Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 11, November 2013, pg.310 – 318

RESEARCH ARTICLE

Comparative Study of Different CPU Scheduling Algorithms

Miss. Jayashree S. Somani[1], Miss. Pooja K. Chhatwani[2]

[1]*Information Technology Department, Dr. N. P. Hirani Institute of Polytechnic, Pusad, India*

[2]*Information Technology Department, Dr. N. P. Hirani Institute of Polytechnic, Pusad, India*

[1] jayashree.somani@gmail.com ; [2] pooja22.chhatwani@gmail.com

***Abstract*— In Multiprogramming operating system, CPU scheduling plays a very important role. CPU scheduling deals with the problem that to which process the CPU should be allocated. For scheduling the processes in different ways, there are many different scheduling algorithms.**

This article deals with various scheduling algorithms like First-Come-First-Serve (FCFS) scheduling algorithm, Shortest Job First (SJF) scheduling algorithm, Priority scheduling algorithm, Round Robin (R-R) scheduling algorithm, Multilevel Queue scheduling algorithm and Multilevel Feedback Queue scheduling algorithm.

FCFS scheduling algorithm is based on First-in-first-out concept and is non-pre-emptive scheduling algorithm. It is generally suitable for batch systems. SJF scheduling algorithm can be either pre-emptive or non-pre-emptive and it is based on burst time of the processes. Priority scheduling algorithm is necessarily a form of pre-emptive scheduling algorithm and it is based on the priorities given to the processes. R-R scheduling algorithm is also a pre-emptive scheduling algorithm and is based on the given time-quantum. It is generally suitable for time sharing systems. In multilevel queue scheduling, when the process enters in the system, it is permanently assigned to a queue depending upon its nature. It is pre-emptive in nature. Multilevel feedback queue is also pre-emptive and it allows the processes to move between queues.

***Keywords*— Scheduling algorithm; First-Come-First-Serve scheduling; Shortest Job First scheduling; Priority scheduling; Round Robin scheduling; Multilevel Queue scheduling; Multilevel Feedback Queue scheduling**

I. INTRODUCTION

The main objective of multiprogramming system is to load many processes in the main memory where they reside in the ready queues making link lists. A process is simply program on execution. A Process is an instance of a computer program that is being executed. It contains the program code and its current activity. CPU Scheduling is the basis of multiprogrammed operating system. By switching the CPU among processes, the operating system can make the computer more productive.

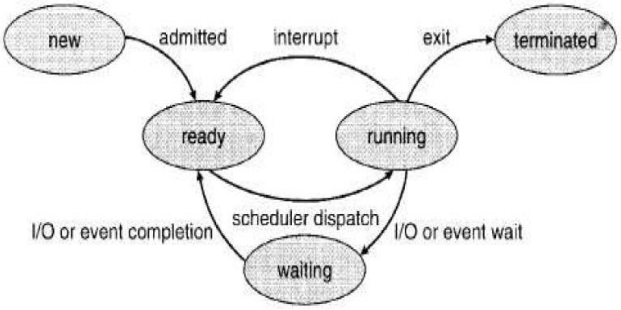


Fig 1: Process State Diagram

There are five states in which a process stay. These states are New, Ready, Run, Wait and Terminate. The Operating System helps in determining which process is allocated to CPU with the help of CPU schedulers. The algorithms concern with CPU schedulers is known as CPU Scheduling algorithms. Thus, scheduling is nothing but the arrangement of processes in ready queue such that the CPU utilization and throughput of the system is as maximum as possible.

II. SCHEDULING CRITERIA

For judging the various scheduling algorithm, there are many scheduling criteria:

A. CPU utilization

It keeps the CPU as busy as possible. It must have maximum value.

B. Throughput

The number of processes that complete their execution per unit of time. It must have maximum value.

C. Turnaround time

It is the amount of time to execute a particular process. It must have minimum value.

D. Waiting time

It is the amount of time a process has been waiting in the ready queue or in the waiting state. It must have minimum value.

E. Response time

It is the time from the submission of the request until the first response is produced. It is the time it takes to start responding, not the time it takes to output the response.

III. SCHEDULING ALGORITHMS

Basically the scheduling algorithms are divided into two categories as Pre-emptive and Non-pre-emptive scheduling algorithms.

In non-pre-emptive, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or switching to the waiting state.

In pre-emptive, CPU allocated to a process is switched if another process of higher priority is scheduled. In this case, the currently running process is interrupted and moved to the ready state by the operating system.

Following are the various scheduling algorithms:

A. First-Come- First- Serve (FCFS) Scheduling

In this, the process that requests the CPU first is allocated the CPU first. Its implementation is easily managed with FIFO queue. When the CPU is free, it is allocated to the process which is at the head of the queue. It is a non-pre-emptive scheduling algorithm.

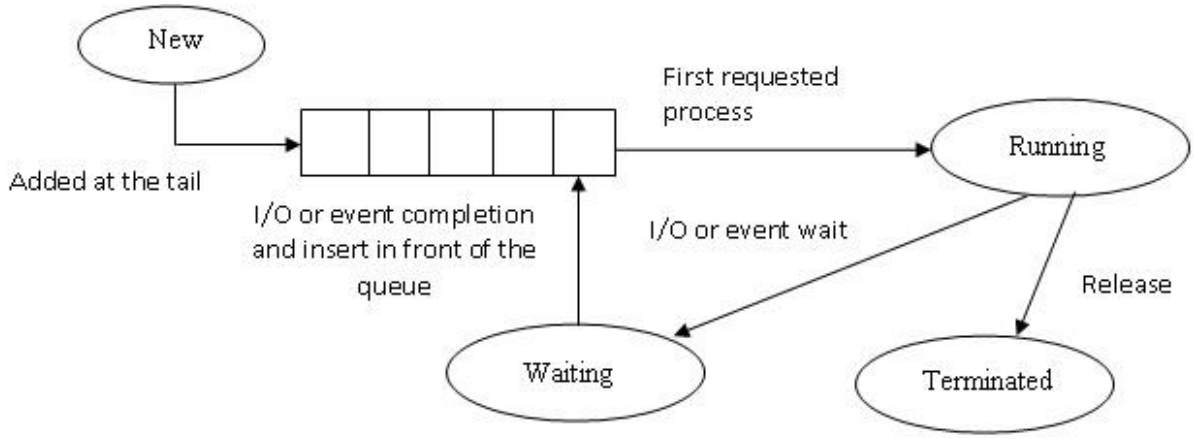
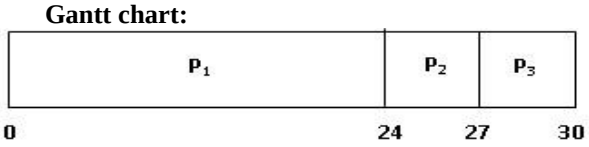


Fig 2: First-Come-First-Serve Scheduling

Example:

Processes	**Burst Time**
P_1	24 ms
P_2	03 ms
P_3	03 ms

Gantt chart:



Thus, average waiting time is 17 ms.

Advantages:

- FCFS scheduling is simple to write and understand.

Disadvantages:

- Average waiting time under FCFS policy is quite long.
- May lead to poor overlap of I/O and CPU since CPU-bound processes will force I/O bound processes to wait for the CPU, leaving the I/O devices idle.

B. Shortest-Job-First(SJF) Scheduling

In this scheduling algorithm, the CPU is allotted to the process which has the smallest next CPU burst. The SJF uses the FCFS to break tie (a situation where two processes have the same length next CPU burst). The SJF algorithm can be pre-emptive or non-pre-emptive. In pre-emptive SJF scheduling, the execution of a process that is currently running is interrupted in order to give the CPU to a newly arrived process with a shorter next CPU burst. On the other hand, the non-pre-emptive SJF will allow the currently running process to finish its CPU burst before a new process is allocated to the CPU.

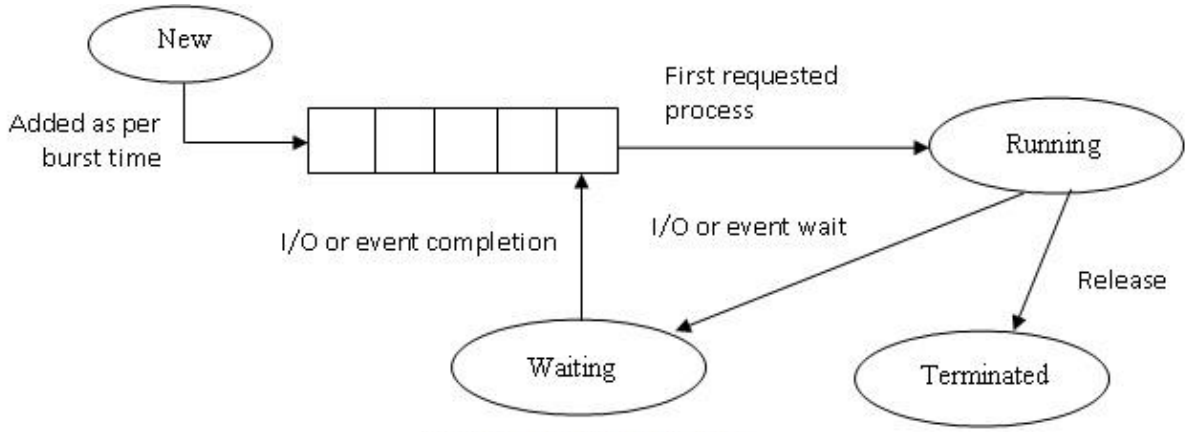
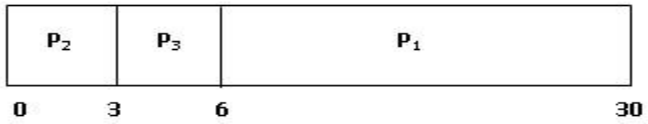


Fig 3: Shortest Job First Scheduling

Example:

Processes	**Burst Time**
P_1	24 ms
P_2	03 ms
P_3	03 ms

Gantt chart:



Thus, average waiting time is 03 ms.

Advantages:

- Provably optimal with respect to minimizing the average waiting time
- I/O bound jobs get priority over CPU bound jobs

Disadvantages:

- If the shortest burst time processes are continuously adding in the ready queue, then long burst time processes will not get the CPU ever. This problem is called as "Starvation".
- Impossible to predict the amount of CPU time a job has left.

C. Priority Scheduling

In priority scheduling algorithm, a priority is associated with each process and the CPU is allocated to the process with the highest priority. The processes which have equal priority; they are scheduled with FCFS policy.

Priority scheduling can be either pre-emptive or non-pre-emptive. When the process arrives at the ready queue, its priority is compared with the priority of the currently running process. If the priority of the newly arrived process is higher, the non-pre-emptive scheduling algorithm will put the newly arrived process at the head of the ready queue. While the pre-emptive scheduling algorithm will interrupt the currently running process and move it to the ready queue and then CPU is allocated to the newly arrived process.

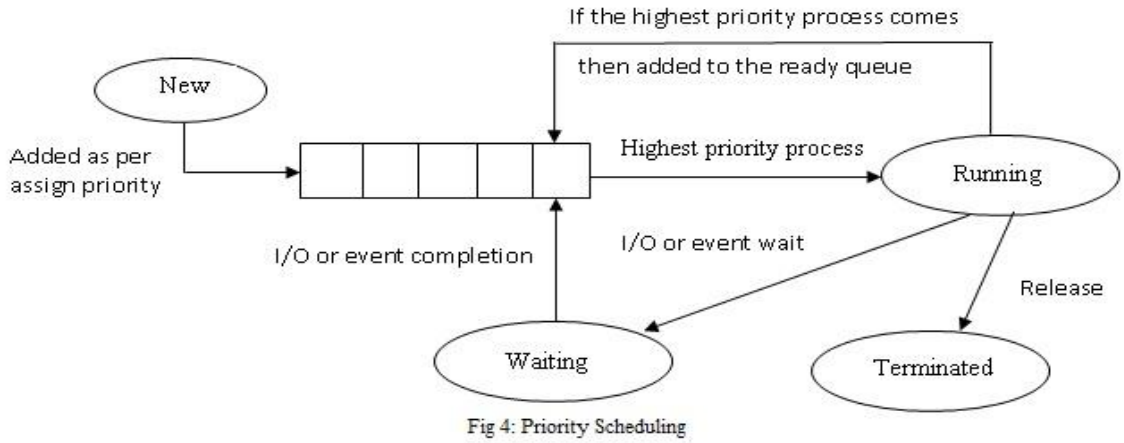
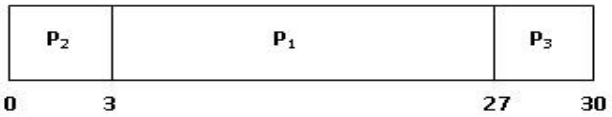


Fig 4: Priority Scheduling

Example:

Processes	**Burst Time**	**Priority**
P_1	24 ms	02
P_2	03 ms	01
P_3	03 ms	03

Gantt chart:



Thus, average waiting time is 10 ms.

Advantages:

- It provides relative importance to each process by providing priorities to them.

Disadvantages:

- If high priority processes are always available in the queue, then low priority processes do not get the CPU ever. This problem is called as "Starvation".

D. Round-Robin(RR) Scheduling

Round robin scheduling is a pre-emptive version of first-come, first-served scheduling. Processes are dispatched in a first-in-first-out sequence but each process is allowed to run for only a limited amount of time. This time interval is known as a **time-slice** or **time quantum**. In this, the ready queue is treated as the circular queue.

One of the two things will happen in Round-Robin. First, the process may have burst-time less than or equal to time quantum. In this case, the process will execute and after completion release the CPU by itself. Second, the process may have burst time greater than time quantum. In this case, the process will execute for 1 time quantum and then it is pre-empted. Then context-switch will be executed and CPU scheduler will select the next process to execute. The pre-empted process will be put at the tail of the ready-queue. This continues till the execution of all the processes is completed.

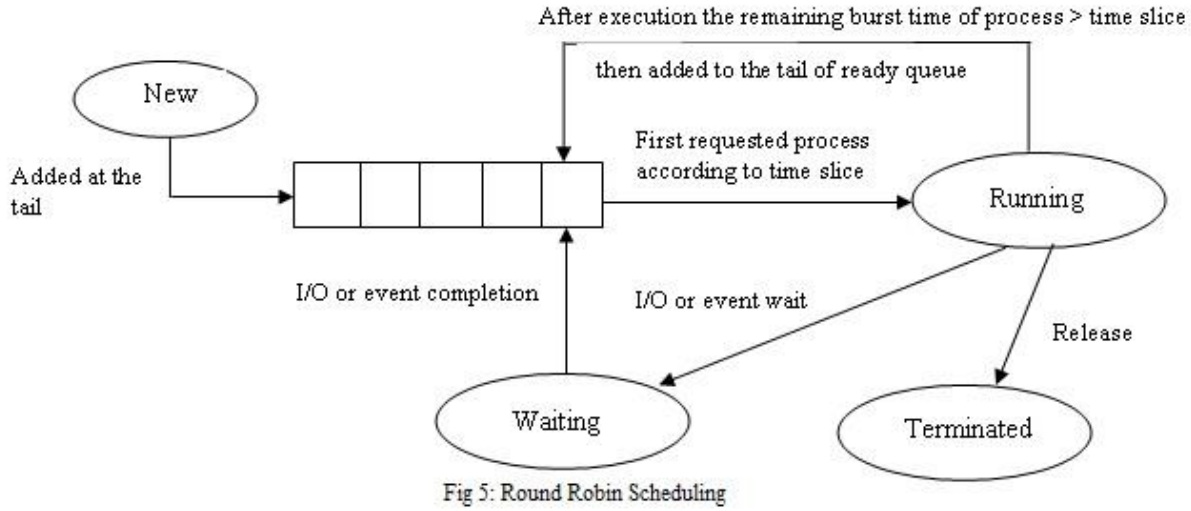


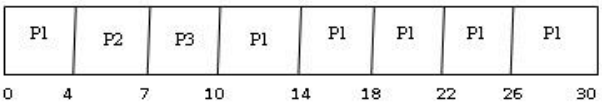
Fig 5: Round Robin Scheduling

Example:

Processes	**Burst Time**
P_1	24 ms
P_2	03 ms
P_3	03 ms

Time quantum=04 ms.

Gantt chart:



Thus, average waiting time is 5.66 ms.

Advantages:

- Round robin scheduling is fair in that every process gets an equal share of the CPU.
- It overcomes the problem of Starvation.

Disadvantages:

- Average waiting time can be bad.
- Giving every process an equal share of the CPU is not always a good idea. For instance, highly interactive processes will get scheduled no more frequently than CPU-bound processes.

E. Multilevel Queue Scheduling

The processes can be classified into different groups depending upon their situation. For example, the common division between processes is foreground processes and background processes. Foreground processes are interactive processes and background processes are batch processes. These types of processes have different response-time requirements and scheduling needs. Also, foreground processes have priority over background processes.

In this, the ready queue is partitioned into several separate queues. And each queue has its own scheduling algorithm. In our example of foreground and background processes, foreground queue uses Round-Robin Scheduling and background queue uses First-Come-First-Serve scheduling.

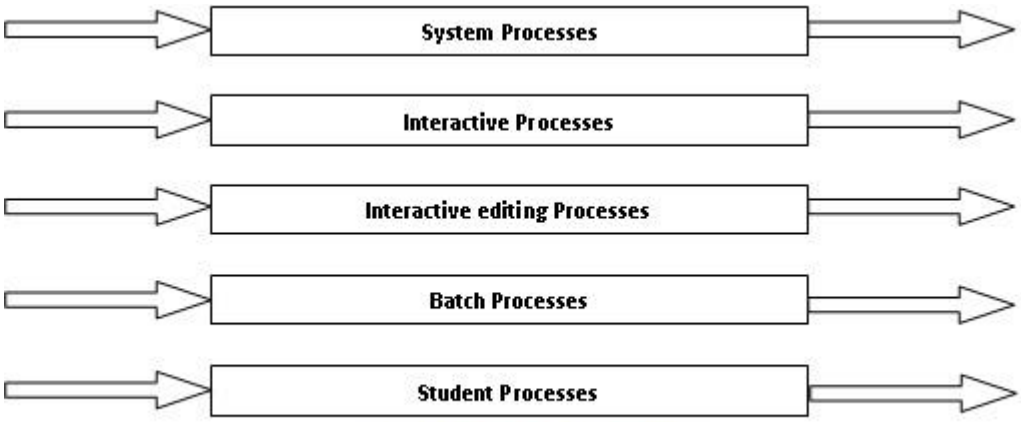


Fig 6: Multilevel queue scheduling

Also, there must be scheduling among the queues which are implemented on the basis of:

1) Fixed-priority pre-emptive scheduling: In this, each queue has absolute priority over lower priority queue.

Let's take an example of five queues listed below in the order of priority as shown in fig 6 above:

- System processes
- Interactive processes
- Interactive editing processes
- Batch processes
- Student Processes

In this, no process in the lower priority queues could run unless all the higher priority queues were empty. So, in our example, if the interactive editing process entered in the ready queue while a batch process was running, then the batch process would be pre-empted. In this, there is a possibility of starvation.

2) Time-slice among the queues: Here, each queue gets a certain portion of CPU time, which it can then schedule among its various processes. By considering our example, the foreground queue can be given 80% of the CPU time for R-R scheduling among its processes, whereas the background queue receives 20% of the CPU time for FCFS scheduling among its processes.

F. Multilevel Feedback Queue Scheduling

In multilevel queue scheduling, the processes are permanently assigned to a queue when they enter in the system. But, multilevel feedback queue allows the processes to move between the queues. In this, the processes are categorized according to their CPU bursts.

- The lower priority queue contains the process that uses too much CPU time.
- The I/O-bound and interactive processes are kept in the higher priority queue.
- If the process waits too long in a lower-priority queue, then it may be moved to a higher-priority queue.

Thus, the multilevel feedback queue scheduling prevents starvation.

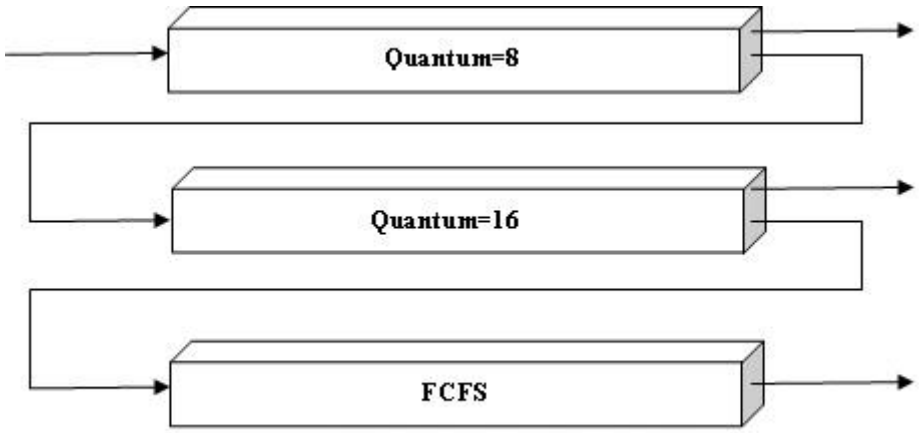


Fig 7: Multilevel feedback queue scheduling

Consider an example of multilevel feedback queue scheduler with three queues, numbered from 0 to 2. The scheduler first executes all the processes in queue 0. When queue 0 is empty, the processes in queue 1 can

execute. Similarly, processes in queue 2 can execute if and only if queue 0 and queue 1 are empty. The running process in queue 2 can be pre-empted if any process arrives in queue 1 or queue 0.

The general working of multilevel feedback queue is as follows:

- A process entering the ready queue is put in queue 0. A process in 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 pre-empted and is put into queue 2.
- Processes in queue 2 are run on an FCFS basis but are run only when queues 0 and 1 are empty.

IV. COMPARISON OF VARIOUS SCHEDULING ALGORITHMS

The following table shows comparison of various scheduling algorithms on different parameters:

TABLE I
COMPARISON OF VARIOUS SCHEDULING ALGORITHMS

Sr. No.	Parameters	FCFS Algorithm	SJF Algorithm	Priority Algorithm	R-R Algorithm	Multilevel Queue Algorithm	Multilevel Feedback Queue Algorithm
1	Preemption	This scheduling algorithm is non preemptive.	This scheduling algorithm is preemptive.	This scheduling algorithm is also preemptive.	This scheduling algorithm is also preemptive.	This scheduling algorithm is also preemptive.	This scheduling algorithm is also preemptive.
2	Complexity	This is simplest scheduling algorithm.	This algorithm is difficult to understand and code.	This algorithm is also difficult to understand.	In this scheduling algorithm, performance heavily depends upon the size of time quantum.	This algorithm is difficult to understand and code.	This algorithm is difficult to understand and code and its performance depends upon the size of time quantum.
3	Allocation	In this, it allocates the CPU in the order in which the process arrives.	In this, the CPU is allocated to the process with least CPU burst time.	It is based on the priority. The higher priority job can run first.	In this, the CPU is allocated in the order in which the process arrives but for fixed time slice.	In this, the CPU is allocated to the process which resides in the higher priority queue.	In this also, the CPU is allocated to the process of higher priority queue.
4	Waiting Time	In this, the average waiting time is large.	In this, the average waiting time is small as compared to FCFS scheduling algorithm.	In this, the average waiting time is small as compared to FCFS scheduling algorithm.	In this, the average waiting time is large as compared to all the three scheduling algorithms.	In this, the average waiting time is small as compared to FCFS scheduling algorithm.	In this, the average waiting time is small as compared to FCFS scheduling algorithm.

V. CONCLUSION

From the above discussion we can say that the first come first serve scheduling algorithm is simple to understand and suitable only for batch system where waiting time is large. The shortest job first scheduling algorithm deals with different approach. In this algorithm, the major benefit is that it gives the minimum average waiting time. The priority scheduling algorithm is based on the priority in which the highest priority job can run first and the lowest priority job need to wait though it will create a problem of starvation. The round robin scheduling algorithm is preemptive which is based on FCFS policy and time quantum. This algorithm is suitable for the time sharing systems. In multilevel queue scheduling, processes are permanently assigned to a queue depending upon its nature and no process in the lower priority queue could run unless the higher priority queues were empty. Also, it is pre-emptive in nature. Multilevel feedback queue scheduling is also pre-emptive in nature and it allows the processes to move between the queues depending upon the given time quantum.

REFERENCES

[1] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne,"Operating system concept", Published in John wiley & sons, Inc. Sixth Edition.
[2] Andrew S. Tanenbaum, Albert S. Woodhull, "Operating Systems Design and Implementation", Second Edition.
[3] Milan Milenkovic, "Operating System Concepts and Design", McGRAW-HILL, Computer Science Series, Second Edition.
[4] Ekta Walia, "Operating system concepts", Published in Khanna Publishing, Second Edition.
[5] Md. Mamunur Rashid and Md. Nasim Adhtar, " A New Multilevel CPU Scheduling Algorithm", Journals of Applied Sciences 6 (9): 2036-2039,2009
[6] D. M. Dhamdhere, "Operating Systems: A Concept Based Approach", Second edition, Tata McGraw-Hill, 2006.
[7] Vikram Singh, Tirlok Gabba, "Comparative Study of Processes Scheduling Algorithms Using Simulator", IJCBR, Volume 4 Issue 2 May 2013
[8] Zafril Rizal M Azmi, Kamalrulnizam Abu Bakar, Abdul Hanan Abdullah, Mohd Shahir Shamsir, Wan Nurulsafawati Wan Manan, "Performance Comparison of Priority Rule Scheduling Algorithms Using Different Inter Arrival Time Jobs in Grid environment", International Journal of Grid and Distributed Computing, Vol. 4, No. 3, September, 2011
[9] Halim Zaim, "Design of a Scheduler Comparison of Different Scheduling Algorithm", Vol. 3, No. 2 May2003.
[10] Neetu Goel, R.B. Garg, A Comparative Study of CPU Scheduling Algorithms, International Journal of Graphics & Image Processing, Vol 2 issue 4, November 2012.

