A Comprehensive Review for Central Processing Unit Scheduling Algorithm

Ryan Richard Guadaña[1], Maria Rona Perez[2] and Larry Rutaquio Jr.[3]

**[1] Computer Studies and System Department, University of the East
Caloocan City, 1400, Philippines**

**[2] Computer Studies and System Department, University of the East
Caloocan City, 1400, Philippines**

**[3] Computer Studies and System Department, University of the East
Caloocan City, 1400, Philippines**

Abstract
This paper describe how does CPU facilitates tasks given by a user through a Scheduling Algorithm. CPU carries out each instruction of the program in sequence then performs the basic arithmetical, logical, and input/output operations of the system while a scheduling algorithm is used by the CPU to handle every process. The authors also tackled different scheduling disciplines and examples were provided in each algorithm in order to know which algorithm is appropriate for various CPU goals.
Keywords: *Kernel, Process State, Schedulers, Scheduling Algorithm, Utilization.*

1. Introduction

The central processing unit (CPU) is a component of a computer system that carries out the instructions of a computer program, and is the primary element carrying out the computer's functions. The central processing unit carries series of program instructions, executes both logical and arithmetical functions, and handles input/output operations of the system. The demand of activities to be performed by the CPU piqued the authors' interest on how CPU handles different tasks given by the user?

The question on how does a CPU handles different tasks given by the user is answered through scheduling. Scheduling is a key concept in computer multitasking, multiprocessing operating system and real-time operating system designs. Scheduling refers to the way processes are assigned to run on the available CPUs, since there are typically many more processes running than there are available CPUs, like shoppers sharing the checkout operators on their way out of the store. There are different types of Operating System schedulers that the authors focused on. First is the Long Term Scheduler also known as the admission scheduler that decides which jobs or processes are to be admitted to the ready queue; that is, when an attempt is made to execute a program, its admission to the set of currently executing processes is either authorized or delayed by the long-term scheduler. Second is the Mid-term Scheduler that temporarily removes processes from main memory and places them on secondary memory (such as a disk drive) or vice versa. Last is the Short Term Scheduler that decides which of the ready, in-memory processes are to be executed.

2. CPU Utilization

In order for a computer to be able to handle multiple applications simultaneously there must be an effective way of using the CPU. Several processes may be running at the same time, so there has to be some kind of order to allow each process to get its share of CPU time. One of the most important components of the operating system is the kernel, which controls low-level processes which is typically unknown to the average user. It controls how memory is read and written, the order in which processes are executed, how information is received and sent by devices like the monitor, keyboard and mouse, and decides how to interpret information received from networks. Kernel is also the central component of most computer operating systems that bridges applications and computer peripherals.

2.1 The CPU Process States

When a process is created, its state is set to new. Once the process is ready to use the CPU its state is set to ready. It is inserted into the ready queue waiting its turn to be assigned CPU time so that its instructions can be executed. Once the CPU is available the process next in line in the ready queue is set to running. This means that the process' instructions are being executed.

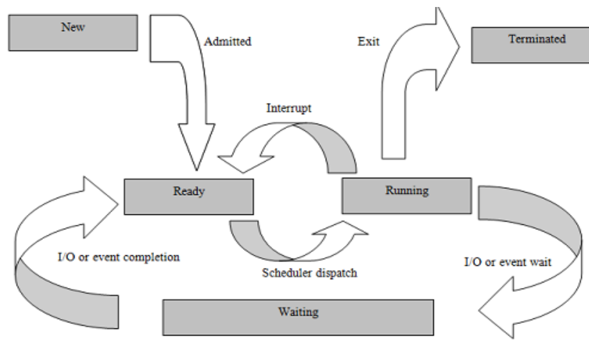

Fig. 1 CPU Process States.

Once the process is being executed two things can happen:

1) The process' instructions are all executed in which case its state will be set to terminate.

2) While the process is running an I/O interrupt or event wait is executed which stops the running program.

In the event the first case takes place, the program finishes executing and then terminates. This means all the instructions of the process have been executed and it has no more need for the CPU. However, this can also happen if there is some kind of error in the program that requires the process to be terminated prematurely. In the second case the procedures taken are much more complex. For example, let us say that there is a process that is currently occupying the CPU. As the instructions of this process are being executed the program needs to get input from the user at the keyboard. This causes the process to stop executing. In this situation the process will enter the waiting state. This means that the process will lose control of the CPU and be inserted into the waiting queue. Once the input is received from the user at the keyboard the process must go back to the ready state. The process cannot take hold of the processor; it must wait in the ready queue until it is assigned the CPU.

Once the process is assigned the CPU again, it will continue executing its instructions. Once again two things may happen. If there is need for more I/O then the process will once again enter into the waiting state. If not, then the process will complete and will become terminated once the final instructions are executed. As stated earlier a process may enter several states in its lifetime. However, where is this information stored? It is stored in the process control block (PCB). The process control block is a representative of each process. It contains information about the process, which it is associated with. The information it contains is the process state, program counter, CPU registers, CPU-scheduling information, memory management information, accounting information, and I/O status information.

CPU scheduling information is information that includes process priority, pointers to scheduling queues, and any other scheduling parameters. This is the basis of multi-programmed operating systems because the CPU is able to switch from process to process while the operating system is able to make the running programs seem as if they are being executed simultaneously. Whenever the CPU has to wait for I/O operations to occur, there are CPU cycles that are being wasted. The idea behind CPU scheduling is to be able to switch from process to process when the CPU becomes idle. In this way, while a process is waiting for an I/O request to complete, the CPU does not have to sit idle. It can begin executing other processes that are in the waiting state.

There are two scheduling schemes that are available. There is the non-preemptive scheduling scheme and there is the preemptive scheduling scheme. Different CPU scheduling algorithms have different properties and may have one class of processes over another. Many criteria have been suggested for comparing CPU scheduling algorithms. The characteristics used for comparison can make a substantial difference in the determination of the best algorithm. The criteria should include the following:

- CPU Utilization: This measures how busy the CPU is. CPU utilization may range from 0 to 100 percent. In a real system, it should range from 40% (for a lightly loaded system) to 90% (for heavily loaded system).
- Throughput: This is a measure of work (number of processes completed per time unit). For long processes, this rate may be one process per hour; for short processes, throughput might be 10 processes per second.
- Turnaround Time (TT): This measures how long it takes to execute a process. Turnaround time is the interval from the time of submission to the time of completion. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing in the CPU, and doing I/O.
- Waiting Time (WT): CPU scheduling algorithm does not affect the amount of time during which process executes or does I/O; it affects only the amount of time a process spends waiting in the ready queue. Waiting time is the total amount of time a process spends waiting in the ready queue.
- Response Time: The time from submission of a request until the system makes the first response. It is the amount of time takes to start responding but not the time that it takes to output that response. The turnaround time is generally limited by the speed of the output device.

3. Scheduling Algorithm

3.1 Non-preemptive scheduling algorithm

Non-preemptive or also known as the cooperative scheduling is the first scheme where once a process has control of the CPU no other processes can preemptively take the CPU away. The process retains the CPU until either it terminates or enters the waiting state. There are two algorithms that can be used for non-preemptive scheduling. There are different algorithms under non-preemptive scheduling scheme and these are the following:

3.1.1 First-Come, First-Served (FCFS) scheduling algorithm

In this scheduling algorithm the first process to request the CPU is the one that is allocated the CPU first. The First-Come, First-Served algorithm is very simple to implement. It can be managed using a First-In, First-Out (FIFO) queue. When the CPU is free, it is allocated to the first process waiting in the FIFO queue. Once that process is finished, the CPU goes back to the queue and selects the first job in the queue. An analogy for this is students waiting in line to pay for their lunch. When one student is ready to pay for their meal, they must go to the back of the line and wait for their turn. This is the idea behind the First-Come, First-Served algorithm.

Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Table1. Given example of processes for FCFS

Process	Burst Time
P1	24
P2	3
P3	3

If the process arrives in the order P1, P2, P3, and are served in FCFS order, the gets the result shown in the following Gantt chart:

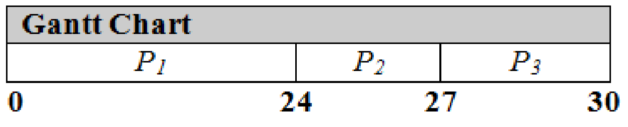

Fig. 2 Gantt Chart illustration of FCFS.

Therefore, the waiting time for each process is:

WT for P1 = 0 – 0 = 0
WT for P2 = 24 – 0 = 24
WT for P3 = 27 – 0 = 27

Average WT = (0 + 24 + 27) / 3
= 17 ms

The turnaround time for each process would be:

TT for P1 = 24 – 0 = 24
TT for P2 = 27 – 0 = 27
TT for P3 = 30 – 0 = 30

Average TT = (24 + 27 + 30) / 3
= 27 ms

3.1.2 Shortest Job First (SJF) scheduling algorithm

In this scheduling scheme the process with the shortest next CPU burst will get the CPU first. The movement of all the short jobs ahead of the longer jobs will decrease the average waiting time. If two processes have the same length of CPU burst, FCFS scheduling is used to break the tie by considering which job arrived first.

Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Table2. Given example of processes for SJF

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

Using SJF, the system would schedule these processes according to the following Gantt chart:

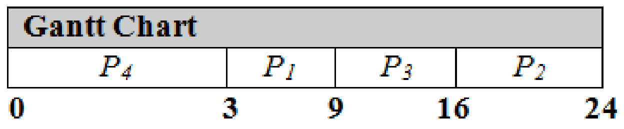

Fig. 3 Gantt Chart illustration of SJF.

Therefore, the waiting time for each process is:

WT for P1 = 3 – 0 = 3
WT for P2 = 16 – 0 = 16
WT for P3 = 9 – 0 = 9
WT for P4 = 0 – 0 = 0

Average WT = (3 + 16 + 9 + 0) / 4
= 7 ms

The turnaround time for each process would be:

TT for P1 = 9 – 0 = 9
TT for P2 = 24 – 0 = 24
TT for P3 = 16 – 0 = 16
TT for P4 = 0 – 0 = 0

Average TT = (9 + 24 + 16 + 0) / 4
= 12.25 ms

3.1.3 Priority (Prio) scheduling algorithm

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. An SJF is simply a priority algorithm where the priority (p) is the inverse of the next CPU burst (ז). The larger the CPU burst, the lower the priority, and vice versa.

Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Table3. Given example of processes for Prio

Process	Priority	Burst Time
P1	3	10
P2	1	1
P3	4	2
P4	5	1
P5	2	5

Using priority algorithm, the schedule will follow the Gantt chart below:

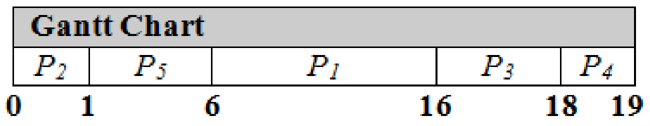

Fig. 4 Gantt Chart illustration of Prio.

Therefore, the waiting time for each process is:

WT for P1 = 6 – 0 = 6
WT for P2 = 0 – 0 = 0
WT for P3 = 16 – 0 = 16
WT for P4 = 18 – 0 = 18
WT for P5 = 1 – 0 = 1

Average WT = (6 + 0 + 16 + 18 + 1)/5
= 8.2 ms

The turnaround time for each process would be:

TT for P1 = 16 – 0 = 16
TT for P2 = 1 – 0 = 1
TT for P3 = 18 – 0 = 18
TT for P4 = 19 – 0 = 19
TT for P5 = 6 – 0 = 6

Average TT = (16 + 1 + 18 + 19 + 6)/5
= 12.25 ms

3.2 Preemptive scheduling algorithm

Preemptive scheduling is the second scheduling scheme. In preemptive scheduling there is no guarantee that the process using the CPU will continually run until it is finished. This is because the running task may be interrupted and rescheduled by the arrival of a higher priority process.

3.2.1 Shortest Remaining Time First (SRTF) scheduling algorithm

The SJF has a preemptive adaptation commonly referred to as shortest remaining time first; the process that is running is compared to the processes in the ready queue. If a process in the ready queue is shorter than the process running, then the running task is preempted and the CPU is given to the shorter process until it is finished.

Consider the following set of processes with the length of the CPU burst given in milliseconds:

Table4. Given example of processes for SRTF

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	1
P4	3	5

If the processes arrive at the ready queue at the times shown and need the indicated burst times, then the resulting preemptive SJF schedule is as depicted in the following Gantt chart:

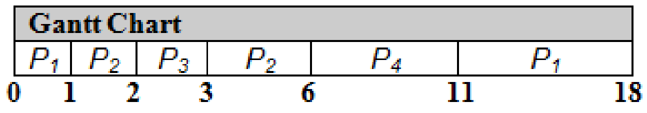

Fig. 5 Gantt Chart illustration of SRTF.

Therefore, the waiting time for each process is:

WT for P1 = 11 – 0– (1) = 10
WT for P2 = 3 – 1– (1) = 1
WT for P3 = 2 – 2 = 0
WT for P4 = 6 – 3 = 3

Average WT = (10 + 1 + 0 + 3) / 4
= 3.5 ms

The turnaround time for each process would be:

TT for P1 = 18 – 0 = 18
TT for P2 = 6 – 1 = 5
TT for P3 = 8 – 2 = 6
TT for P4 = 13 – 3 = 10

Average TT = (18 + 5 + 6 + 10) / 4
= 9.75 ms

3.2.2 Preemptive Priority (P-Prio) scheduling algorithm

Priority scheduling can either be preemptive or non-preemptive. When a process arrives at the ready queue, its priority is compared with the priority of the process, which is currently executing at the CPU. A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrive process is higher than the currently running process. A major problem with the priority scheduling algorithms, whether preemptive or non-preemptive is indefinite blocking or starvation. In a heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU. A solution to the problem of indefinite blocking is aging. Aging is the technique of gradually increasing the priority of process that wait in the system for a long time.

Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Table5. Given example of processes for P-Prio

Process	Arrival Time	Burst Time	Priority
P1	1	5	5
P2	2	10	4
P3	3	18	3
P4	4	7	2
P5	5	3	1

Using preemptive priority algorithm, the schedule will result to the Gantt chart as follows:

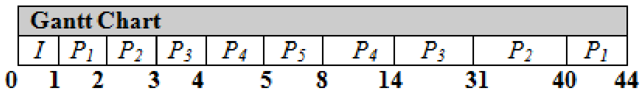

Fig. 6 Gantt Chart illustration of P-Prio

Therefore, the waiting time for each process is:

WT for P1 = 40 – 0– (1) = 38
WT for P2 = 31 –2– (1) = 28
WT for P3 = 14 – 3– (1) = 10
WT for P4 = 8 – 4– (1) = 3
WT for P5 = 5 – 5 = 0

Average WT = (38 + 28 + 10 + 3 + 0)/5
= 15.8 ms

The turnaround time for each process would be:

TT for P1 = 44 – 1 = 43
TT for P2 = 40 – 2 = 38
TT for P3 = 31 – 3 = 28
TT for P4 = 14 – 4 = 10
TT for P5 = 8 – 5 = 3

Average TT = (43 + 38 + 28 + 10 + 3)/5
= 24.4 ms

3.2.2 Round – Robin (RR) scheduling algorithm

This algorithm is specifically for time – sharing systems. A small unit of time, called a time quantum or time slice, is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum. The RR algorithm is therefore preemptive.

Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Table6. Given example of processes for RR

Process	Burst Time
P1	24
P2	3
P3	3

If the system uses a time quantum of 4ms, then the resulting RR Gantt chart is:

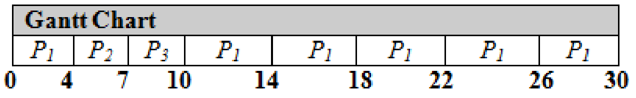

Fig. 7 Gantt Chart illustration of RR

Therefore, the waiting time for each process is:

WT for P1 = 26 – 0– (20) = 6
WT for P2 = 4 – 0 = 4
WT for P3 = 7 – 0 = 7

Average WT = (6 + 4 + 7) / 3
= 5.67 ms

4. Analysis

The authors looked into a number of different scheduling algorithms and the two different scheduling schemes that was discussed in this paper, the preemptive and non-preemptive scheduling scheme. In order to know which algorithm to use for which CPU scheduling goal, different examples were given in each algorithm. Therefore, based on performance, the shortest job first (SJF) algorithm is recommended for the CPU scheduling problems of minimizing either the average waiting time or average turnaround time but the addition of preemption to the SJF algorithm gives supplementary increase in waiting and turnaround time, without affecting the response time. Long jobs have an even higher tendency to cause delay at the back of the queue since they can be interrupted by short jobs so even when long jobs get a chance to execute, they can be interrupted.

Also, the first come first serve (FCFS) algorithm is recommended for the CPU scheduling problems of minimizing either the average CPU utilization or average throughput but the discrepancy about FCFS is it promotes starvation[1].

The performance of the RR algorithm depends heavily on the size of the time quantum. It is concluded that if the quantum is too large, the RR policy degenerates into the FCFS policy. If the time quantum is too small, on the other hand, then the effect of the context – switch time becomes a significant overhead. As general rule, 80 percent of the CPU burst should be shorter than the time quantum.

In general, task given by the user to OS will use Priority based, Round Robin and preemptive while Real Time OS will use Priority and non preemption scheme.

Acknowledgments

The authors wish to thank their chairman, Prof. Amelia Damian, for sharing her knowledge on how to write a thesis onto journal format and her encouragement to finish this article. A special gratitude is also extended to their fellow faculty and colleagues.

[1] Starvation means that a job with low priority would never get a chance to enter the processor if there is steady stream of jobs or processes.

References

[1] Gisela May A. Albano, and Angelito G. Pastrana, Fundamentals of Operating System, A & C Printers, 2009.
[2] Silberschatz ,Galvin and Gagne, Operating systems concepts, 8th edition, Wiley, 2009.
[3] Silberschatz, Abraham, Peter B. Galvin, and Greg Gagne, Operating System Principles, 6th Edition, John Wiley and Sons Inc., 2006.
[4] Flynn, Ida M., and Ann Mclver McHoes, Understanding Operating Systems, 4th Edition, Thomson Learning Inc., 2007.
[5] Mehdi Neshat, Mehdi Sargolzaei, and Adel Najaran, "The New Method of Adaptive CPU Scheduling Using Fonseca and Fleming's Genetic Algorithm", Journal of Theoretical and Applied Information Technology, Vol. 37, No. 1, 2012, pp. 1-16.
[6] Huda Salih, and Yousra Fadil, "CPU Scheduling Simulation", Diyala Journal of Engineering, Vol. 02, No. 7, 2009, pp.39-52
[7] Abbas Noon, Ali Kalakech, and Seifedine Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average", IJCSI International Journal of Computer Science Issues, Vol. 8, No. 1, 2011, pp. 224-229.

Ryan Richard H. Guadaña is a University of Caloocan City BS Computer Science graduate (2004) presently writing his thesis in Polytechnic University of the Philippines for his Masters of Science in Information Technology degree. He is at this time (2012) a full time faculty of University of the East Philippines. He has gained experience in Informatics International College as an OIC-Academic Head from 2008-2010 and as OIC-Office of Students Affairs from 2007-2009. Currently, he is interested and engaged in researches about computer aided training especially for Non-IT training such as disaster response. Now, he is a member of PSITE Philippines and ProICT Philippines.

Maria Rona Perez is a full time faculty of the Department of Computer Studies and Systems, College of Engineering, University of the East, Philippines while pursuing her Thesis Writing towards M.S. in Information Technology from Polytechnic University of the Philippines. She obtained B.S Information Technology "Cum Laude" from AMA Computer College - Manila Campus. Her research interest includes social studies, security and ethical issues and information system. She is a member of PSITE Philippines.

Larry T. Rutaquio Jr. is currently finishing his Graduate Study at Polytechnic University of the Philippines with the degree of Master of Science in Information Technology with specialization in Management Information System. Presently, he is working as a full-time faculty at the University of the East – Caloocan campus. He is also a member of Philippine Society of Information Technology Educators (PSITE) and ProICT Philippines. His area of interest are cloud computing and network security.