JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

CPU SCHEDULING ALGORITHMS: A COMPREHENSIVE STUDY OF PRE-EMPTIVE SCHEDULING ALGORITHMS

Abhishek Kumar Singh
School of Computer Science and Engineering, Lovely Professional University, abhiaks.lpu@gmail.com, +918003889970

ABSTRACT
CPU Scheduling is the most basic task that a CPU has to finish in order to be a fast processor. A processor has to be in sync to meet the Operating System design goals. The focus is to never let the CPU remain idle because that is a wastage of resources and according to Operating system design principles, wastage of resources should be minimal. There have been dozens of CPU scheduling algorithms already in place, we will look at few of them in detail here. This paper talks about some of the most efficient CPU scheduling algorithms and help them explain in a detailed manner with the help of Gantt charts and figures.

INTRODUCTION

I. **CPU SCHEDULING:** CPU Scheduling is a process of determining which process will occupy CPU for execution while another process is on hold. A process in the most basic terms is a predefined sequence of instructions and its execution must take place in that predefined order. In a single processor system, only one process can execute at a one given time. [(1)] The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization. [(1)]

Simply put, CPU Scheduling is a way of putting processes to CPU from Ready queue. Before going further, we should be aware of different states of a process. CPU Scheduler also known as short term scheduler does this task of selecting the processes from waiting state and putting it into a state which is ready for execution. It selects one of the processes in memory that is ready for execution. The main purposes of scheduling algorithms are to minimize resource starvation and to ensure fairness amongst the parties utilizing the resources. [(9)]
The minimum number of states a process can have is five. The process in CPU passes through different states in its entire span from creation to completion/suspend and it can be in any of the below states at a given point of time.

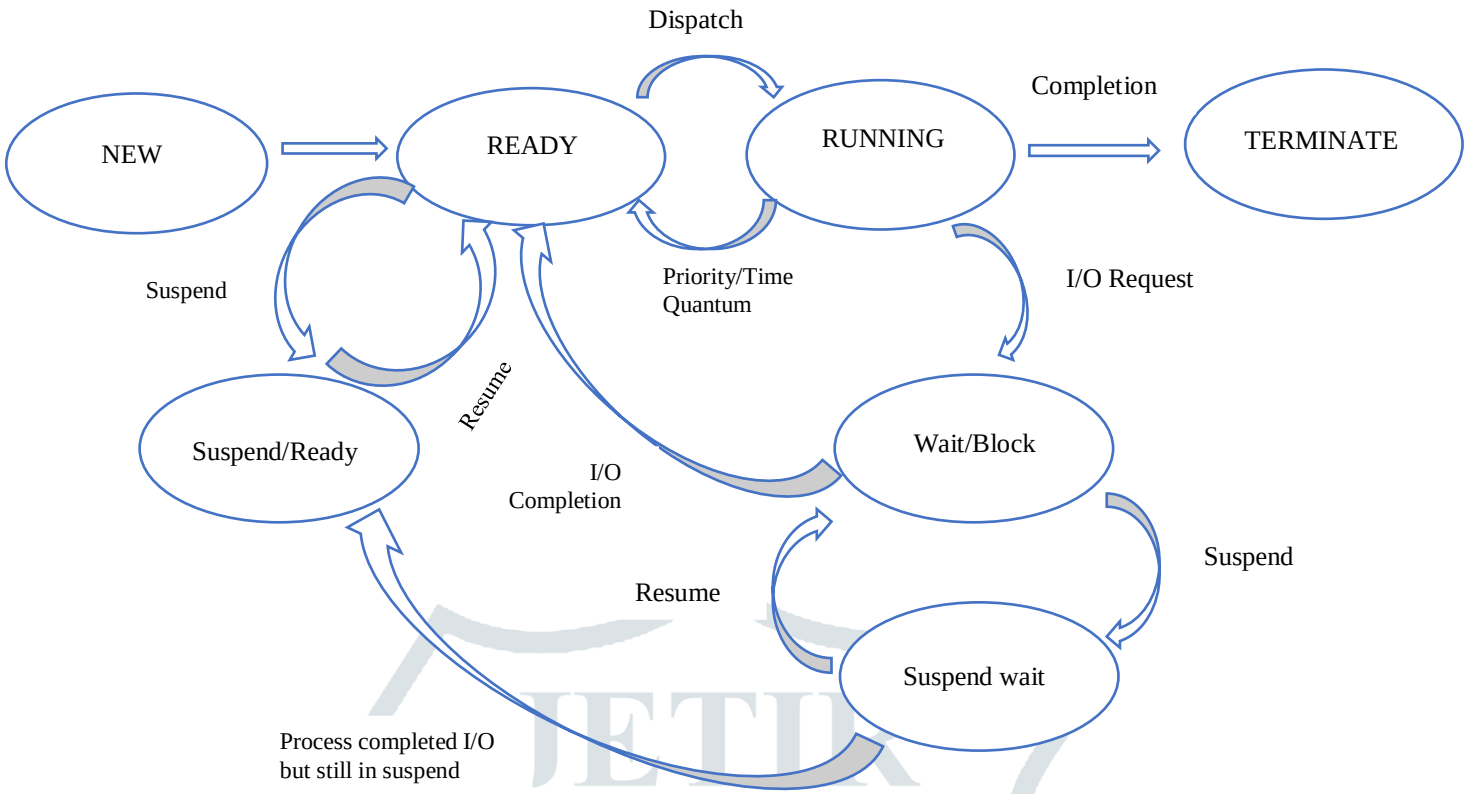


Figure 1: Different States of a process

II. STATES OF A PROCESS

New: A process which is just created and ready to be put in Ready Queue in primary memory. A new process always resides in the secondary memory.

Ready: A process in ready state, waits for the CPU to be assigned. The short-term scheduler, or CPU scheduler, selects from among the processes that are ready to execute and allocates the CPU to one of them. (4)

Running: A process being executed by CPU in main memory is said to be in running state. Higher the number of processors, more the process can be picked and assigned to CPU at once.

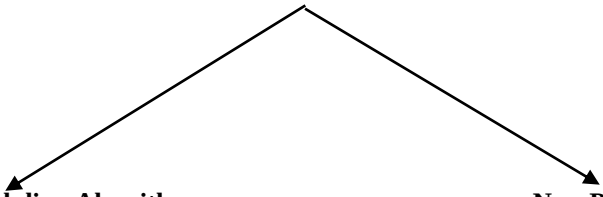
Block/Wait: A process from Running State can be put in Block/Wait queue if the process in running state needs a certain resource or any input from user during its course of execution. Block/Wait queue resides in main memory. CPU then goes back to execute other processes until the I/O is provided.

Complete/Terminate: When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.

Suspend ready: A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state. (3)

Suspend Wait: Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory. Since it is already waiting for some resource to get available hence it is better if it waits in the secondary memory and make room for the higher priority process. These processes complete their execution once the main memory gets available and their wait is finished.

III. CPU SCHEDLUNG ALGORITHMS: A BIGGER PICTURE



Pre-Emptive CPU Scheduling Algorithms

-SRTF (Shortest Remaining Time First)
-LRTF (Longest Remaining Time First)
-RR (Round Robin)
-Priority Based Scheduling Algorithms

Non-Pre-Emptive CPU Scheduling Algorithms

-FCFS (First Come First Serve)
-SJF (Shortest Job First)
-LJF (Longest Job First)
-HRRN (Highest Response Ratio Next)

IV. PRE-EMPTIVE CPU SCHEDULING ALGORITHMS

1. Shortest Remaining Time First

As the name states, this scheduling algorithm assign CPU to processes having the least burst time first. It is the pre-emptive version of Shortest Job First. There might be a situation when the burst time of two process is same, then the tie is broken using First Come First Serve Algorithm. No pre-emption will be performed after all the processes are available in ready queue. When a process is removed from execution and the next process is scheduled, the context of the process is saved in the **Process Control Block** (PCB). On the next execution of this process, this PCB is accessed to reload necessary information like the instructions from where to resume execution and other necessary instructions. [(1)] Advantages of SRTF scheduling includes minimum average waiting time and the disadvantage is starvation, there may be a situation where processes with shorter burst time keeps on arriving and a process with larger burst time may wait for too long to get allocated to CPU.

Given a set of processes with the following Burst time and Arrival time.

Process Number	Arrival Time (AT) (in seconds)	Burst Time (BT) (in seconds)
P0	0	5
P1	1	3
P2	2	4
P3	4	1

The CPU checks the Burst Time of every process in Ready State after every unit time.

At Arrival Time=0

P0 is ready in the queue and the CPU will execute it for 1 second and check if there is any other process with the lower burst time is available, if there is available process with the lower burst time, CPU will perform a Context Switching and the process with the lower burst time will start executing.

P0 will execute for 1 second, time remaining to complete its execution is now 5-1=4 seconds.

P0	

Time *0* *1*

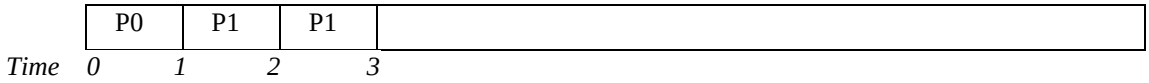
At Arrival Time = 1

Clearly, P1 is available in the queue with burst time of 3 seconds, which is lower than all the existing processes available in ready queue so context switching will happen and CPU will start executing P1 for unit time. Time remaining to complete process P1 execution is now 3-1=2 seconds.

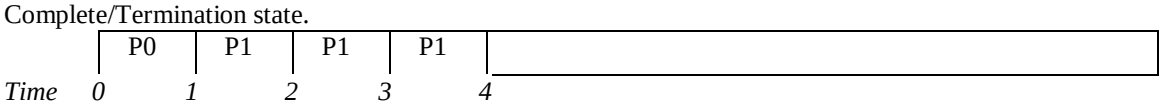
P0	P1	

Time *0* *1* *2*

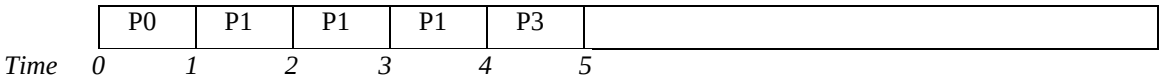
At Arrival Time = 2 CPU will check again for possible processes with lower burst time. Clearly, there are 3 Processes namely P0, P1 and P2 waiting to be executed but P1 is having the lowest burst time among all, hence Process P1 will resume execution for another 1 second. Total Time remaining to complete Process P1 execution is now 2-1=1 seconds.



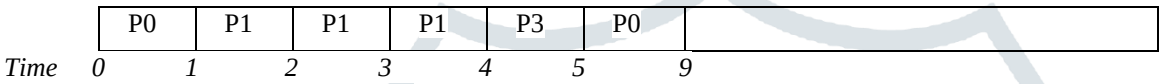
At arrival Time = 3 seconds, P0, P1 and P2 are waiting to be executed but P1 has the least burst time therefore it will be picked by short-term scheduler and will execute for unit time. P1 now has been executed completely and it will be put into Complete/Termination state.



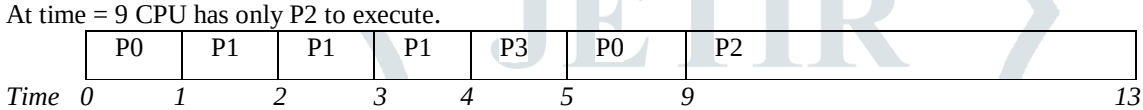
At arrival time of 4 seconds, P0, P2 and P3 are in waiting state having burst time of 4, 4 and 1 seconds respectively. Clearly P3 will be picked up by scheduler and it will be executed for 1 second thereafter, it will be put into the termination state.



Now, P1 and P3 are executed, only P0 and P2 are remaining to be executed with the burst time of 4 seconds each. CPU scheduler will break this tie of burst time on the basis of FCFS, i.e. First Come First Serve. Process P0 came at Arrival time 0 hence it will be picked to execute by CPU. Since there are no more processes waiting hence it will be executed completely.



At time = 9 CPU has only P2 to execute.



Analysing Turn Around Time, Waiting Time and Response Time

TAT: Turn Around Time is the total time a process spent in CPU until it gets to completion state. It can also be considered as the sum of the time periods spent waiting to get into memory or ready queue, execution on CPU and executing input/output. [(3)] Mathematically it is completion Time minus arrival time.

CT: The Time at which the process enters into the completion state or the time at which the process completes its execution, is called completion time. [(5)] Obvious to see completion time of a process from Gantt charts, the time at which a process is seen the last time is its completion time.

WT: Waiting Time is the total time a process spent in waiting in the ready queue until it gets allocated to CPU. Mathematically it is TAT minus Burst Time.

RT: Response Time is the total time a process spent until the time it gets picked for execution for the very first time.

Process	TAT (in seconds)	WT (in seconds)	RT (in seconds)
P0	9	4	0
P1	3	0	0
P2	11	7	7
P3	1	0	0

Table: *Table showing Turn Around Time, Waiting Time and Response Time for each process in Shortest Remaining Time First Scheduling.*

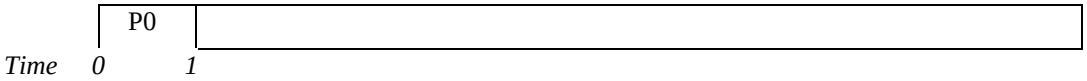
2. Longest Remaining Time First

Herein, the process is allocated CPU on the basis of largest burst time. After every quantum of time, CPU checks if there is any process having the longer burst time than the currently process in execution, if there is any, context switching happens and CPU is allocated to that process, if there is none, CPU keeps executing the same process for another time quantum. The cycle keeps on repeating till there is no more processes left in CPU to be executed. Advantages of LRTF is easy implementation while disadvantages include a higher waiting time and sometimes process with lower burst time may have to wait a little too long to get assigned to CPU.

Given a set of processes with the following burst time and arrival time.

Process Number	Arrival Time (AT) (in seconds)	Burst Time (BT) (in seconds)
P0	0	2
P1	1	5
P2	4	3
P3	5	2

At arrival time = 0 there is only P0 in the CPU, hence P0 will be executed for unit time. Time left for P0 to complete its execution is now 2-1 = 1

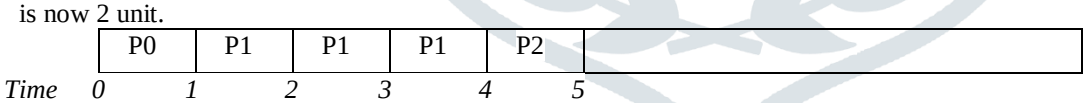


At arrival time = 1, Process P1 arrives with the larger burst time, hence according to Burst time criteria, CPU will be allocated to P1 and the current progress of P0 will be save in Process Control Block. P1 shall execute for unit time quantum. Time left for P1 to complete execution is now 5-1=4 seconds

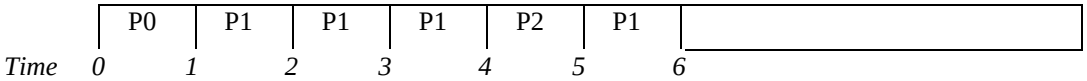
At Arrival time = 2 CPU again checks for new possible process in Ready queue, but there is none so CPU will be allocated to the process with the existing largest burst time. In this case, CPU will continue execution of P1 as it has the largest burst time among all the available processes in ready queue. Time remaining for P1 to finish execution is now 3 seconds.

At arrival time = 3 CPU checks again but still there is no new process, so it will continue execution of P1. Now total time left for P1 to complete execution is 2 seconds.

At arrival time = 4, CPU finds that a new process P2 having burst time of 3 seconds is available, note that this process has the largest burst time of all the available processes now. Therefore, it will start executing. Time remaining for P2 to finish execution is now 2 unit.

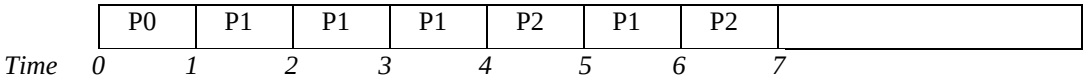


At arrival time = 5, CPU checks again for the possible and finds out there is P3 waiting to be executed with the burst time of 2 seconds. Note that now all the processes are in ready state, and P1, P2 and P3 each has a burst time of 2, this tie will be broken up by First Come First Serve criteria and the firstly arrived process shall execute for another time quantum. In this case, P1 arrived first, so it will be allocated CPU for another second. Now P1 has 1 unit of burst time left to its complete execution.



At time = 6 P0 and P1 has 1second left to complete execution while P2 and P3 has burst time of 2 seconds each. P2 will be executed for another time quantum as it was arrived before P3

P2 now has 1 unit of time left to complete execution along with P0, P1 and P2



At arrival time of 7, P3 will be allocated CPU and it will execute for one time quantum.

Now time left for P3 is 1second to finish execution.

P0	P1	P1	P1	P2	P1	P2	P3	

Time 0 1 2 3 4 5 6 7 8

Now each process has one time quantum left to complete execution, hence they will be allocated CPU on FCFS basis, P0->P1->P2->P3 and after executing fully they shall be put into Termination state.

P0	P1	P1	P1	P2	P1	P2	P3	P0	P1	P2	P3

Time 0 1 2 3 4 5 6 7 8 9 10 11 12

Lets' have a look at their Turnaround Time, Waiting Time and Response Time

Process	**TAT (in seconds)**	**WT (in seconds)**	**RT (in seconds)**
P0	9	7	0
P1	9	4	0
P2	7	4	0
P3	7	5	2

Table: *Table showing Turn Around Time, Waiting Time and Response Time for each process in Longest Remaining Time First Scheduling.*

3. Round Robin

Round Robin is a CPU scheduling algorithm where a process only executes for a predefined time quantum. Thereafter, the process is made to go back in ready state and the next process from the ready state is picked up to be executed for the time quantum. It is basically the pre-emptive version of First Come First Serve CPU Scheduling Algorithm and can actually be implemented in most of the operating systems. [6]Advantages includes no fairness to any process as every process gets equal share of CPU. [6] Disadvantages being larger waiting and response time along with fair share of context switches, thus giving more overhead to CPU. Important to note, there is a context switching only after every time quantum in Round Robin.

Given a set of processes with the following burst time and arrival time, and time quantum of 2 seconds.

Process Number	**Arrival Time (AT) (in seconds)**	**Burst Time (BT) (in seconds)**
P0	0	5
P1	1	4
P2	2	2
P3	4	1

Ready Queue

P0	

Since Time Quantum is given as 2, the process from ready state will be picked up and shall run for 2 seconds and it will be sent to either termination state or back to ready state, depending upon its burst time.

Running Queue

P0	

Time 0 2

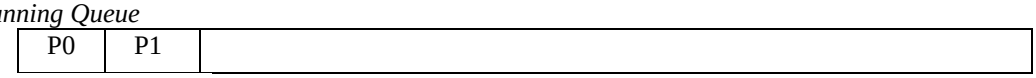
At time = 2, P1 and P2 are also ready and context switching happens so P0 from running state is sent back to ready queue as it still has 3 seconds of burst time. Updated Ready queue is below after context switching is over.
Ready queue.

Ready Queue

P1	P2	P0	

Updated order of processes in running state should be this.

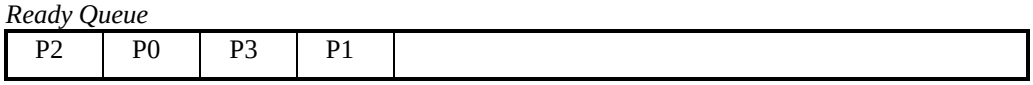
Running Queue



Time 0 2 4

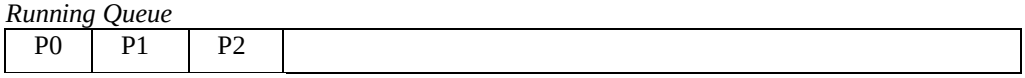
At time = 4 context switching takes place and P1 is sent back to ready state, it still has 2 seconds of burst time left after being processed for one time quantum. All the processes are now available in ready state.
Ready queue

Ready Queue



Processes in running queue have the below order of execution.

Running Queue



Time 0 2 4 6

At time = 6, another context switching takes place and the running process (P2) will be sent to termination/complete queue as its remaining burst time of 2 seconds is over.

Ready state

P0 will be picked up and it will run for another time quantum. It has 3 seconds of burst time left.

Running State

Time 0 2 4 6 8

P0 still has 1 second of burst time until it is fully executed, hence it will be put in ready queue once again.

Ready Queue

P3 will be put in Running State and it should run for one time quantum but it has a burst time of 1 which is less than time quantum, hence it will run and put to termination queue after its burst time is over.

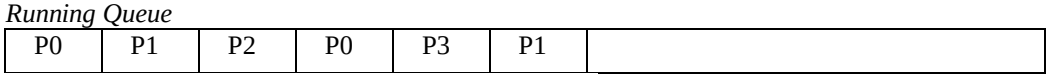
Running Queue

Time 0 2 4 6 8 9

Ready queue

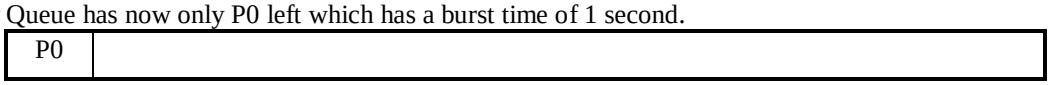
At time = 9, P1 will run for 2 seconds and it will be put into termination queue after it is processed.

Running Queue



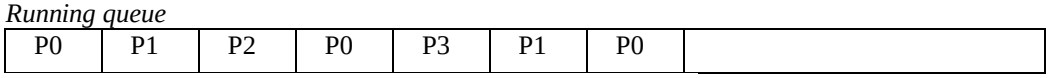
Time 0 2 4 6 8 9 11

Ready Queue has now only P0 left which has a burst time of 1 second.



P0 will run for 1 second before it is finally put to terminations queue.

Running queue



Time 0 2 4 6 8 9 11 12

Lets' have a look at their TAT, Waiting Time and Response Time.

Process	TAT (in seconds)	WT (in seconds)	RT (in seconds)
P0	12	7	0
P1	10	6	1
P2	4	2	2
P3	5	4	4

Table: *Table showing Turn Around Time, Waiting Time and Response Time for each process in Round Robin Scheduling.*

4. Priority Based CPU Scheduling

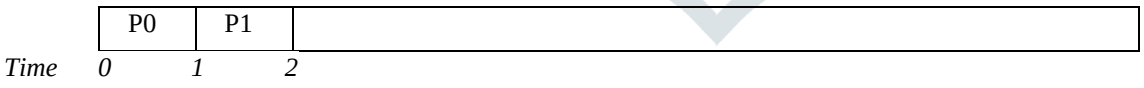
Here the criteria for processes to allocate CPU is the priority, each process has its own set priority and depending upon that it is served CPU. Context switching takes place when there is another process in the ready queue with the higher priority than the ongoing process in execution. In real time systems, priority of any process can be set according to the deadline of that process because in real time systems, main aim is to meet the deadline. [8] Advantages include easy implementation and disadvantages include starvation. Starvation is a phenomenon where process with lower priority ends up waiting too long for the CPU, while the process with higher priority is being processed.

Given a set of processes with their arrival time, burst time and priority. It is worthwhile noting here that the higher the number, greater the priority is.

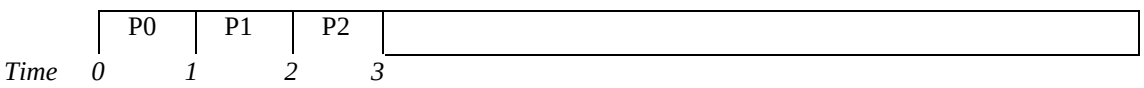
Process Number	Arrival Time (AT) (in seconds)	Burst Time (BT) (in seconds)	Priority
P0	0	5	1
P1	1	4	2
P2	2	2	3
P3	4	1	4

At arrival time=0 there is only one process in ready queue, so it will be picked up irrespective of its priority.

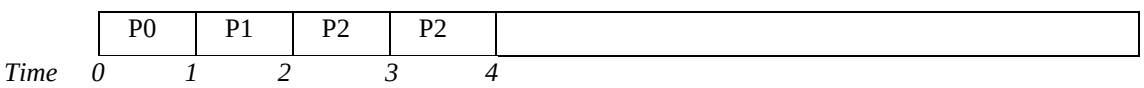
At T=1, context switching takes place and P1 is picked up to process in CPU as it has higher priority. P0 still has 4 seconds of burst time remaining.



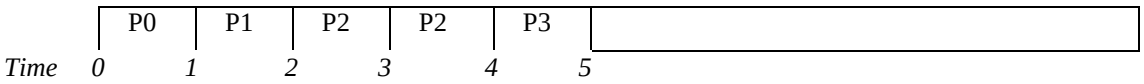
At arrival time =2 another context switching happens and now P2 is picked up as it has higher priority than all the available processes in ready queue.



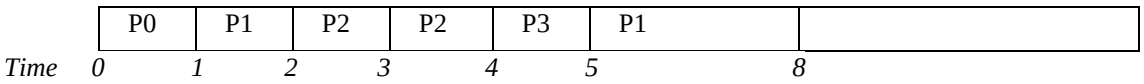
At arrival time=3, P2 still has the highest priority, so it will continue to execute for another second of time. It will be put to termination queue as its burst time is over.



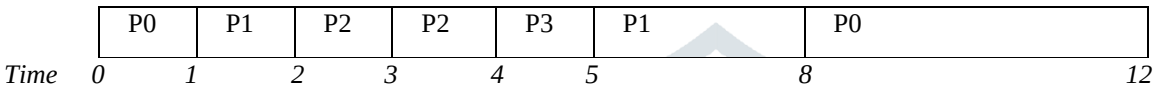
At arrival time = 4, P0, P1 and P3 are available in ready queue and out of all P3 has the highest priority so it will be allocated CPU until the time it is processed fully. It has a burst time of 1 second, hence it will be transferred to termination queue after allocating CPU once.



Now P0 and P1 are available in the ready queue, P1 has the higher priority and has a burst time of 3 second left. There will no context switching as no new process is available that has higher priority. Hence it will continue executing until it is processed to termination queue.



At time = 8, only P0 is available having burst time of 4 seconds remaining, hence it will execute and will be eventually be put in termination queue.



At time = 12 all processes are over and the order of order of putting the process in termination state was:
P2->P3->P1->P0

Analysing TAT and WT in Priority Based Scheduling

Process	TAT (in seconds)	WT (in seconds)
P0	12	7
P1	7	3
P2	2	0
P3	1	0

Table: *Table showing Turn Around Time and Waiting Time for each process in Priority Based Scheduling.*

CONCLUSION

CPU Scheduling is a process by which processes gets assigned to CPU with the help of scheduler or dispatcher.
According to the design principles of operating system, wastage of resources in any means is not considered a good sign for that operating system. Operating systems are made to maximise CPU utilization and minimise waiting time of processes, i.e., to process jobs rapidly. Now, jobs can be scheduled to process by CPU in a variety of ways, some of them have been discussed earlier in this paper. It is worthwhile noting here that there is no switching of jobs in non- pre-emptive scheduling, once a job is assigned to CPU it will continue to execute until it terminated or is moved to waiting state. While the scenario is opposite in Pre-emptive CPU Scheduling Algorithms. This the factor which brings down the waiting time in case of Pre-emptive Scheduling. In every Pre-emptive algorithm discussed in this paper, the time involved in context switching is not taken into consideration. There is a minute time taken by CPU for context switching which also increases overhead on CPU. Higher the context switched, higher is the CPU overheard. While FCFS may seem a fair algorithm because processes are scheduled in the order, they arrive but this can lead to a wait state for the process which need to process quickly. Everyday computers do not use any of the above discussed scheduling algorithms alone, however, they use a combination of scheduling algorithms to process jobs at a rapid rate. There is no “one size fits all” criteria in case of CPU scheduling algorithms.

ABBREVIATIONS USED

CPU: Central Processing Unit
AT: Arrival Time
TAT: Turn Around Time
WT: Waiting Time
BT: Burst Time
RT: Response Time
PCB: Process Control Block

REFERENCES

(1) Goel, Neetu, and R. B. Garg. "A comparative study of cpu scheduling algorithms." *arXiv preprint arXiv:1307.4165* (2013).
(2) https://www.codingninjas.com/codestudio/library/srtf-scheduling

(3) https://www.javatpoint.com/os-process-states
(4) https://www.tutorialspoint.com/what-are-process-states
(5) Silberschatz, P.B. Galvin and. Gagne (2012), Operating System Concepts, 8th edition, Wiley India
(6) https://www.javatpoint.com/os-round-robin-scheduling-algorithm
(7) https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/
(8) Himanshi Arora, Deepanshu Arora, Bagish Goel and Parita Jain. Article: An Improved CPU Scheduling Algorithm. International Journal of Applied Information Systems 6(6):7-9, December 2013
(9) Saraswathi Seemakuthi1, Venkat Alekhya.Siriki2 , Dr. E.Laxmi Lydia3. "A Review on Various Scheduling Algorithms." International Journal of Scientific & Engineering Research, Volume 6, Issue 12, December-2015