# **Exercises-Chapter 5 CPU Scheduling**

## 1. A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given *n* processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of *n.*

## *2.* Explain the difference between preemptive and non-preemptive scheduling.

## Give an example of each type of scheduling.

|  |  |
| --- | --- |
| **Preemptive scheduling** | **Non-preemptive scheduling** |
| It has flexible nature | It is of rigid nature |
| CPU is mostly utilized in high amount | Low utilization of CPU occurs |
| CPU allocation to processes is for limited amount of time | CPU allocation to processes is until termination takes place or sifts to waiting state. |
| Processes can face interruptions | Processes do not face interruptions until the time gets over or termination occurs |
| **Example-Shortest remaining time first scheduling, etc.** | **Example- First come first serve scheduling, etc.** |

Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Nonpreemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

## 3. What events cause the CPU scheduler to run?

When a process…

* Switches from running to waiting state
* Switches from running to ready state
* Switches from a waiting to ready state
* Terminates

## 4. What is a good job mix? How does it affect the performance of the system?

## 5. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use non-preemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** |
| *P*1 | 0 | 8 |
| *P*2 | 4 | 4 |
| *P*3 | 10 | 1 |

AT -> Arrival Time

BT -> Burst Time

CT -> Completion Time

TAT -> Turn around Time

TAT = CT – AT

## What is the average turnaround time for these processes with the FCFS (First Come First Serve) scheduling algorithm?

|  |  |  |
| --- | --- | --- |
| P1 | P2 | P3 |

0 8 12 13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **AT** | **BT** | **CT** | **TAT** |
| P1 | 0 | 8 | 8 | 8 |
| P2 | 4 | 4 | 12 | 8 |
| P3 | 10 | 1 | 13 | 3 |

## What is the average turnaround time for these processes with the SJF (Shortest-Job-First) (non-preemtive) scheduling algorithm?

|  |  |  |
| --- | --- | --- |
| P1 | P3 | P2 |

0 8 9 13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **AT** | **BT** | **CT** | **TAT** |
| P1 | 0 | 8 | 8 | 8 |
| P2 | 4 | 4 | 13 | 9 |
| P3 | 10 | 1 | 9 | 1 |

## The SJF (Shortest-Job-First) algorithm is supposed to improve performance, but notice that we chose to run process *P*1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 10 units and then SJF (Shortest-Job-First) scheduling is used. Remember that processes *P*1 and *P*2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.

|  |  |  |
| --- | --- | --- |
| P3 | P2 | P1 |

10 11 15 23

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **AT** | **BT** | **CT** | **TAT** |
| P1 | 0 | 8 | 23 | 23 |
| P2 | 4 | 4 | 15 | 11 |
| P3 | 10 | 1 | 11 | 1 |

## 6. Compute the waiting time, turnaround time and response time for each process under the following scheduling schemes for the process list below.

AT -> Arrival Time

BT -> Burst Time

Entrance Time -> when process enters the CPU

CT -> Completion Time

RT -> Response Time

WT -> Wait Time

TAT -> Turn around Time

TAT = CT – AT

WT = TAT – BT

RT = Entrance Time – AT

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority |
| P1 | 0 | 10 | 3 |
| P2 | 2 | 1 | 1 |
| P3 | 3 | 5 | 2 |
| P4 | 5 | 8 | 2 |
| P5 | 7 | 6 | 3 |

## SJF scheduling (Shortest Job First) (Non-preemptive)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P5 | P4 |

## 0 10 11 16 22 30

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **AT** | **BT** | **Entrance Time** | **CT** | **RT** | **WT** | **TAT** |
| P1 | 0 | 10 | 0 | 10 | 0 | 0 | 10 |
| P2 | 2 | 1 | 10 | 11 | 8 | 8 | 9 |
| P3 | 3 | 5 | 11 | 16 | 8 | 8 | 13 |
| P4 | 5 | 8 | 22 | 30 | 17 | 17 | 25 |
| P5 | 7 | 6 | 16 | 22 | 9 | 9 | 15 |

## SRTF scheduling (Shortest Remaining Time First) (Preemptive SJF)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P3 | P5 | P1 | P4 |

## 0 2 3 5 8 14 22 30

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **AT** | **BT** | **Entrance Time** | **CT** | **RT** | **WT** | **TAT** |
| P1 | 0 | 10 | 0 | 22 | 0 | 12 | 22 |
| P2 | 2 | 1 | 2 | 3 | 0 | 0 | 1 |
| P3 | 3 | 5 | 3 | 8 | 0 | 0 | 5 |
| P4 | 5 | 8 | 22 | 30 | 17 | 17 | 25 |
| P5 | 7 | 6 | 8 | 14 | 1 | 1 | 7 |

## FCFS scheduling (First Come First Served)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 |

## 0 10 11 16 24 30

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **AT** | **BT** | **Entrance Time** | **CT** | **RT** | **WT** | **TAT** |
| P1 | 0 | 10 | 0 | 10 | 0 | 0 | 10 |
| P2 | 2 | 1 | 10 | 11 | 8 | 8 | 9 |
| P3 | 3 | 5 | 11 | 16 | 8 | 8 | 13 |
| P4 | 5 | 8 | 16 | 24 | 11 | 11 | 19 |
| P5 | 7 | 6 | 24 | 30 | 17 | 17 | 23 |

## Priority scheduling (with preemption)

## 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P1 | P5 |

## 0 2 3 8 16 24 30

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **AT** | **BT** | **Priority** | **Entrance Time** | **CT** | **RT** | **WT** | **TAT** |
| P1 | 0 | 10 | 3 | 0 | 24 | 0 | 14 | 24 |
| P2 | 2 | 1 | 1 | 2 | 3 | 0 | 0 | 1 |
| P3 | 3 | 5 | 2 | 3 | 8 | 0 | 0 | 5 |
| P4 | 5 | 8 | 2 | 8 | 16 | 3 | 3 | 11 |
| P5 | 7 | 6 | 3 | 24 | 30 | 17 | 17 | 23 |

## RR scheduling with time slice of 2 units (Round Robin)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P1 | P3 | P4 | P1 | P5 | P3 |

## 0 2 3 5 7 9 11 13 15

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P4 | P1 | P5 | P3 | P4 | P1 | P5 | P4 |

## 15 17 19 21 22 24 26 28 30

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | **AT** | **BT** | **Entrance Time** | **CT** | **RT** | **WT** | **TAT** |
| P1 | 0 | 10 | 0 | 26 | 0 | 16 | 26 |
| P2 | 2 | 1 | 2 | 3 | 0 | 0 | 1 |
| P3 | 3 | 5 | 5 | 22 | 2 | 14 | 19 |
| P4 | 5 | 8 | 7 | 30 | 2 | 17 | 25 |
| P5 | 7 | 6 | 11 | 28 | 4 | 15 | 21 |

7. Name one advantage and disadvantage of each of the following scheduling schemes.

## SRTF scheduling (Shortest Remaining Time First)

* + 1. Advantages –
       1. Shortest jobs are favored.
       2. It is provably optimal, in that it gives the minimum average waiting time for a given set of processes.
    2. Disadvantages –
       1. SJF may cause starvation, if shorter processes keep coming. This problem is solved by aging.
       2. It cannot be implemented at the level of short term CPU scheduling.

## FCFS scheduling (First Come First Served)

* + 1. Advantages –
       1. It is simple and easy to understand.
    2. Disadvantages –
       1. The process with less execution time suffer i.e. waiting time is often quite long.
       2. Favors CPU Bound process then I/O bound process.
       3. Here, first process will get the CPU first, other processes can get CPU only after the current process has finished it’s execution. Now, suppose the first process has large burst time, and other processes have less burst time, then the processes will have to wait more unnecessarily, this will result in *more average waiting time*, i.e., convey effect.
       4. This effect results in lower CPU and device utilization.
       5. FCFS algorithm is particularly troublesome for time-sharing systems, where it is important that each user get a share of the CPU at regular intervals.

## Priority scheduling (with preemption)

1. Advantages –
   1. This provides a good mechanism where the relative importance of each process maybe precisely defined.
2. Disadvantages –
   1. If high priority processes use up a lot of CPU time, lower priority processes may starve and be postponed indefinitely.The situation where a process never gets scheduled to run is called starvation.
   2. Another problem is deciding which process gets which priority level assigned to it.

## 

## RR scheduling with time slice (Round Robin)

* + 1. Advantages –
       1. Every process gets an equal share of the CPU.
       2. RR is cyclic in nature, so there is no starvation.
    2. Disadvantages –
       1. Setting the quantum too short, increases the overhead and lowers the CPU efficiency, but setting it too long may cause poor response to short processes.
       2. Average waiting time under the RR policy is often long.

## 8. Consider the following priority scheduling scheme. Larger  priority numbers imply higher priorities. When the process is waiting for CPU, its priority changes at rate x. When it is running, its priority changes at the rate y. All processes have priority 0 to start with.

## a) What is the algorithm we get if y > x > 0? Explain.

FCFS

This would result in algorithm FCFS. When a process enters a waiting queue ti will have the priority of 0. The longer it stays in the waiting queue the larger priority of that process will be. From this we conclude that the process which got first in waiting queue would have the largest priority. When a process is running its priority will increase by a larger factor so it won’t be interrupted until it is finished.

## b) What is the algorithm we get if x > y > 0? Explain.

LCFS

Last come first serve – a process here is rewarded the less time it spends in queue. When a process enters a waiting queue it will have priority of 0. The longer it stays in the waiting queue the smaller the priority of the process will be.

## 9. Describe the differences between multilevel queues and multilevel feedback queues? Can any of these cause starvation? Explain

In Multilevel queue (MLQ) the processes are permanently assigned to one queue based on their memory size, process priority or process type. In Multilevel Feedback queue (MLFQ) it allows a process to move between the queues, according to the characteristics of their CPU burst.

Starvation is the problem that occurs when high priority processes keep executing and low priority processes get blocked for indefinite time. In heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU.

MLQ and MLFQ prevent starvation by moving a process that waits too long for lower priority queue to the higher priority queue.

## 10. Explain the differences in the degree to which the following algorithms discriminate in favor of a short process: (heavily favors short process, somewhat favors short process, heavily discriminates against short process)

## a) FCFS

* In this algorithm the process that requires the CPU first is allotted the CPU first and its implementation can be easily maintained using FIFO queue.
* When a CPU is free, the process is allotted CPU and it will continue holding CPU fill it is terminated or requests I/O devices. So process waiting for CPU will have to wait for its execution.
* Thus waiting time is large if a larger process executes before a shorter process.
* **Thus we can say, FCFS discriminates against short job since any short job arriving after long job will have a longer waiting time.**

## b) RR

* This algorithm is designed for time sharing system.
* A small unit of time called time quantum or time slice is defined and each process is switched in and out of CPU depending on this time quantum value.
* The time quantum is generally from 10 ms to 100 ms in length.
* Here a process is not allotted CPU more than 1 time quantum.
* The performance of RR algorithm depends heavily on the size of time quantum, if time quantum Is very long time RR policy behave same as FCFS and if the time quantum is very small (say 1, ms) then RR approach is called as processor sharing and creates the appearance that each process has its own processor running at 1/n the speed of real processor.
* **We can say that, RR – treats all job equally ( given them equal burst time of CPU) so short jb will be able to leave the system faster since they will finish first.**

## c)  Multilevel Feedback queues

## 

* It allow the process to move between queues, the idea is to separate processes according to its characteristics of its burst time.
* If the process uses too much CPU time it will be moved to lower priority queue due to this scheme all I/O bounded and inter ache process are in higher priority queue.
* In addition, to this a process which wait too long in a lower priority queue may be moved to a higher priority queue and this form of aging prevents saturation.
* **Thus we can say, multilevel feedback queues work similar to the RR algorithm so they discriminates favorably towards short job.**

## 11. What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?

## Processes that need more frequent servicing, for instance, interactive processes such as editors, can be in a queue with a small time quantum. Processes with no need for frequent servicing can be in a queue with a larger quantum, requiring fewer context switches to complete the processing, and thus making more efficient use of the computer.

## 12. Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on.

## These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What, if any, relation holds between the following pairs of algorithm sets?

## a. Priority and SJF

The shortest job has the highest priority.

## b. Multilevel feedback queues and FCFS

The lowest level of MLFQ is FCFS

## c. Priority and FCFS

FCFS gives the highest priority to the job having been in existence the longest.

## d. RR and SJF

None

## 13. Suppose that a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

It will favor the I/O-bound programs because of the relatively short CPU burst request by them; however, the CPU-bound programs will not starve because the I/O-bound programs will relinquish the CPU relatively often to do their I/O.

## 14. The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: the higher the number, the lower the priority. The scheduler recalculates process priorities once per second using the following function:

## where base = 60 and *recent CPU usage* refers to a value indicating how often a process has used the CPU since priorities were last recalculated. Assume that recent CPU usage for process *P*1 is 40, for process *P*2 is 18 and for process *P*3 is 10.

## What will be the new priorities for these three processes when priorities are recalculated? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process?

The priorities assigned to the processes are **80, 69, and 65** respectively. **The scheduler lowers the relative priority of CPU-bound processes.**