to FCFS algorithm. For example consider the following set of processes.

Process	Burst-time (milliseconds)
P1	6
P2	8
P3	7
P4	3

- The Gantt chart for the above problem is,



Waiting Time for Process P1 = 3 milliseconds

Waiting Time for Process P2 = 16 milliseconds

Waiting Time for Process P3 = 9 milliseconds

Waiting Time for Process P4 = 0 milliseconds

$\therefore$ Average Waiting Time = $\frac{(3 + 16 + 9 + 4)}{4}$ = 8 milliseconds

- The SJF scheduling algorithm is probably optimal; it gives minimal average waiting time for a given set of processes.
- By moving short process before a long one, the waiting time of short process decreases. Consequently average waiting time reduces.
- The real difficulty with SJF knows the length of next CPU request so it can not be implemented at the level of short term scheduling. Instead it can be used for long term scheduling where user estimates the process time limit.

- At short-term scheduling, there is no way to find out the length of next CPU burst. One approach is to approximate SJF scheduling. We can predict the next CPU burst from previous value.
- The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU burst.
- Let t_n be the length of the n^{th} CPU burst, and let T_{n+1} be our predicted value for the next CPU burst. Then for α, $0 <= \alpha <= 1$,

$$\boxed{T_{n+1} = \alpha t_n + (1 - \alpha) T_n}$$

- This gives exponential average. Here,
  - t_n : Most recent information
  - T_n : Stores the past history
  - α : Relative weight of recent and past history

If $\alpha = 0$, Then $T_{n+1} = T_n$ (Current T_n and recent having same values)

If $\alpha = 1$, Then $T_{n+1} = t_n$ (Most recent CPU burst matters)

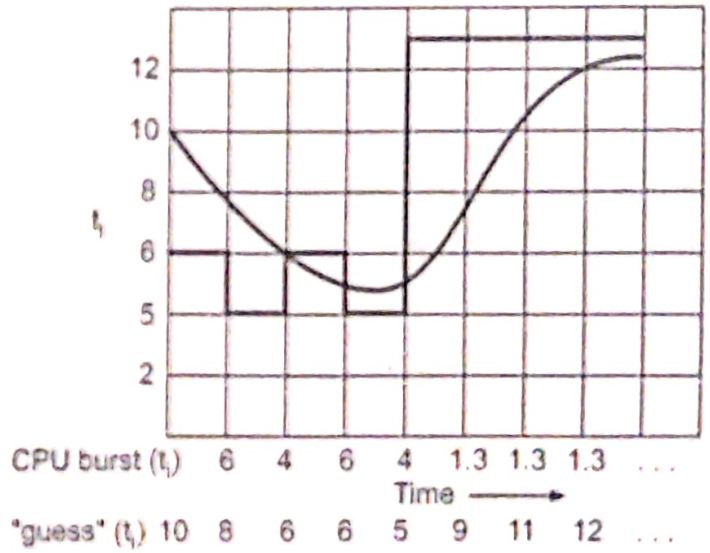


Fig. 4.3: Prediction of the length of the next CPU Burst

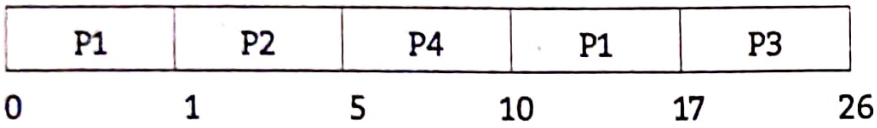
Types of SJF Algorithm:

- Two types of SJF algorithm are:
  1. Preemptive
  2. Non-preemptive
- The choice arises when a new process arrives at the ready queue while a previous process is executing.
- The new process may have a shorter next CPU burst than what is left at the currently executing process.
- **A Preemptive SJF algorithm** will preempt the currently executing process, whereas as a **Non-preemptive SJF** algorithm will allow the currently running process to finish its CPU burst.
- **Preemptive SJF scheduling is sometimes called shortest-remaining time first scheduling.**

- For example consider the processes below:

Process	Arrival Time	Burst-time (milliseconds)
P1	0	8
P2	1	4
P3	2	9
P4	3	5

- The Gantt chart for the above problem is shown below:



- Process P1 arrives at Time 0.
- Process P2 arrives at Time 1. The remaining time for process P1 is 7 milliseconds is larger than the time required by process P2 i.e. 4 milliseconds, so process P1 is preempted and process P2 is scheduled.
- The average waiting time for this is:

  Formula for waiting time in Preemptive SJF Scheduling:

  Waiting time = Start time – Arrival time + New start time – Old finish time.

  $\therefore$ Waiting time $= \frac{((10-1)+(1-1)+(17-2)+(5-3))}{4} = \frac{(9+0+15+2)}{4}$

  $\therefore \quad = \frac{26}{4}$

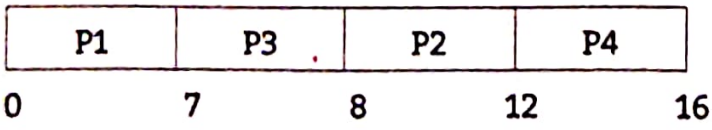
  $\therefore \quad = 6.5$ milliseconds
- A **Non-preemptive SJF scheduling** would have average waiting time at 7.75 milliseconds. The calculation is left for the students to solve.

Example 4: Non-Preemptive SJF.

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

Solution:

SJF (Non-Preemptive)



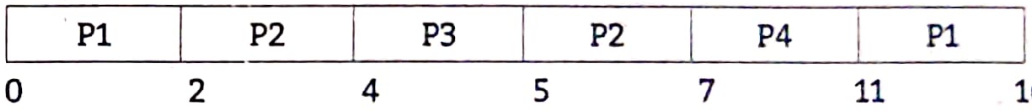
$\therefore$ Average waiting time $= \frac{0+6+3+7}{4} = \frac{16}{4} = 4$

Example 5: Preemptive SJF.

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

Solution:

SJF (Preemptive)

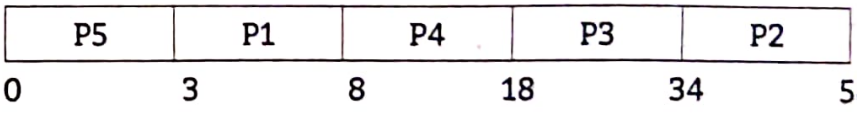


$\therefore$ Average waiting time $= \frac{9+1+0+2}{4} = \frac{12}{4} = 3$

Example 6: Consider the process and CPU burst time in milliseconds. [S-23]

Process	CPU Burst-time (milliseconds)
P1	5
P2	24
P3	16
P4	10
P5	3

The Gantt chart for the above problem is shown below:



By observing the above example, we see that P5 has the shortest CPU burst time. So we allocate P5 the CPU first. After finishing P5 the next shortest job is searched which is P1 and CPU is given to P1 the same process continuous till all the processes are finished. Now we calculate the average waiting time, Average turnaround time and Average response time.

(i) Average Waiting Time:

Waiting time = Starting time – Arrival time

Process	Starting Time	Arrival Time	Waiting Time
P1	3	0	3
P2	34	0	34
P3	18	0	18
P4	8	0	8
P5	0	0	0

$\therefore$ Average waiting time $= 3 + 34 + 18 + 8 + 0$

$= \frac{63}{5} = 12.6$ milliseconds

(ii) Average Turnaround Time:

Turnaround time = Finish time – Arrival time

Process	Finish Time	Arrival Time	TurnAround Time
P1	8	0	8
P2	58	0	58
P3	34	0	34
P4	18	0	18
P5	3	0	3

$$\therefore \text{ Average turnaround time } = \frac{8 + 58 + 34 + 18 + 3}{5}$$

$$= \frac{121}{5} = 24.2 \text{ milliseconds}$$

(iii) Average Response Time:

Response time = First Response Time – Arrival time

Process	First Response Time	Arrival Time	Response Time
P1	3	0	3
P2	34	0	34
P3	18	0	18
P4	8	0	8
P5	0	0	0

$$\therefore \text{ Average Response time } = \frac{3 + 34 + 18 + 8 + 0}{5}$$

$$= \frac{63}{5} = 12.6 \text{ milliseconds}$$

Advantage:

1. It has the least Average waiting time, Average turnaround time and Average Response time.

Disadvantages:

1. It is difficult to know the length of the next CPU burst time.
2. It is optimal algorithm it cannot be implemented in short-term CPU scheduling.
3. Aging is another problem where big jobs are waiting for long-time in the CPU.
4. Starvation of process having long burst time may cause because processor is selecting the process having smallest burst time.

4.4.3 Priority Scheduling [S-23]

- SJF algorithm is a special case of Priority scheduling algorithm.
- Priority is assigned to each process and CPU is allocated to the process with highest priority. Equal priority process is scheduled with FCFS algorithm.

- Priorities are the numbers ranging from 0 to 7 or 0 to 4095 have given to each processes. Some systems use low numbers to represent low priority and other use low numbers for high priority. Here we use low numbers to represent high priority.
- Protocol is set whether 0 is the highest or lowest priority.
  1. **Starvation:** A major problem with priority scheduling is indefinite blocking or starvation. The high priority process indefinitely blocks a low priority process in a heavily loaded system. As per algorithm high priority processes can prevent the low priority process for allocating CPU and thus such a process will never get a chance to allocate a CPU.
  2. **Aging:** A solution to this problem is to gradually increase the priority of a process that wait in the system for a long time called Aging.

Note: To summarize Priority Scheduling

- A priority number (integer) is associated with each process.
- The CPU is allocated to the process with the highest priority (smallest integer = highest priority).
- Preemptive
- Non-preemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time.
- Problem = Starvation-low priority processes may never execute.
- Solution = Aging-as time progresses increase or decrease the priority of the process.

- Priorities are of two types:
  1. **Internal priorities:** Internally defined priorities use some measurable quantity to compute the priority of a process.
  2. **External priorities:** External priorities are set by criteria that are external to the operating system, such as the importance of the process.

Modes of Priority Scheduling:

- Priority scheduling are of two modes:
  1. Preemptive
  2. Non-preemptive.
- When a process arrives at the ready queue, its priority is compared with the priority of the currently running process.
- A **Preemptive priority**: This scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process.
- A **Non-preemptive priority**: This scheduling algorithm will simply put the new process at the head of the ready queue.

Advantages:

1. It considers the priority of the processes and allows the important processes to run first.
2. Priority scheduling in preemptive mode is best suited for real time operating system.
3. Priorities in the Priority scheduling are chosen on the basis of time requirements, memory requirements, and user preferences.

Disadvantages:

1. Processes with lesser priority may starve for CPU.
2. In priority scheduling, if the system is crashed, then all low-priority processes that are not yet completed will also get lost.
3. There is no idea of response time and waiting time.

Example 7: Consider the following set of processes. All processes arrived time 0 and in the order P1, P2, P3, P4, P5 having CPU burst time in milliseconds. Calculate average waiting time using Non-Preemptive Priority Scheduling Technique.

Solution:

Process	Burst Time (in milliseconds)	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The Gantt chart is:

P2	P5	P1	P3	P4
0	1	6	16	18 19

Waiting time of process P1 = 6 milliseconds
Waiting time of process P2 = 0 milliseconds
Waiting time of process P3 = 16 milliseconds
Waiting time of process P4 = 18 milliseconds
Waiting time of process P5 = 1 milliseconds

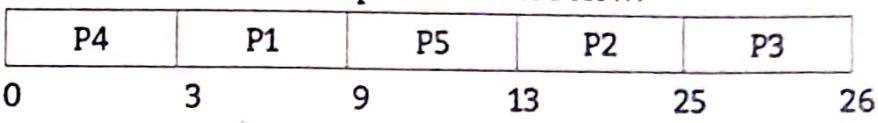
Average waiting time = $\frac{6 + 0 + 16 + 18 + 1}{5}$ = 8.2 milliseconds

Example 8: Consider the following set of processes:

Process	CPU Burst Time (in milliseconds)	Priority
P1	6	2
P2	12	4
P3	1	5
P4	3	1
P5	4	3

By observing the above example we can say that process P4 has highest priority. Therefore, we allocate CPU to process P4 first. The next priority is given to process P1 and CPU is allocated to process P1, next priority is given to process P5, so the CPU is allocated to P5 and this process continuous until all the process are completed.

The Gantt chart for the above example is shown below:



(i) The Average Waiting Time:

Process	Waiting time
P1	3
P2	13
P3	25
P4	0
P5	9

$$\text{Average waiting time} = \frac{3 + 13 + 25 + 0 + 9}{5}$$

$$= \frac{50}{5}$$

$$= 10 \text{ milliseconds}$$

(ii) The Average Turnaround Time:

Process	Turnaround time
P1	9
P2	25
P3	26
P4	3
P5	13

$$\therefore \quad \text{Average turnaround time} = \frac{9 + 25 + 26 + 3 + 13}{5}$$

$$= \frac{76}{5}$$

$$= 15.2 \text{ milliseconds}$$

4.4.4 Round Robin Scheduling

[W-18, S-19, 23]

- Round Robin (RR) Scheduling is designed for time-sharing system.
- This Scheduling is also called as FCFS scheduling along with preemption to switch between processes.
- In this scheduling algorithm, we will define a time slice or time quantum generally from 10 to 100 ms. Ready queue is treated as circular queue and processes are entered in ready queue as they are coming in the system.
- CPU scheduler goes around this circular queue and CPU is allocated to each process for a fixed time slice.
- To implement RR algorithm, we keep the ready queue as a FIFO.
- New processes are added to the tail of the ready queue.

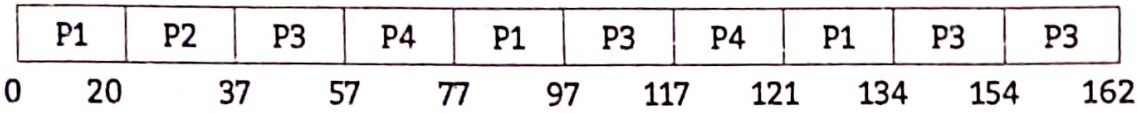
- The CPU scheduler picks from the ready queue, sets a timer to interrupt after 1 time quantum which will interrupt the operating system.
- A context switch will be executed; process will be put at a tail of ready queue.

Example 9: Example of RR with Time Quantum = 20

Process	Burst Time
P1	53
P2	17
P3	68
P4	24

Solution:

The Gantt chart is:



Average waiting time is (0 + 20 + 37 + 57 + 57 + 40 + 40) ms.

- Typically, higher average turnaround than SJF, but better response.
- In RR scheduling algorithm, no process is allocated to the CPU for more than one time quantum in a row. If a process CPU burst exceeds 1 time quantum, that process is preempted and is put back in ready queue. RR scheduling algorithm is preemptive.
- If there are n processes in ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units.
- Each process must wait no longer than (n-1)*q time units until its next time quantum.
- For example, if there are 5 processes, with a time quantum of 20 ms, then each process will get up to 20 ms every 100 ms.
- The performance of RR algorithm depends heavily on the size of time quantum.
- At one extreme, if the time quantum is very large (Infinite), RR policy is the same as the FCFS policy.
- If the time quantum is very small (say 1 ms), RR approach is called processor sharing, and appears (in theory) to the users as though each n processes has its own processor running 1/n in the speed of the real processor. This approach was used in the Control Data Corporation (CDC).
- In software, however, we need also to consider the effect of context switching on the performance of RR scheduling.
- Let's assume that we have only one process of 10 time units. If the quantum 12 time units, the process finishes in less than 1 time quantum with no overhead.
- If the quantum is 6 time units, however the process required 2 quantum, resulting in a context switch. If the time quantum is 1 time unit, 9 context switches will occurred, slowing the execution of the process accordingly, (Fig. 4.4)

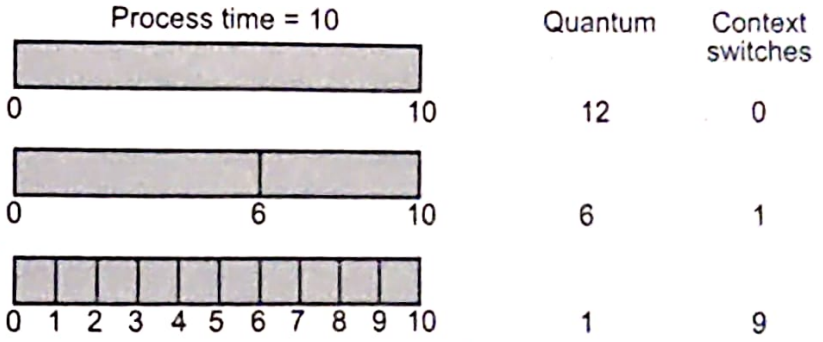
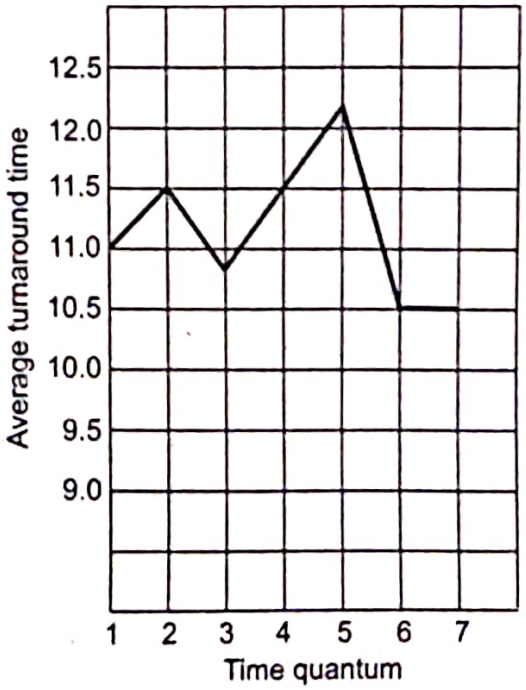


Fig. 4.4: Time Quantum and Context Switch Time

- Thus, we want the time quantum to be large with respect to the context switch time. If the context switch time is approximately 10 % of the time quantum, then about 10% of the CPU time will be spent in the context switch.
- Turnaround time also depends on the size of time quantum. As we can see from Fig. 4.5, the average turnaround time of a set of processes does not necessarily improve as the time quantum size increases.
- In general, the average turnaround time can be improved if most processes finish their next CPU burst in a single time quantum.
- For example, given three processes of 10 time units each and a quantum of 1 time unit, the average turnaround time is 29. If the time quantum is 10, however, the average turnaround time drops to 20.
- If context-switch time is added in, the average turnaround time increases for a smaller time quantum, since more context switches will be required.



Process	Time
P_1	6
P_2	3
P_3	1
P_4	7

Fig. 4.5: Turnaround time varies with the Time Quantum

- On the other hand, if the time quantum is too large, RR scheduling degenerates to FCFS policy. A rule of thumb is that 80% of CPU bursts should be shorter than the time quantum.

Advantages:

1. If you know the total number of processes on the run queue, then you can also assume the worst-case response time for the same process.
2. It deals with all process without any priority.
3. Allows OS to use the Context switching method to save states of preempted processes.
4. It gives the best performance in terms of average response time.

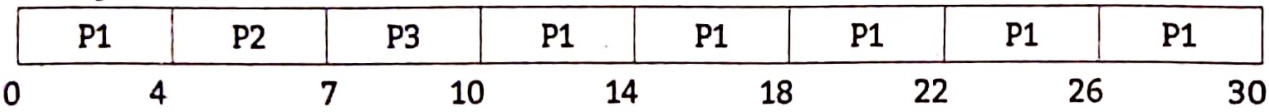
Disadvantages:

1. Its performance heavily depends on time quantum.
2. Deciding a perfect time quantum is really a very difficult task in the system.
3. Priorities can not be set for the processes.

Example 10: Consider the following set of process that arrives at time 0, with the length of the CPU burst time given in milliseconds.

Process	Burst Time
P1	24
P2	3
P3	3

Solution: If we use a time quantum of 4 milliseconds, then process P1 gets the first 4 milliseconds. Since, it requires another 20 milliseconds, it is preempted after the first time quantum and the CPU is given to the next process in the queue process P2. Since, process P2 does not need 4 milliseconds, it quits before its time quantum expires. The CPU is then given to the next process in the queue, process P3. Once, each process has received 1 time quantum, the CPU is returned to process P1 for an additional time quantum. The resulting RR schedule is:



Round robin scheduling is always Preemptive Scheduling.

Waiting time = Start time – arrival time + new start time – old finish time.

Waiting time for process P1 = (0 – 0) + (10 – 4) = 6 milliseconds

Waiting time for process P2 = (4 – 0) = 4 milliseconds

Waiting time for process P3 = (7 – 0) = 7 milliseconds

$\therefore$ Average Waiting Time is $\frac{17}{3}$ = 5.66 milliseconds.

In RR scheduling algorithm, no process is allocated to the CPU for more than 1 time quantum in a row. If a process CPU burst time exceeds 1 time quantum, that process is preempted and is input back in the ready queue. The RR scheduling algorithm is preemptive.

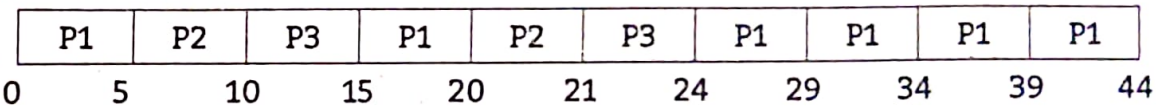
Example 11: Consider the following set of processes:

Process	CPU Burst Time (in milliseconds)
P1	30
P2	6
P3	8

Given the quantum is of 5 milliseconds.

Solution: Initially process P1 is given 5 milliseconds. When time quantum of P1 expired, the CPU switches from process P1 to P2. When the time quantum of P2 expired, the process switches to process P3. When time quantum of P3 expired, the CPU switch to P1 as P1 was in the ready queue.

The below Gantt chart shows this:



(i) Average Waiting Time:

Waiting time = Start time − Arrival time + New start time − Old finish time

Waiting time For P1 = 0 + (15 − 5) + (24 − 20) = 10 + 4=14 milliseconds

P2 = 5 + (20 − 10) = 15 milliseconds

P3 = 10 + (21 − 15) = 16 milliseconds

Average waiting time = 45/3 = 15 milliseconds

(ii) Average Turnaround Time:

Turnaround time for P1 = 44

Turnaround time for P2 = 21

Turnaround time for P3 = 24

$$\therefore \quad \text{Average Turnaround Time} = \frac{44 + 21 + 24}{3}$$

$$= \frac{89}{3}$$

$$= 29.66 \text{ milliseconds}$$

(iii) Average Response Time:

Response time for P1 = 0

Response time for P2 = 5

Response time for P3 = 10

$$\therefore \quad \text{Average Response Time} = \frac{0 + 5 + 10}{3}$$

$$= \frac{15}{3}$$

$$= 5 \text{ milliseconds}$$

The performance of round robin depends on the size of time quantum chosen.

4.4.4.1 Multilevel Queues

- These scheduling algorithms are created for areas in which we classify process into different groups.
- For example, a common division can be made between foreground (or interactive) processes and background (or batch) processes.
- Priority of foreground processes may be higher than background processes.
- The multilevel queue - scheduling algorithm partitions the ready queue into several separate queues as shown in Fig. 4.6.

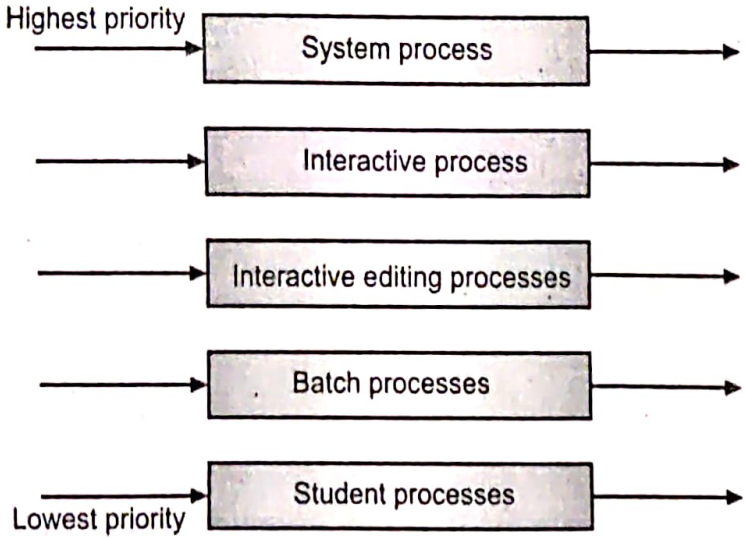


Fig. 4.6: Multilevel Queue Scheduling

- The processes are assigned to one queue depending on some property of the process.
- The property may be memory size, process priority or may be process type. Each queue is associated with its own scheduling algorithm.
- For example, separate queues might be used for foreground and background processes. The foreground queue might be scheduled by an RR algorithm and the background queue is scheduled by an FCFS algorithm.

4.4.4.2 Multilevel Feedback Queues [S-18]

- Multilevel feedback queue allows a process to move between queues. The idea is to separate process with different CPU - Burst characteristics.
- If a process uses too much CPU times, it will be moved to a lower priority queue. This leaves I/O bound and interactive processes in the higher priority queues.
- If a process waits for long time in a lower priority queue, then it is moved to a higher priority queue. This form of aging prevents starvation.
- For example, consider a multilevel feedback queue scheduler with three queues and number them from 0 to 2 as shown in Fig. 4.7.

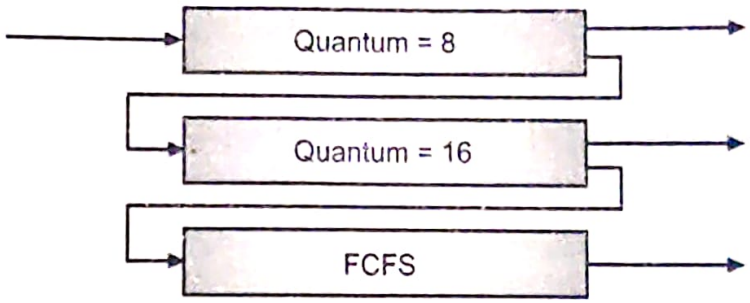


Fig. 4.7: Multilevel Feedback Queues

- The first task the scheduler does is that it executes all processes in queue 0. Only when queue 0 is empty then only it will go to queue 1 and then executes the process there in queue 1.
- Similarly, when queue 0 and queue 1 are empty then only it will process the processes in queue 2. But when a process is getting executed in queue 2 and at that time if a process arrives for queue 1 then it will preempt a process in queue 2. Similarly, when a process arrives for queue 0 will in turn preempt a process in queue 1.
- A process entering the ready queue is put in queue 0.
- A process is queue 0 is given a time quantum of 8 milliseconds.
- If it does not finish within this time, it is moved to the tail of queue 1.
- If queue 0 is empty, the process at the head of queue 1 is given a quantum of 16 milliseconds. If it does not complete, it is preempted and is put into queue 2.
- Processes in queue 2 are run on an FCFS basis only when queues 0 and 1 are empty.

Parameters:

- The below are the parameters for a multilevel feedback queue scheduler:
  - The number of queues.
  - The scheduling algorithm for each queue.
  - The method used to determine when to upgrade a process to a higher-priority queue.
  - The method used to determine when to demote a process to a lower priority queue.
  - The method used to determine which queue a process will enter when that process needs service.

SOLVED EXAMPLES OF PREVIOUS EXAMS

Example 1: Consider the following set of processes with the length of CPU burst time and arrival time.

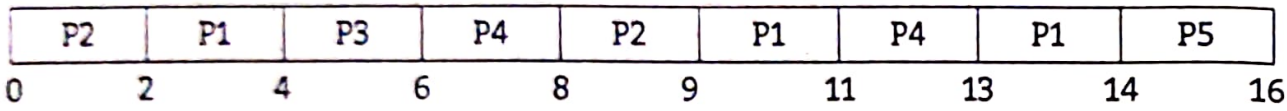
Process	Burst time	Arrival time
P1	5	1
P2	3	0
P3	2	2
P4	4	3
P5	2	13

Calculate turnaround time, waiting time, average waiting time and average turnaround time using Round Robin Algorithm with time quantum = 2.

Solution:

Process	Burst Time (ms)	Arrival Time (ms)	Start Time (ms)	Finish Time (ms)	Waiting Time (ms)	Turnaround Time (ms)
P1	5	1	2	14	8	13
P2	3	0	0	9	6	9
P3	2	2	4	6	2	4
P4	4	3	6	13	6	10
P5	2	13	14	16	1	3

The Gantt chart for the above scenario would be as shown below:



(i) **Waiting Time:** The Waiting time formula for Round Robin Scheduling.

Waiting Time = Starting Time – Arrival Time +new start time – old finish time

So waiting time of P1 = (2 – 1) + (9 – 4) + (13 – 11)
= 1 + 5 + 2
= 8 ms

Waiting time of P2 = (0 – 0) + (8 – 2)
= 6 ms

Waiting time of P3 = (4 – 2)
= 2 ms

Waiting time of P4 = (6 – 3) + (11 – 8) = (3 + 3)
= 6Ms

Waiting time of P5 = (14 – 13) = 01 ms

Formula for Average Waiting time = $\frac{\text{Sum of all processes waiting time}}{\text{Total number of processes}}$

$\therefore$ Average Waiting Time = $\frac{8 + 6 + 2 + 6 + 1}{5}$

= 4.6 milliseconds

Turnaround Time:

Turnaround Time = Finished time – Arrival Time

Process	Turnaround Time
P1	13
P2	9
P3	4
P4	10
P5	3

(ii) **Average Turnaround time** = $\frac{\text{Sum of turnaround time of processes}}{\text{Total number of processes}}$

$\therefore$ Average Turnaround Time = $\frac{13 + 9 + 4 + 10 + 3}{5}$

= 7.8 milliseconds

Example 2: Consider the following set of processes with the length of CPU Burst time and Arrival time given in milliseconds:

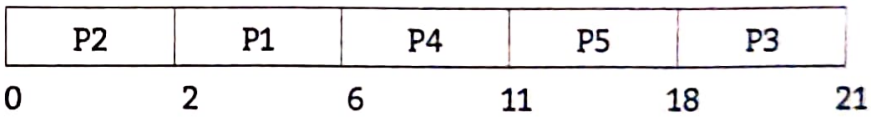
Process	Burst time	Arrival time
P1	4	1
P2	2	0
P3	3	3
P4	5	2
P5	7	2

Calculate turnaround time, waiting time, average waiting time and average turnaround time using FCFS algorithm.

Solution:

Process	Burst Time (ms)	Arrival Time (ms)	Start Time(ms)	Finish Time (ms)	Waiting Time (ms)	Turnaround Time (ms)
P1	4	1	2	6	1	5
P2	2	0	0	2	0	2
P3	3	3	18	21	15	18
P4	5	2	6	11	4	9
P5	7	2	11	18	9	16

The Gantt chart for the above scenario would be as shown below:



(i) Waiting Time:

The Waiting time formula for Non Preemptive scheduling using FCFS:

Waiting Time = Starting Time – Arrival Time

So, Waiting Time of P1 = 2 – 1 = 1 ms

Waiting Time of P2 = 0 – 0 = 0 ms

Waiting Time of P3 = 18 – 3 = 15 ms

Waiting Time of P4 = 6 – 2 = 4 ms

Waiting Time of P5 = 11 – 2 = 9 ms

(ii) Average Waiting time:

$$\text{Average Waiting time} = \frac{\text{Sum of all processes waiting time}}{\text{Total number of processes}}$$

$$\therefore \quad \text{Average Waiting Time} = \frac{1 + 0 + 15 + 4 + 9}{5}$$

$$= 5.8 \text{ milliseconds}$$

(iii) Turnaround Time:

Turnaround Time = Finished time – Arrival Time

Process	Turnaround Time
P1	5
P2	2
P3	18
P4	9
P5	16

(iv) **Average Turnaround time** = $\frac{\text{Average Turn Around time}}{\text{Total number of processes}}$

$\therefore$ Average Turnaround Time = $\frac{5+2+18+9+16}{5}$

= 10 milliseconds

Example 3: Consider the following set of processes with the length of CPU Burst time and arrival time in milliseconds: [W-22]

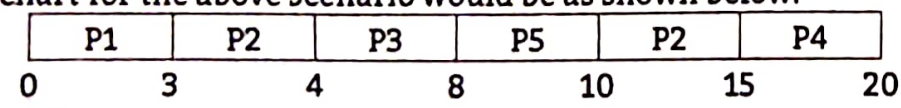
Process	Arrival	Time Burst Time
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

Calculate turnaround time, waiting time, average waiting time and average turnaround time using preemptive SJF Scheduling algorithm.

Solution:

Process	Burst Time (ms)	Arrival Time (ms)	Start Time(ms)	Finish Time (ms)	Waiting Time (ms)	Turnaround Time (ms)
P1	0	3	0	3	0	3
P2	2	6	3	15	7	13
P3	4	4	4	8	0	4
P4	6	5	15	20	9	14
P5	8	2	8	10	0	2

The Gantt chart for the above scenario would be as shown below:



(i) Waiting Time:

The Waiting time formula for SJF Preemptive Scheduling Algorithm.

Waiting Time = Starting Time – Arrival Time + New start time – Old finish time

So waiting time of P1 = (0 – 0) = 0 ms

Waiting time of P2 = (3 – 2) + (10 – 4) = 7 ms

Waiting time of P3 = (4 – 4) = 0 ms

Waiting time of P4 = (15 – 6) = 9 ms

Waiting time of P5 = (8 – 8) = 0 ms

(ii) Average Waiting time:

$$\text{Average Waiting time} = \frac{\text{Sum of all processes waiting time}}{\text{Total number of processes}}$$

$$\therefore \quad \text{Average Waiting Time} = \frac{0+7+0+9+0}{5}$$

$$= 3.2 \text{ milliseconds}$$

(iii) Turnaround Time:

Turnaround Time = Finished time – Arrival Time

Process	Turnaround Time
P1	3
P2	13
P3	4
P4	14
P5	2

(iv) Average Turnaround time:

$$\text{Average Turnaround Time} = \frac{\text{Sum of turnaround time of processes}}{\text{Total number of processes}}$$

$$\therefore \quad \text{Average Turnaround Time} = \frac{3+13+4+14+2}{5}$$

$$= 7.2 \text{ milliseconds}$$

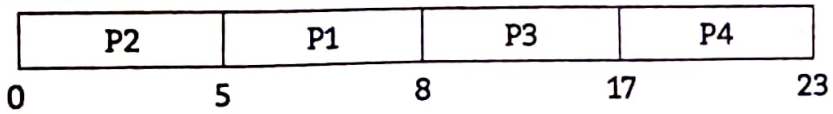
Example 4: Calculate average turnaround time and average waiting time for all set of processes using FCFS algorithm:

Process	Burst Time	Arrival Time
P1	3	1
P2	5	0
P3	9	4
P4	6	5

Solution:

Process	Burst Time (ms)	Arrival Time (ms)	Start Time(ms)	Finish Time (ms)	Waiting Time (ms)	Turnaround Time (ms)
P1	3	1	5	8	4	7
P2	5	0	0	5	0	5
P3	9	4	8	17	4	13
P4	6	5	17	23	12	18

The Gantt chart for the above scenario would be as shown below:



(i) Waiting Time:

The Waiting time formula for FCFS Scheduling:

Waiting Time = Starting Time – Arrival Time

So waiting time of P1 = (5 – 1) = 4 ms

Waiting time of P2 = (0 –0) = 0 ms

Waiting time of P3 = (8 – 4) = 4 ms

Waiting time of P4 = (17 – 5) = 12 ms

(ii) Average Waiting time:

$$\text{Average Waiting Time} = \frac{\text{Sum of all processes Waiting Time}}{\text{Total number of processes}}$$

$$\text{Average Waiting Time} = \frac{4+0+4+12}{4}$$

$$= 4.5 \text{ milliseconds}$$

(iii) Turnaround Time:

Turnaround Time = Finished time – Arrival Time

Process	Turnaround Time
P1	7
P2	5
P3	13
P4	18

(iv) $$\text{Average Turnaround time} = \frac{\text{Sum of turnaround time of processes}}{\text{Total number of processes}}$$

$$\text{Average Turnaround Time} = \frac{7+5+13+18}{4}$$

$$= 10.75 \text{ milliseconds}$$

Summary

- CPU Scheduling algorithms are very important in scheduling the processor amongst different processes as the CPU has to allocate to multiple processes and idle time of CPU should be minimized, it is achieved with CPU Scheduling algorithms.
- A component involved In the CPU-scheduling function is the dispatcher, which is the module that gives control of the CPU to the process selected by the short-term scheduler. It receives control in kernel mode as the result of an interrupt or system call.
- Schedulers are special system software which handles 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, Medium Term Scheduler, and Short Term Scheduler.
- In FCFS, the processor is allocated to the processes according to arrival time which process arrives first CPU will be allocated to it.
- In SJF Scheduling algorithm, the processor will be allocated to shortest burst time of processes that may lead to starvation and aging. SJF can be implemented with Non-preemptive and preemptive scheduling.
- Preemption stands for switching the processor amongst processes. No preemption stands for not switching the processor until the task is finished. In Priority Scheduling process has priority assigned to it the processor will be allocated to that process which has highest priority.
- Priority is the number which can ranges from 0 to 4999. In this case, there are chances that low process will have high waiting time.
- Round Robin(RR) scheduling technique uses time quantum or time slot where equal time slot is given to processes. It uses preemptive scheduling algorithm where equal amount of CPU time is given for all processes if the process doesn't complete in time quantum the processor will be allocated to another process. RR tries to give fare justice to all the processes.
- Multilevel Queue Scheduling algorithms are created for areas in which we classify process into different groups.
- Multilevel feedback queue allows a process to move between queues. The idea is to separate process with different CPU - Burst characteristics.

Check Your Understanding

1. Process in main memory which is ready and waiting to execute is kept in _____.
   (a) process queue (b) ready queue
   (c) schedule (d) dispatcher
2. The time interval from time of submission of a process to completion of time is called as _____.
   (a) completion time (b) turnaround time
   (c) waiting time (d) none of above
3. The unit of measurement of work of CPU is called as _____.
   (a) latency (d) delay
   (c) throughput (d) context switch
4. In priority scheduling the process is allocated with _____.
   (a) low priority (b) high priority
   (c) equal priority (d) extra priority

5. Which scheduling technique applies time quantum?
   (a) SJF (b) FCFS
   (c) Round Robin (d) Priority
6. In multilevel feedback queue algorithm _____.
   (a) a process can move to different classified ready queue
   (b) processes are not classified into different groups
   (c) classification of ready queue is not permanent
   (d) process is upgraded

Answers

1. (b)	2. (b)	3. (c)	4. (b)	5. (c)	6. (a)

Practice Questions

Q.I Answer the following questions in short:

1. What is meant by CPU scheduling?
2. What is CPU scheduler?
3. Describe the following scheduling schemes:
   (a) Preemptive scheduling,
   (b) Non-primitive scheduling.
4. What is priority?
5. Define dispatch latency.

Q.II Answer the following questions:

1. Explain scheduling criteria in detail.
2. With the help of example describe following scheduling algorithms:
   (i) FCFS (ii) RR (iii) SJF
3. With the help of diagram explain multilevel queue scheduling.
4. Describe the term priority scheduling with example.
5. What is the difference between preemptive and non-preemptive scheduling?
6. With the help of diagram describe CPU-I/O burst cycle.
7. Explain the term dispatcher in detail.
8. With suitable diagram describe multilevel feedback queue scheduling.
9. Compare RR and SJF scheduling algorithm.
10. Differentiate SJF and FCFS.
11. Calculate Average Turnaround Time and Average Waiting Time for all set of processes using SJF.

Process	Burst Time	Arrival Time
P1	4	1
P2	3	0
P3	2	2
P4	4	3
P5	1	2

12. Consider the following set of processes:

Process CPU	Burst Time (in milliseconds)
P1	30
P2	6
P3	8

Calculate the Average Waiting Time and Average Turnaround Time by using Round Robin CPU Scheduling Algorithm. (The time quantum is of 5 milliseconds).

Q.III Define terms:

1. Waiting Time
2. Turnaround Time
3. Throughput
4. Relative Delay.
5. Burst time

Previous Exam Questions

Summer 2018

1. Define Burst Time. **[2 M]**

Ans. Refer to Section 4.3.

2. What is Turn-Around Time? **[2 M]**

Ans. Refer to Section 4.4.

3. What is CPU I/O Burst Cycle? **[2 M]**

Ans. Refer to Section 4.3.

4. Explain multilevel feedback queue algorithm. **[4 M]**

Ans. Refer to Section 4.4.4.2.

5. Consider the following set of processes with the length of CPU Burst Time and Arrival Time. **[4 M]**

Process	Burst Time	Arrival Time
P1	5	1
P2	3	0
P3	2	2
P4	1	3

Calculate turnaround time, waiting time, average turnaround time, average waiting time using FCFS CPU scheduling algorithm.

Ans. Refer to Section 4.5.1.

Winter 2018

1. Consider the following set of processes: **4 M**

Processes	CPU Burst Time (in ms)	Arrival Time (in ms)
P1	28	3
P2	7	1
P3	9	2

Calculate the Average Waiting Time and Average Turn-around Time by using Round Robin CPU Scheduling Algorithm. (The time quantum is of 5 milliseconds)

Ans. Refer to Section 4.5.4.

2. What is CPU Scheduler? State the criteria of CPU scheduling. **4 M**

Ans. Refer to Section 4.1.

3. What are the differences between Preemptive and Non-preemptive Scheduling? **4 M**

Ans. Refer to Sections 4.2.3 and 4.2.5

Summer 2019

1. Define Dispatch Latency Time. **2 M**

Ans. Refer to Section 4.2.5.

2. Round Robin algorithm is non-preemptive. Comment. **2 M**

Ans. Refer to 4.5.4.

3. List and explain four criteria for computing various scheduling algorithms. **4 M**

Ans. Refer to Section 4.4.

■■■