A Comparative Study of CPU Scheduling Policies in Operating Systems

RAMESH PONNALA*
Asst. Professor,
Dept. of CSE, Sree Chaitanya Institute of Technological Sciences,
Karimnagar, Andhra Pradesh-India 505527
ramesh.ponnala@gmail.com

ABSTRACT

Operating systems are vital system software that, without them, human beings can't manage and use computer system. In essence, an operating system is a collection of software programs whose role is to manage computer resources and provide an interface for client applications to interact with the different computer hardware. Central Processing Unit (CPU) resource management is commonly known as Scheduling. It is the part of the process manager that handles the removal of the running process from the CPU and selection of another process on the basis of a particular strategy. CPU Scheduling is basic functionality of an operating system, since almost all the computer resources are to be scheduled before they use. CPU Scheduling is the basis of multi programmed operating systems. In this paper, I would like to compare and analyze different CPU Scheduling Algorithms for single CPU and show the results in the form of graphs. The objective of the study is to analyze the high efficient CPU scheduler on design of the high quality scheduling algorithms which suits the scheduling goals.

Keywords: Operating System, CPU Scheduling, Processor, Process Scheduling, Scheduler, Scheduling Policies, FCFS, SJF, RR, Priority

I. INTRODUCTION

In a multiprogramming system, multiple processes exist concurrently in main memory. Each process alternates between using a processor and waiting for some event to occur, such as the completion of an I/O operation. The processor or processors are kept busy by executing one process while the others wait. The key to multiprogramming is scheduling.

The aim of processor scheduling is to assign processes to be executed by the processor or processors over time, in way that meets system objectives, such as response time, through put, and processor efficiency. In many systems, this scheduling activity is done by the schedulers. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization. The objective of time sharing is to switch the CPU among the processes so frequently that users can interact with each program while running. To meet these objectives, the process scheduler selects an available process for program execution on the CPU.

Scheduling Queues:

As processes enter the system, they are put into a **job queue**, which consists of all processes in the system. The processes which are resided in the main memory and ready to execute are kept in the queue called as **ready queue.** As there are so many processes in the system, the disk may be busy with the I/O request of some other process. Then the list of waiting processes for I/O device will be kept into **device queue**.

A new process is initially put in the ready queue. It waits there until it is selected for execution, or is dispatched. Once the process is allocated the CPU and is executing one of the several events could occur:

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

In the first two cases, the process eventually switches from the waiting state to the ready state and is then put back in the ready queue. A process continues this cycle until it terminates, at which time it is removed from all queues and it has its PCB and resources de-allocated. The following queuing diagram depicts various queues utilized during process scheduling.

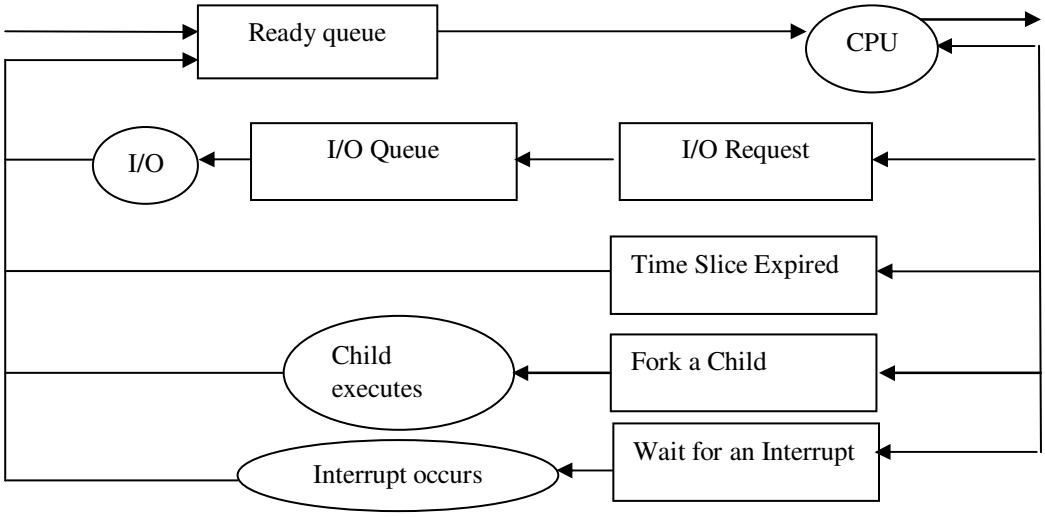

Figure 1: Queuing diagram representation of Process Scheduling

Schedulers:

A process moves among the various scheduling queues throughout its life time. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler by the operating system. It provides the following 3 different schedulers:

- Long-Term Scheduler/ Job Scheduler
- Mid-Term Scheduler/ Medium Term Scheduler
- Short-Tem Scheduler/ CPU Scheduler

a. Long-Term Scheduler/ Job Scheduler

The long-term scheduler determines which programs are admitted to the system for processing. It executes much less frequently. Thus, long-term scheduler controls the degree of multiprogramming. In a batch system, or for the batch portion of general-purpose operating system, newly submitted jobs are routed to disk and held in a batch queue. The long-term scheduler or job scheduler selects processes from this and loads them into main memory for execution. Because of longer interval between executions, the long-term scheduler can afford to take more time to decide which process should be selected for execution.

b. Mid-Term Scheduler/ Medium Term Scheduler

Some operating systems, such as time-sharing systems, may introduce an intermediate level of scheduling, called as mid-term scheduler. The key idea behind a medium term scheduler is that sometimes it can be advantageous to remove processes from memory and thus reduce the degree of multiprogramming. Later, the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called Swapping. The following diagram shows the medium term scheduler

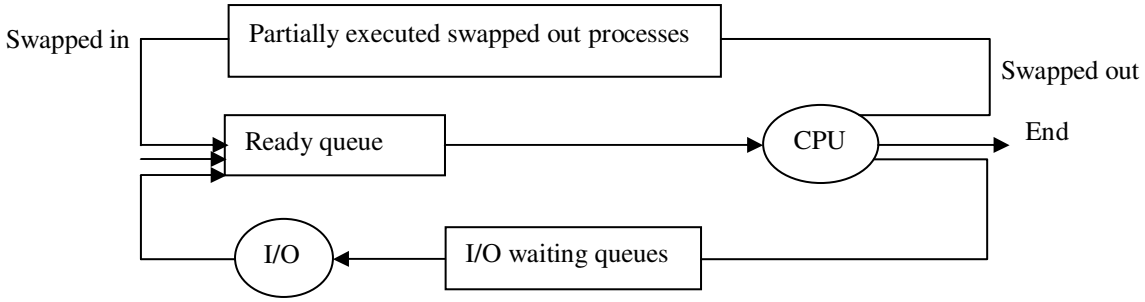

Figure 2: Queuing diagram representation of medium term scheduling

c. Short-Tem Scheduler/ CPU Scheduler

The short-term scheduler or CPU scheduler selects from among the processes that are ready to execute and allocates the processor tone of them. The short-term scheduler must select a new process for the CPU frequently. A process may execute for only a few milliseconds before waiting for an I/O request. Often, the short-term scheduler executes at least once for every 100 milliseconds. Because of the short time between executions, the short-term scheduler must be fast. The main objective of short-term scheduling is to allocate processor time in such a way as to optimize one or more aspects of system behavior. A set of criteria is established against which various scheduling policies may be evaluated.

CPU Schedulers:

When the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler or CPU scheduler. It selects a process from the processes in memory that are ready to execute and allocates the CPU to the process. CPU schedulers are of the of the following categories

- **Non-Preemptive Scheduling**
  In this case, once a process is in the running state, it continues to execute until it terminates or it blocks itself to wait for I/O or to request some operating system service.
- **Preemptive Scheduling:**
  The currently running process may be interrupted and moved to the ready queue by the operating system. 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

CPU scheduling decisions may take place under the following 4 circumstances:

- When a process switches from the running state to the waiting state
- When a process switches from the state to the ready state
- When a process switches from the waiting state to the ready state
- When a process terminates

Scheduling Criteria:

Different CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another. Many criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are used for comparison can make a significant difference in which algorithm is judged to be best. The criteria include the following

d. CPU Utilization: We would like keep CPU as much busy as possible; CPU utilization can range from 0 to 100%. In a real time system, it could range 40% to 90%.

e. Throughput: The scheduling policy should attempt to maximize the number of processes completed per unit of time, also called throughput. It is a measure of how much work is being performed. It clearly depends on the average length of the process but is also influenced by the scheduling algorithm, which may affect utilization.

f. **Turnaround time: T**his is the interval from the time of submission of a process to the time of completion. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.\

g. Waiting time: The CPU scheduling algorithm doesn't affect the amount of time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

h. Response time: In an interactive system turnaround time is may not be the best criterion. For an interactive process, this is the time from the submission of a request until the first response is produced.

It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time.

II. SCHEDULING POLICIES/ALGORITHMS

A process has three basic states namely running, ready, and waiting. A process is said to be running in the running state if it is currently using the CPU. A process is said to be ready in the ready state if it could use the CPU, if it were available. A process is said to be blocked in the waiting state if it is waiting for some event to happen, such as the completion of an I/O operation, before it can proceed. Various events can cause a process to change states. For example, when the currently running process makes an I/O request, it will change from running state to waiting state. When its I/O request completes, an I/O interrupt is generated and then that process will change from waiting state to ready state. For a single CPU system, only one process can run at a time, but several processes may be ready. When more than one process is ready, the operating system must then use a CPU scheduling algorithm to decide which one is to run first and for how long. There are various scheduling algorithms.

a. First-Come, First-Served (FCFS): The simplest CPU scheduling policy is First-Come First-Served scheduling algorithm. The FCFS policy is non-preemptive scheduling. Once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by requesting I/O. Processes are assigned to the CPU in the order they request it. The implementation of FCFS policy is easily managed with a FIFO queue. The average waiting time under the FCFS policy is quite long.

b. Shortest-Job-First (SJF): This is a non preemptive scheduling policy in which the process with the shortest expected processing time is selected next. When the CPU is available, it is allocated to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, the FCFS scheduling policy is used. The most appropriate word for this scheduling is "Shortest-next-CPU-burst algorithm", because scheduling is depends on the length of the next CPU burst of a process, rather than its total length. The SJF policy is optimal, in which it gives the minimum average waiting time for a given set of processes. Starvation is possible, especially in a busy system with many small processes being run.

The difficulty of SJF policy is finding the next shortest CPU burst. The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU bursts. Let t_n be the length of n^{th} CPU burst, and let T_{n+1} be our predicted value for the next CPU burst. Then, for α, $0 \leq \alpha \leq 1$, define

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

c. Shortest-Remaining-Time-First (SRTF): This is a preemptive scheduling policy. When the CPU is available; it is allocated to the process that has the shortest remaining CPU burst. When a process arrives at the ready queue, it may have a shorter remaining CPU burst than the currently running process. Accordingly, the operating system will preempt the currently running process.

d. Priority Scheduling: The SJF Policy is the special case of general priority scheduling policy. A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal priority processes are scheduled in FCFS order. In generally, we treat the priority by some fixed range of numbers i.e., 1-10. Low number indicates highest priority and high number indicates least/low priority. Priority scheduling can be either preemptive or non preemptive. When a process arrives at the ready the ready queue, its priority is compared with the priority of the currently running process. A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the currently running process.\

A major problem with the preemptive priority scheduling policy is indefinite blocking or starvation. A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low priority processes waiting indefinitely. To overcome this problem of starvation, we have a solution called as aging. Aging is a technique of gradually increasing the priority of processes that wait in the system for long time.

e. Round-Robin (RR): The round-robin (RR) scheduling algorithm is designed especially for time-sharing systems. It is similar to FCFS scheduling algorithm, but preemption is added to switch between processes. Thus it is a preemptive scheduler. Each process is given a limited amount of CPU time, called a time slice/ time quantum, to execute. A time quantum can be generally from 10-100 milliseconds. If the required CPU burst of the process is less than or equal to the time slice, it releases the CPU voluntarily. Otherwise, the operating system will preempt the running process after one time slice and put it at the back of the ready queue, then dispatch another process from the ready queue. To implement the RR scheduling, we keep the ready queue as FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU

scheduler picks the first process from the ready queue, sets a timer interrupt after 1 time quantum and dispatches the process.

If there are n processes in the ready queue and time quantum is q, then each process gets 1/n of the CPU time chunks of at most q time units. Each process must wait no longer than (n-1)*q time units until its next time quantum. The performance of the RR scheduling is depends on the size of the time quantum. If the time quantum is too large, the RR policy is same as the FCFS policy. If it is too small, it will cause too many context switches and lower the CPU efficiency.

f. **Multilevel Feedback Queues (MLFQ)**: This scheduling algorithm is a variant version of priority scheduling algorithm designed to prevent high priority processes from running indefinitely. Rather than giving a fixed priority to each process like priority scheduling algorithm, MLFQ varies the priority of a process based on its observed behavior. If, for example, a process repeatedly relinquishes the CPU while waiting for input from the keyboard, MLFQ will keep its priority high, as this is how an interactive process might behave. If, instead, a process uses the CPU intensively for long periods of time, MLFQ will reduce its priority. In this way, MLFQ will try to learn about processes as they run and thus use the history of the process to predict its future behavior.

III. ILLUSTRATION OF SCHEDULING ALGORITHMS WITH EXAMPLE PROBLEM

In this section I would like to present Ghantt charts and calculation of waiting time and turnaround time for the following table named Table 1. And also I'm presenting the results in the form of graph for easy way of comparison of scheduling algorithms.

Process	Burst Time (ms)	Arrival Time	Priority
P1	10	0	3
P2	1	1	1
P3	2	2	3
P4	1	3	4
P5	5	4	2

Table 1: Process with its id, Burst Time, Arrival time, and Priority

a. FCFS

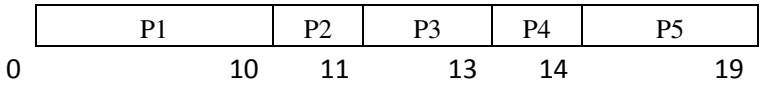

Figure 3: Ghantt chart for FCFS scheduling algorithm

b. SJF

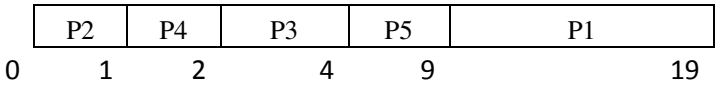

Figure 4: Ghantt chart for SJF scheduling algorithm

c. SRTF

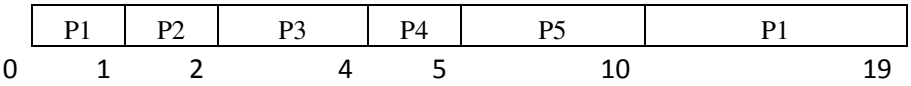

Figure 5: Ghantt chart for SRTF scheduling algorithm

d. Priority

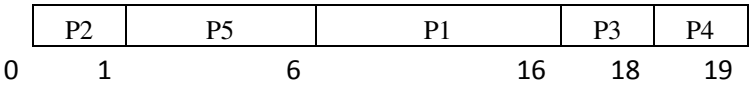

Figure 6: Ghantt chart for priority scheduling algorithm

e. Round-Robin

Time Quantum/Slice=2ms

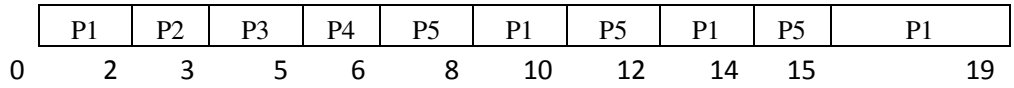

Figure 7: Ghantt chart for round-robin scheduling algorithm

IV. RESULTS

The waiting time for the processes is calculated as time taken by the process to wait in the ready queue is observed from Ghantt chart. For FCFS scheduling, the order of the process is taken as P1,P2,P3,P4, and P5.The Waiting time for processes P1, P2, P3, P4, & P5 is obtained as 0, 10, 11, 13,and 14 respectively and average waiting time is (0+10+11+13+14)/5=9.6ms. For SJF scheduling (without arrival time) the waiting time for processes P1, P2, P3, P4, & P5 is obtained as 9, 0, 2, 1,and 4 respectively and average waiting time is (9+0+2+1+4)/5=3.2ms.

Coming to SRTF (with arrival time) scheduling which is a preemptive scheduling, here based on the arrival time, process burst time, we do get the waiting times for the processes is obtained as **Waiting time=burst time – arrival time.** Thus, the waiting times of the processes is P1=9-0=9, P2=1-1=0, P3=2-2=0, P4=4-3=1, P5=5-4=1 and average waiting time is (9+0+0+1+1)/5=2.2ms. Similarly the waiting time is calculated for all other algorithms and summarized in Table 2 and comparison of the results is done in the form of the bar graph in Figure 7.

Process	WAITING TIME (ms)				
	FCFS	SJF	SRTF	PRIORITY	RR (TQ=2ms)
P1	0	9	9	6	9
P2	10	0	0	0	2
P3	11	2	0	16	3
P4	13	1	1	18	5
P5	14	4	1	1	10
Avg. Waiting Time	9.6	3.2	2.2	8.2	5.8

Table 2: Processes with its waiting time and avg. waiting time in different scheduling policies

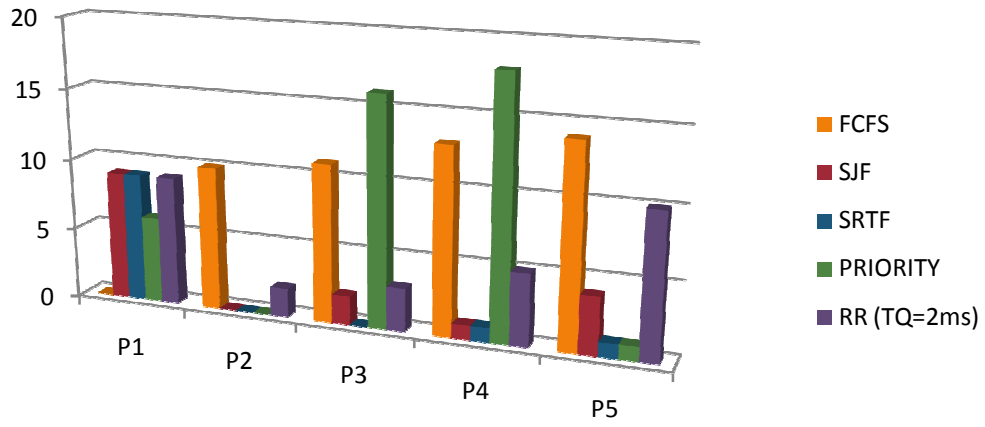

Figure 7: Comparison of CPU scheduling algorithm's waiting time

The turnaround time for the processes is calculated as time taken by the process to wait in the ready queue and its execution time together i.e., **Turnaround time = Waiting time + Burst time;** For FCFS scheduling, the turnaround time for processes P1, P2, P3, P4, & P5 is obtained as 10, 11, 13, 14 and 19 respectively and average waiting time is (10+11+13+14+19)/5=13.4ms. For SJF scheduling (without arrival time) the turnaround time for processes P1, P2, P3, P4, & P5 is obtained as 19, 1, 2, 2,and 6 respectively and average turnaround time is (19+1+2+2+6)/5=7ms. Similarly turnaround time is calculated for all other algorithms and summarized in Table 3 and comparison of the results is done in the form of the bar graph in Figure 8.

Process	TURNAROUND TIME (ms)				
	FCFS	SJF	SRTF	PRIORITY	RR (TQ=2ms)
P1	10	19	19	16	19
P2	11	1	1	1	3
P3	13	4	2	18	5
P4	14	2	2	19	6
P5	19	9	6	6	15
Avg. Turnaround Time	13.4	7	6	12	9.6

Table 3: Processes with its turnaround times and avg. turnaround times in different scheduling policies

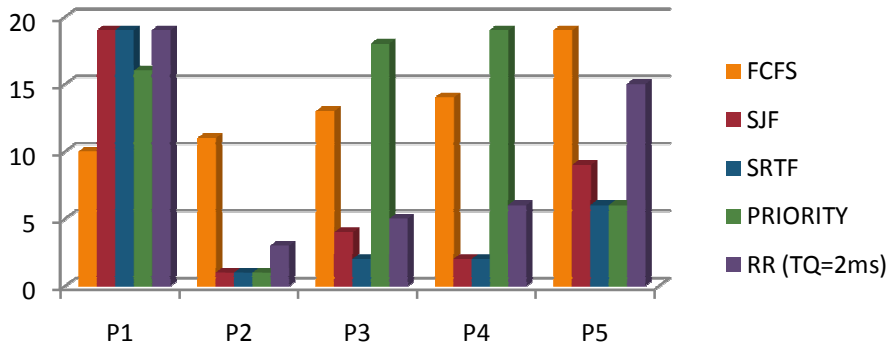

Figure 8: Comparison of CPU scheduling algorithm's waiting time

These results have been justified with the C Programming Language Code for CPU Scheduling algorithms. From the above discussion it is clear that First Come First Serve (FCFS) & Shortest Job First (SJF) is generally suitable for batch operating systems and Round Robin (RR) & Priority Scheduling (PS) is suitable for time sharing systems. No algorithm is optimum for all type of jobs. Hence it is necessary to develop an algorithm with an optimum criteria and suitable for all scenarios.

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 round robin policy one of the scheduling algorithm which follows the interactive system and the round robin scheduling algorithm is deal with the time sharing system.

V. CONCLUSION

The SJF (Preemptive or non preemptive) scheduling algorithm is to serve all types of job with optimum scheduling criteria. The treatment of shortest process in SJF scheduling tends to result in increased waiting time for long processes. And the long process will never get served, though it produces minimum average waiting time and average turnaround time. The shortest job first scheduling algorithm deals with different approach, in this algorithm the major benefit is it gives the minimum average waiting time. It is recommended that any kind of simulation for any CPU scheduling algorithm has limited accuracy. The only way to evaluate a scheduling algorithm to code it and has to put it in the operating system, only then a proper working capability of the algorithm can be measured in real time systems.

It is clearly observed that turnaround time, waiting time and response time of the processes are optimum for SRTF (Preemptive SJF) scheduling algorithm compared to all other scheduling algorithms from Figure 7, Figure 8.

REFERENCES

[1] Silberschatz, A. P.B. Galvin and G. Gagne (2012), Operating System Concepts, 8th Edition, Wiley India

[2] Willium Stallings, Operating Systems Internals and Design Principles, 5th Edition, Pearson Education, 2009

[3] Neetu Goel, Dr. R.B. Garg "A Comparative Study of CPU Scheduling Algorithms", IJGIP, vol.2, Issue 4, pp.245-251, November 2012

[4] Sun Huajin', Gao Deyuan, Zhang Shengbing, Wang Danghui; " Design fast Round Robin Scheduler in FPGA", 0-7803-7547-5/021/$17.00@2002 IEEE

[5] Md. Mamunur Rashid and Md. Nasim Adhtar, " A New Multilevel CPU Scheduling Algorithm", Journals of Applied Sciences 6 (9): 2036-2039,2009

[6] Sukanya Suranauwarat, "A CPU Scheduling Algorithm Simulator", October 10-13, 2007, Milwaukee, WI 37th ASEE/IEEE Frontiers in Education Conference

[7] Sun Huajin', Gao Deyuan, Zhang Shengbing, Wang Danghui;"Design Fast Round Robin Scheduler in FPGA", 0-7803-7547-5/021/$17.00 @ 2002 IEEE

[8] Andrew S. Tanenbaum, Albert S. Woodhull, "Operating Systems Design and Implementation", Second Edition

[9] Milan Milenkovic, "Operating System Concepts and Design", McGRAW-HILL, Computer Science Series, Second Edition

[10] Sukanya Suranauwarat, "A Visual and Interactive Learning Tool for CPU Scheduling Algorithms", International Journal of Computer Science Issues, Vol. 10, Issue 2, No 2, March 2013 ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784