INTERNATIONAL JOURNAL OF ADVANCED STUDIES IN COMPUTER SCIENCE AND ENGINEERING
IJASCSE VOLUME 5, ISSUE 5, 2016

05/31/2016

Evaluating the various CPU scheduling algorithms on the basis of Simulation made in C++

Vaibhav Kant Singh

Department of Computer Science and Engineering
Institute of Technology, Guru Ghasidas Vishwavidyalaya,
Central University, Koni, Bilaspur, Chhattisgarh, India

Abstract **– One goal of Operating System is to use the Computer Hardware/Resource in an efficient manner. Since CPU is a resource it should be utilized efficiently. To achieve high degree of CPU utilization Scheduling algorithms are implemented by operating system. In this paper we will see comparison between the various CPU scheduling algorithms on the basis of the simulation made in C++. The performance was tested on same workload. The paper also gives a brief introduction of the basic evaluation techniques used for the CPU scheduling algorithms. Simulation is made of First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling and Round Robin (RR) Scheduling in C++. Also for assessing the performance of the algorithms they are also implemented in Code block. Beside the above algorithms working of Improved Round Robin Scheduling, Highest Response Ratio Next (HRRN), Virtual Round Robin Scheduling, Multilevel Queue Scheduling, Multi-level Feedback/ Adaptive Queue Scheduling, Fair-share Scheduling and Lottery Scheduling are discussed.**

Index Terms – CPU scheduling, First Come First Serve(FCFS), Shortest Job First(SJF), Priority, Round Robin(RR),

I. INTRODUCTION

A Computer is an Electronic device that accepts data processes it and gives the output in a desired manner. Since computer is an electronic device it understands only logic of 0's and 1's. To achieve this concept of Software and Program came. A program is a set of instruction in a specific sequence to perform a specific objective. Whereas Software is basically set of instruction group into programs that make the Computer to function in the desired manner. Software can be classified into Pre-Written Software and Custom made Software. Pre-Written Software could be further classified into Application and System. Operating System for which is going to use CPU-Scheduling is a type of System Software. System Software is software that is meant to operate the hardware of a computer and to provide and maintain a platform for application running software. Operating System is thus system software that acts as an intermediary between a user of computer and the computer hardware. The major purpose of OS is to provide an environment in which a user can execute programs. Some OS system structures include Monolithic, Layered, Client-Server, Virtual Machine, and Exo-Kernel respectively. The basic components of OS are Process Management, Main Memory Management, File Management, I/O System Management, Secondary Storage Management, Networking, Protection and Command Interpreter System. The services provided by OS are I/O operations, Program Execution, File-System Manipulation, Communications, Error Detection, Resource Allocation, Accounting and Protection. The basic functions of OS are Memory Management, Processor Management, Device Management and File Management. Operating System could have various classifications like Real-time(Hard and Soft), Multiprogramming, Time sharing, Multiprocessor, Batch etc. Examples of OS are Microsoft Windows family of OS (Windows 7, Windows 8, Windows XP etc.), UNIX family of OS which is having two classifications namely UNIX or UNIX system-like(LINUX Ubuntu, Fedora, Redhat etc.) etc.

In system of the yester years the system was dedicated to process one program at one time. But now scenario has changed, we are entertaining multiple processes at the same time. This require construct to implement, here came the concept of process. Process is a program in execution. It is an instance of a program. Process is the unit of work done in modern time sharing operating system. Process is represented in the OS by a Process Control Block (PCB) which contains several information which are dynamic, and because of which process is called an active entity. Process is basically the element which is going to be considered by the CPU for execution. A point to be noted is that until and unless the process component which is supposed to be executed is in memory it can't be executed. The List of processes that are ready for execution, are kept in a list called ready queue. During the process lifetime it goes through various states New (When a process is newly admitted), Ready (When the process attains the resources it desires for its execution and is placed in the ready queue), Waiting (When a process is waiting for an

event completion or waiting for an execution from the I/O device), Running (When a process is having processor allocation and is running) and Terminated (When process completes its execution). As already discussed that one goal of OS is to have efficient resource utilization, since the speed of CPU is in nanoseconds the time could be used productively by enabling the concept of Multiprogramming.

In this paper total XI sections are there excluding references. In section I a brief introduction to the Operating System is given. In section II the problem statement is given, the problem for which the research is made is given. In section III Literature survey of the various methodologies in the Operating System domain for the purpose are discussed. In IV the basic CPU scheduling concepts are discussed. In section V the scheduling criteria used for making assessment of the CPU scheduling algorithm is discussed. In section VI the basic evaluation methods utilized for assessing the performance of CPU scheduling algorithm is given. In section VII the basic working of the algorithms by means of deterministic modeling is explained(Gantt chart used). In section VIII snapshots of the algorithms implemented in Turbo C++ is given. In section IX Experiment carried out on Code block and Turbo C++ which gave results on various scenarios is discussed by means of Tables. In section X the results generated by means of the Experiment done in expressed by means of Graph. In section XI the conclusion of the research work is discussed.

II. Problem Statement

The objective of Multiprogramming is to keep the CPU busy at all time. In a uniprocessor system at one time only one process is suppose to execute. The remaining processes are supposed to wait. As the process is able to have the resources required for its execution, and is having processor allocation it will start executing. During execution it may acknowledge various events one of which could be occurrence of a request for an I/O device. When this type of request from the process arrives in simple Computer System the system will sit idle for the time I/O operation is accomplished by the Process. Now, as already discussed the objective of OS is to keep efficient utilization of the resource which is hampered in the above case. This gave rise to the notion of CPU scheduling which enables selection of a process from the ready queue to make the efficient utilization of CPU. CPU scheduling is done by a construct called CPU scheduler or Short-term scheduler. Short-term name is given on the basis of frequency of calling time.

III. Literature Survey

In [1] the dilemma is addressed where the Implementability of threads to achieve parallelism is not satisfactorily achieved. In [2] notion of concurrent programming is presented some approaches regarding the usage of important language notation for representation of concurrency is addressed. In [3] the authors had made a study on the problem of efficiently scheduling fully strict multithreaded computations on parallel computers. In [4] Coordinated thread scheduling is discussed as a critical factor in achieving good performance for tightly coupled parallel jobs on Workstation clusters. In [5] Surplus Fair scheduling (SFS), a proportionate share, CPU scheduler designed for symmetric processor is proposed. In [6] BVT Borrowed Virtual Time Scheduling is proposed, showing that it provides low-latency for real-time and interactive applications yet weighted sharing of the CPU across applications according to system policy. In [7] performance of Multiprogramming systems is made. In [8] the SDC time sharing system is revisited. In [9] a fair share scheduler is proposed. In [10] solution of a problem in concurrent programming control is discussed.

IV. CPU Scheduling Concepts

During the process life time it toggles between two states either it is engaged in doing some operation using CPU or engaged in doing some I/O operation. Cycle of CPU-execution is called CPU burst and Cycle of I/O execution is called I/O burst. During execution the first and last cycle is CPU-burst. There are four conditions which may be acknowledged during the execution of a process:-

1) Process switches from running state to waiting state.
2) Process switches from waiting state to ready state.
3) Process switches from running state to ready state.
4) Process terminates

In 1 and 4 definitely new selection will be made. But in 2 and 3 forceful deallocation of CPU make take place. If scheduling it takes place only under 1 and 4 it is called non preemptive otherwise it is preemptive. Another important point that must be noted is that CPU-scheduler merely makes selection of the process for allocation of CPU. Physical allocation is made by dispatcher. Dispatching requires three events i.e. Context switching, Switching to user mode and restarting process from an appropriate location.

V. CPU Scheduling Criteria

CPU scheduling criteria can be classified into two categories namely user based and system based. A user based includes Turnaround time, Waiting time, Response time, Predictability and Deadlines. System based include Throughput, CPU utilization, Fairness and Balance.

A. *User Based*

1) Turnaround time: The time elapsed between the time of submission to its completion.

2) Waiting time: The time the process spends waiting in the ready queue.

3) Response time: The time from process submission to the first response.

4) Predictability: This is based on the assessment made by the human mind for the amount of time a process may take for its execution.

5)Deadlines: In real-time systems the process are supposed to be completed in rigid time constraint. Thus deadlines achievability could be criteria for making a comparison between algorithms that imposes deadlines in their operations.

B. System Based

1) Throughput: The number of processes completed in unit time, where unit time may be sec, minutes, hours etc.

2) CPU Utilization: It is the percentage of time for which the CPU was engaged in doing work.

3) Fairness: Every process must be entertained is the policy of this criterion.

4) Balance: A good balance of I/O bound and CPU bound processes while execution.

VI. Evaluation Methods for CPU Scheduling

There are four basic techniques which are used to make an assessment of the CPU Scheduling algorithm. The Techniques are Deterministic Modeling, Queuing Models technique, Simulation and Implementation. The first technique is the one where using mathematical formulation for a given snapshot the different scheduling algorithms are assessed. In the first technique Gantt chart is first prepared and on the basis of the chart formed the calculations are made for the criteria defined above. In the second technique Computer is viewed as a network of servers providing various resources to the users. Since CPU is a resource and has to provide services with the help of queues. Little's formula is announced for calculating the Waiting time for a given scheduling algorithm. In the third technique we will make simulation of the given CPU scheduling algorithm in the language we desire and then carry out experiments on the simulated software for different workload. This is the most cost effective approach utilized for making evaluation of performance of the CPU scheduling algorithm. The fourth technique is the most costly and the most accurate one. In this approach we implement the algorithm and then deploy it into the operating system as a CPU scheduling algorithm and then test the performance made by the system. The above criteria's are tested to obtain the comparison between the algorithms. Since, construction of CPU scheduling algorithm for deployment require consideration of additional constraints which deal with the practical aspect of execution thus more effort and cost is required for its construction.

VII. Working of Some Existing CPU-Scheduling Algorithms

In FCFS the process which arrives first get the CPU allocation first. It is a non-preemptive scheduling algorithm. SJF is algorithm that considers the process for scheduling which is having the least value for next CPU-burst. It is both preemptive and non-preemptive. Priority Scheduling considers the process which is having the highest priority as the process for CPU allocation. It is both preemptive and non-preemptive. Round Robin is used to give fast response to the user. It does that by executing each process for a time slice. It is preemptive scheduling algorithm. In Improved Round Robin Scheduling the policy that is generally adopted is to schedule a process if its CPU consumption ratio is greater than 0.60, else schedule a process whose CPU consumption ratio is minimum. In Highest Response Ratio Next Scheduling that process will be scheduled which has the highest response ratio. In Virtual Round Robin Scheduling along with ready queues auxillary queues are utilized to serve the I/O bound processes in better fashion. In Multilevel queue scheduling the ready queue is partitioned into various groups of queues and the processes on the basis of the properties that they exhibit are destined to the respective queues. The Individual queues are scheduled using the basic approaches namely FCFS, SJF, Priority, RR. Between the queues the scheduling policy is Preemptive Priority. In Multilevel queues there may be problem of starvation. In Multilevel feedback queue the processes are allowed to move between the queues. This removes the starvation problem acknowledged in the previous case to a wide extent. In Fair-share scheduling algorithms the needs of a user or group of users is considered and processor time is distributed not among the individual processors, but among the users or group of users as the case may be. Lottery scheduling is scheduling is another mechanism which is based on lottery tickets. Every user is provided tickets based on their required share of processor execution. When there is need to perform scheduling, a lottery is held by executing a program for generating a random number from the set of tickets provided to all users. The user holding the winning ticket is allowed to execute. Below we will see the analytic evaluation of the CPU-Scheduling algorithms namely FCFS, SJF (non-preemptive and preemptive), Priority (non-preemptive and preemptive) and Round Robin (Time Slice-2msec). While solving Priority scheduling assumption made is lower number in priority assumes higher priority. We know that for analytic evaluation we require Gantt chart which is a pictorial representation of the manner in which the allocation of CPU took place for the various processes according to the CPU scheduling policy. Below we are having TABLE I which represent a snapshot for which we will be making an analytic evaluation.

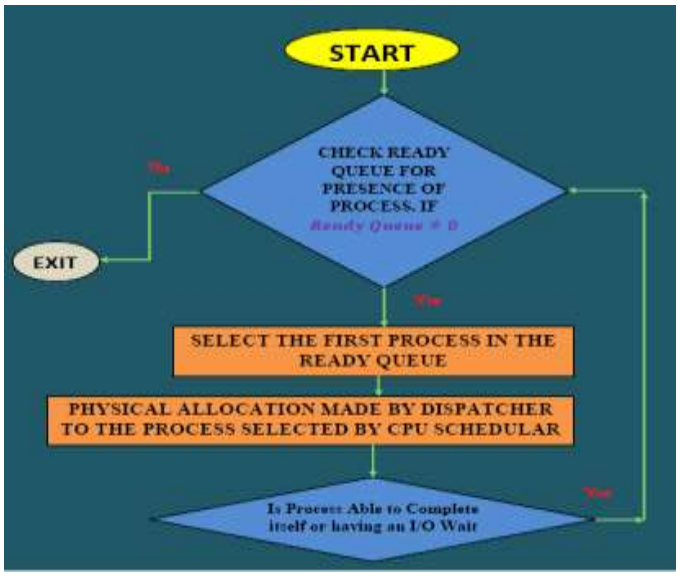

Fig. 1 Flowchart Chart for FCFS

The working of the algorithms is shown by means of Flowchart. In this section we show the four Flowcharts by means of figures Fig. 1, Fig. 2, Fig. 3 and Fig. 4. The four basic algorithms follow different policies for allocation of CPU. Also where FCFS is non-preemptive, SJF and Priority can be both Preemptive and non-preemptive whereas Round Robin in Preemptive having the selection process carried by the policy that the process at the top of the FIFO queue (Representing the Ready Queue) is used. FCFS is a fair policy and gives the first opportunity of execution to the process that arrived first. FCFS suffers from Convoy effect. SJF selects the process that is having the least value for the next CPU burst. SJF is a special case of Priority Scheduling.

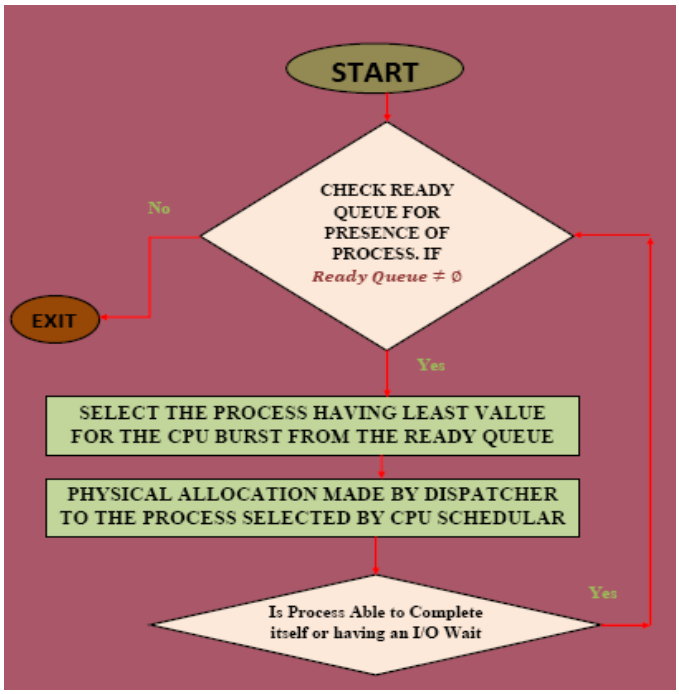

Fig. 2 Flowchart Chart for SJF(Non Preemptive)

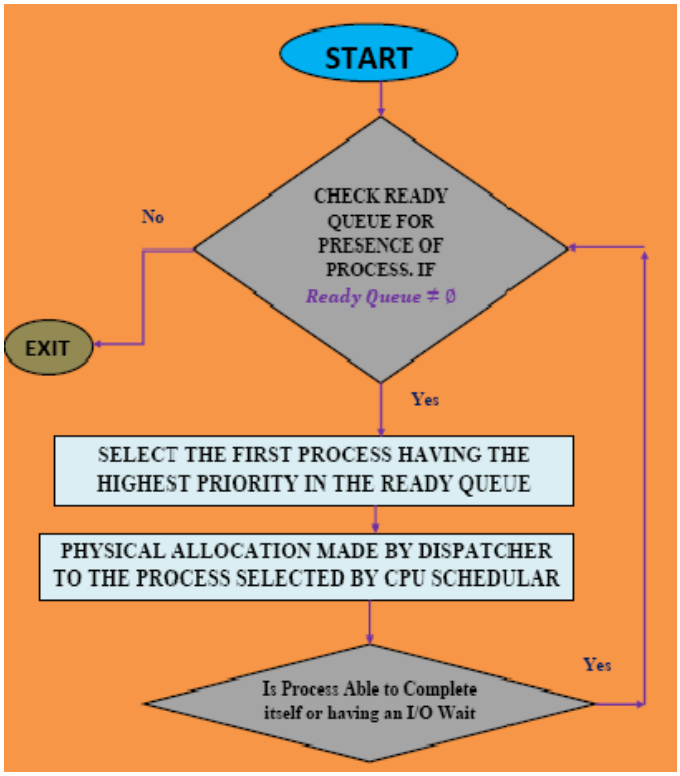

Fig. 3 Flowchart Chart for Priority(Non Preemptive)

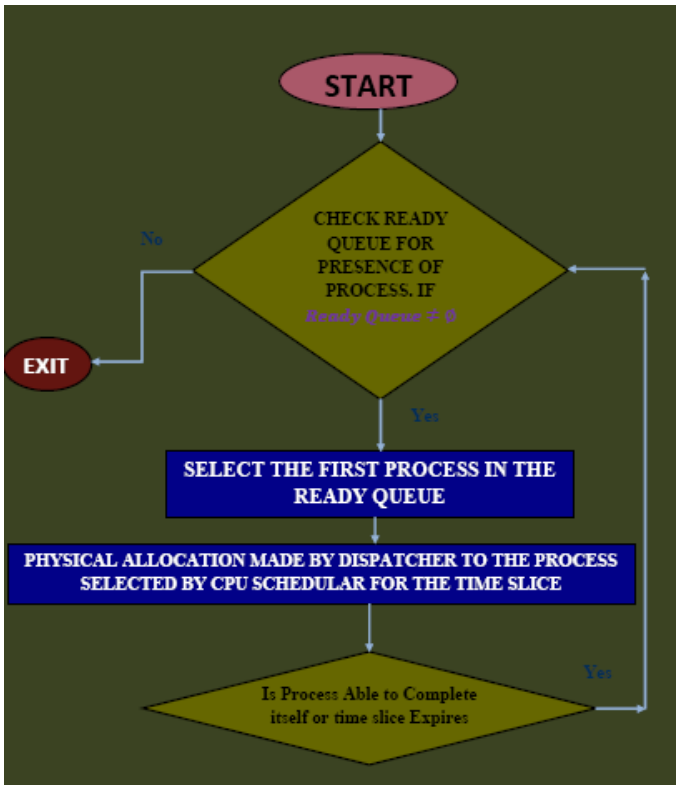

Fig. 4 Flowchart Chart for ROUND ROBIN

TABLE I
SNAPSHOT OF WORKLOAD ASSIGNED FOR MAKING VALUATION OF SCHEDULING ALGORITHMS FCFS, SJF, PRIORITY AND ROUND ROBIN

Process	Arrival Time in msec	CPU Burst Time in msec	Priority
P1	0	3	3
P2	1	7	1
P3	4	2	2
P4	6	4	2

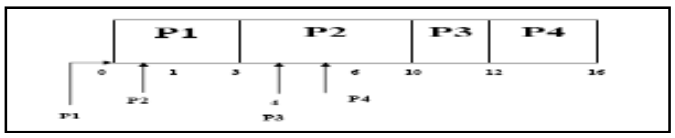

Fig. 5 Gantt Chart for FCFS

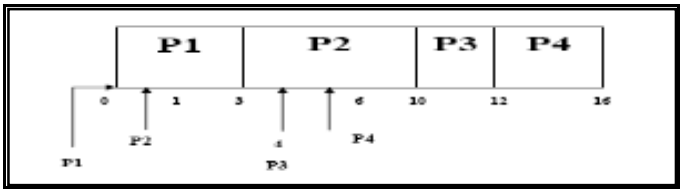

Fig. 6 Gantt Chart for SJF (Non preemptive)

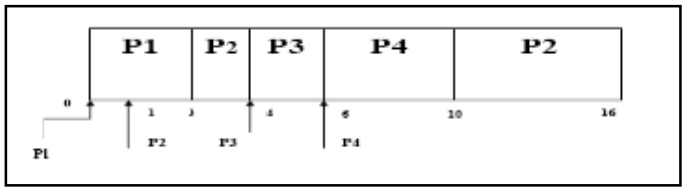

Fig. 7 Gantt Chart for SJF (Preemtive)

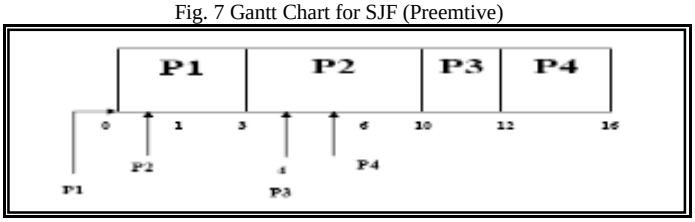

Fig. 8 Gantt Chart for Priority (Non Preemptive)

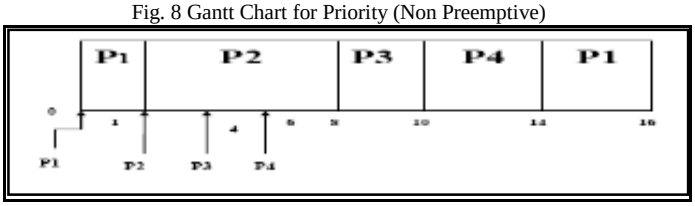

Fig. 9 Gantt Chart for Priority (Preemptive)

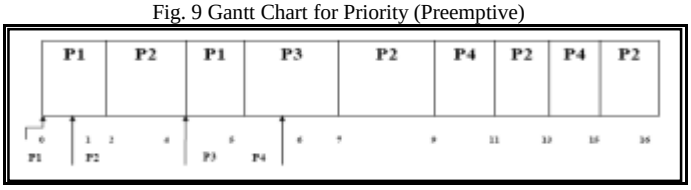

Fig. 10 Gantt Chart for Round Robin Scheduling (Time Slice 2 msec)

TABLE II
ATAT FOR THE ALGO'S WHOSE GANTT CHART IS GIVEN ABOVE

ALGORITHM	TAT OF P1	TAT OF P2	TAT OF P3	TAT OF P4	ATAT
FCFS	3	9	8	10	7.5
SJF (NON PREEMTIVE)	3	9	8	10	7.5
SJF (PREEMTIVE)	3	15	2	4	6
PRIORITY (NON PREEMTIVE)	3	9	8	10	7.5
PRIORITY (PREEMTIVE)	16	7	6	8	9.25
ROUND ROBIN	5	15	3	9	8

TABLE III
AWT FOR THE ALGO'S WHOSE GANTT CHART IS GIVEN ABOVE

ALGORITHM	WT OF P1	WT OF P2	WT OF P3	WT OF P4	AWT
FCFS	0	2	6	6	3.5
SJF (NON PREEMTIVE)	0	2	6	6	3.5
SJF (PREEMTIVE)	0	8	0	0	2
PRIORITY (NON PREEMTIVE)	0	2	6	6	3.5
PRIORITY (PREEMTIVE)	13	0	4	4	5.25
ROUND ROBIN	2	8	1	5	4

For making an evaluation through analytic model we will be making a calculation of Average Turn Around Time (ATAT), Average Waiting Time (AWT) and Average Response Time (ART). TABLE II, TABLE III and TABLE IV show the comparison of the values produced by the algorithms using analytical model. In the TABLES TAT represents Turn Around Time, WT represent Waiting Time and RT represents Response time. It is important to note that the values present in the TABLES are generated from the formulation that we discussed in Section V. Analytic model requires manual approach for every snapshot the full mathematical calculation is supposed to be made. Also it does not consider various practical aspects acknowledged in the working environment.

TABLE IV
ART FOR THE ALGO'S WHOSE GANTT CHART IS GIVEN ABOVE

ALGORITHM	RT OF P1	RT OF P2	RT OF P3	RT OF P4	ART
FCFS	0	2	6	6	3.5
SJF (NON PREEMTIVE)	0	2	6	6	3.5
SJF (PREEMTIVE)	0	2	0	0	0.5
PRIORITY (NON PREEMTIVE)	0	2	6	6	3.5
PRIORITY (PREEMTIVE)	0	0	4	4	2
ROUND ROBIN	0	1	1	3	1.25

VIII. SIMULATION IN C++

In this research paper we had made simulation of algorithms FCFS, SJF, Priority and Round Robin is made in C++. It is assumed that all processes arrive at the same time. The snapshots of the Algorithms are given in the section. All the simulations made are supplied the same input. The OS of the system in which the experiment is made is Windows 10 Home (64 bit). System Configuration is INTEL(R) CORE(TM) i5-3210 M CPU@2.50 GHz, RAM 4 GB Nanya Technology 1600 MHz.

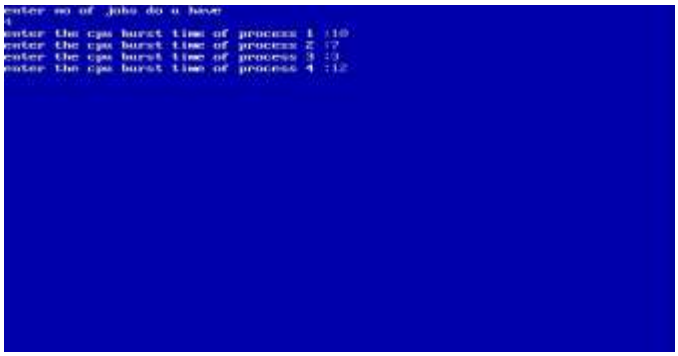

Fig. 11 Input given FCFS Scheduling

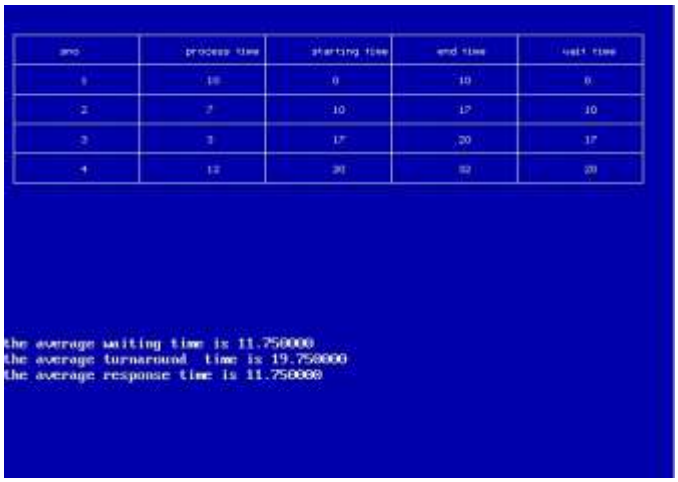

Fig. 12 Output obtained for given input in FCFS Scheduling

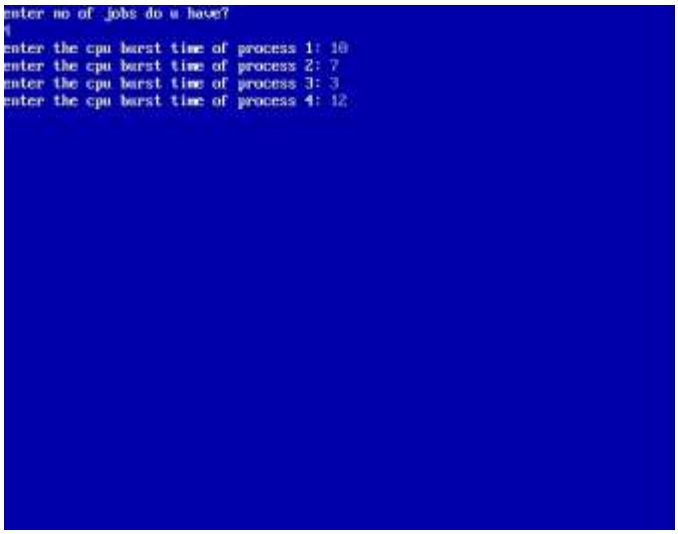

Fig. 13 Input Given to SJF Simulation

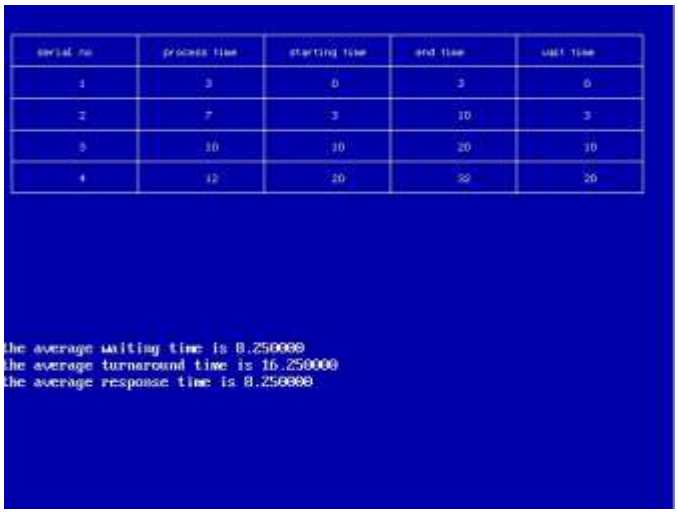

Fig. 14 Output Obtained for SJF Simulation

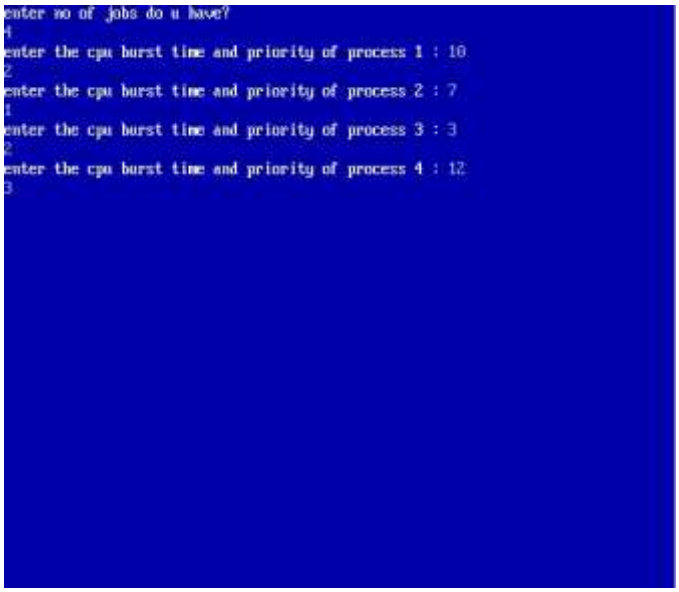

Fig. 15 Input Given to Priority Simulation

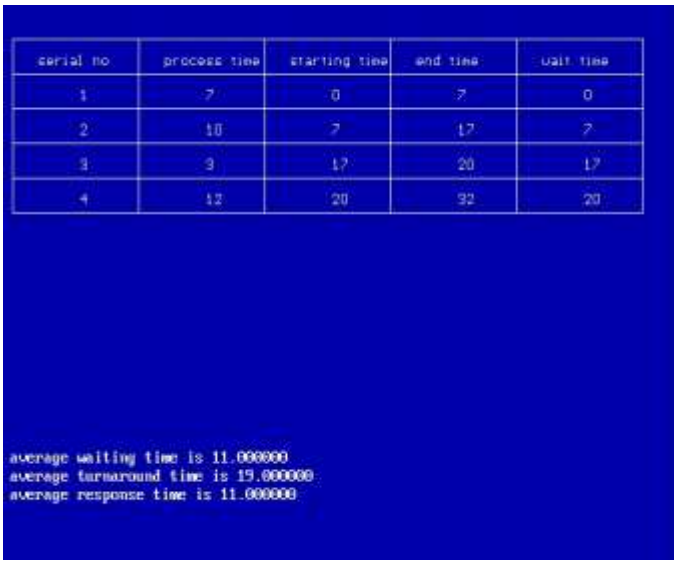

Fig. 16 Output obtained after providing the Input to Priority Simulation

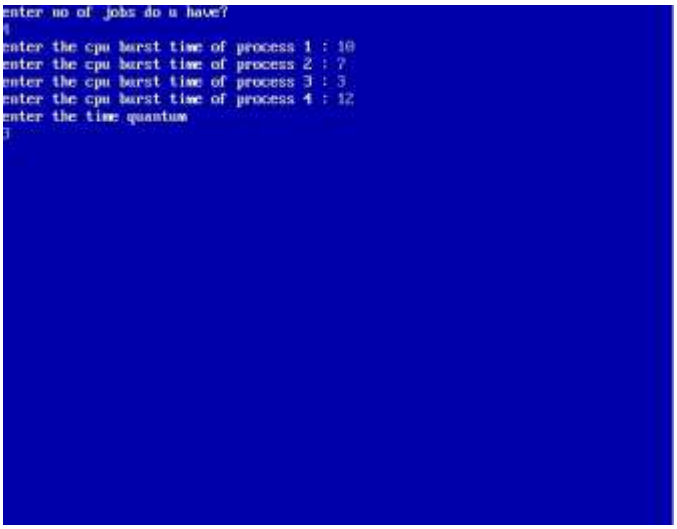

Fig. 17 Input given to Round Robin Simulation

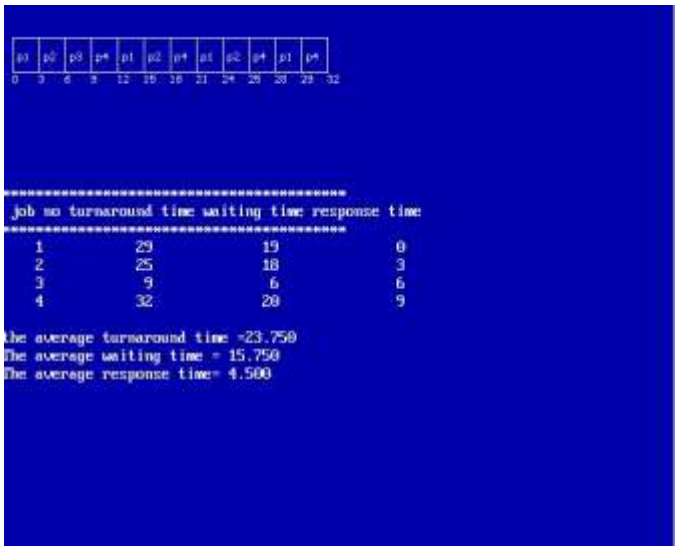

Fig. 18 Output obtained for Round Robin Simulation

The above mentioned algorithms are reconstructed without graphics constructs in GNU GCC Compiler having version 4.9 series. The outputs of which are given below.

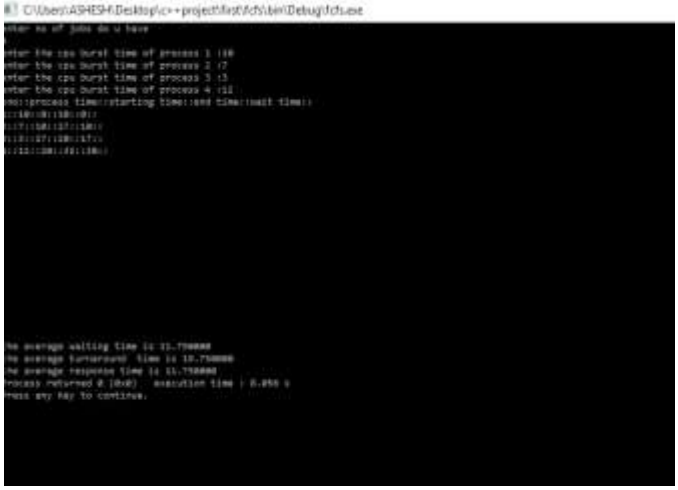

Fig. 19 Output obtained for FCFS Simulation

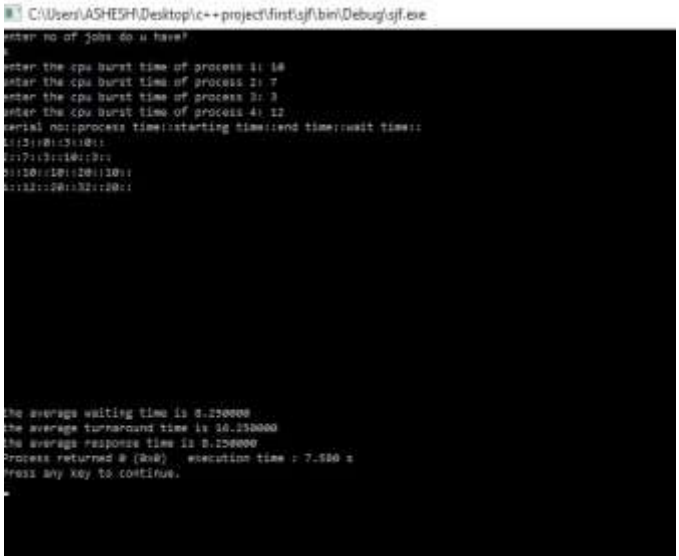

Fig. 20 Output obtained for SJF Simulation

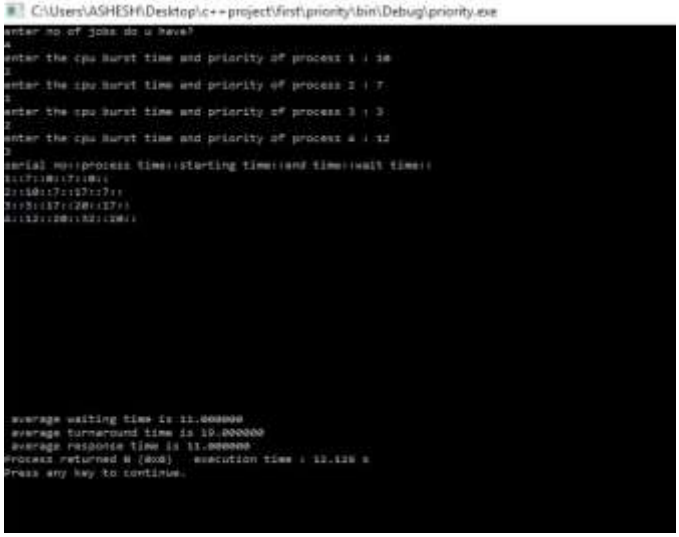

Fig. 21 Output obtained for Priority Simulation

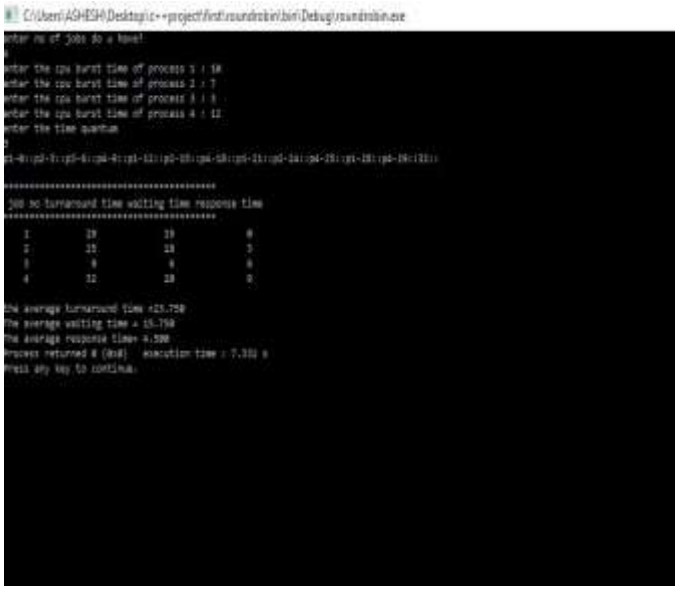

Fig. 22 Output obtained for RR Simulation

IX. Experiment and Result

From the simulation that we made on the Operating System Windows 10 Home (64-bit) having hardware configuration of RAM 4GB Nanya Technology 1600 MHz , INTEL (R) CORE (TM) i5- 3210 M CPU @ 2.50 GHz. The execution time obtained for the different simulations is given below. Simulation is a technique which uses the computing power of the computer. For Deterministic modeling every time the person required to make assessment was suppose to do manual calculation. In simulation Program is written which can give the same results which are generated by deterministic modeling at a faster pace and with a reduction in human effort.

TABLE V
EXPERIMENTAL RESULTS OBTAINED OF EXECUTION TIME

Time Taken For Execution By the Different Simulations Given below				
Snapshot Number	FCFS	SJF	Priority	Round Robin
1	8.056 sec	7.580 sec	12.126 sec	7.332 sec
2	8.339 sec	7.890 sec	13.270 sec	7.890 sec
3	8.556 sec	8.755 sec	14.845 sec	8.472 sec
4	7.758 sec	7.498 sec	13.076 sec	9.809 sec

TABLE VI
COMPLEXITY OF THE SIMULATIONS AND DATA STRUCTURES USED

Line of Codes Compiled for the different Simulations Implemented in C			
FCFS	SJF	Priority	Round Robin
764	785	786	799
Total Number of Integer variables including array count			
FCFS	SJF	Priority	Round Robin
51	52	60	113
Total Number of float variables including array count			
FCFS	SJF	Priority	Round Robin
1	1	1	1
Total Number of char variables including array count			
FCFS	SJF	Priority	Round Robin
131	131	130	181
Total Amount of Data Structure required in terms of variables in the Different types of Simulation in Bytes(Taking int as 2, char as 1 and float as 4)			
FCFS	SJF	Priority	Round Robin
51X2+1X4+1 X131=237	52X2+1X4+1X 131=239	60X2+1X4+1X130= 254	113X2+1X4+1X 181=408

X. Graphs On the basis of results obtained

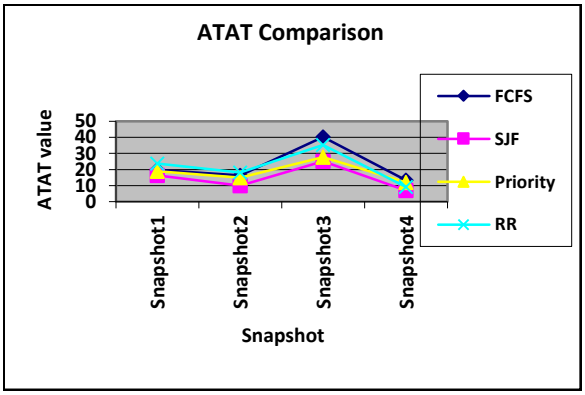

Fig. 23 Graph representing the Comparison of ATAT for different Snapshots.

The above graph shows the variation in the values obtained for four observations on the simulated algorithms. From the graph it is clear that SJF gives the minimum values whereas the maximum values are attained for FCFS. This gives a message that SJF policy yields better result in terms of performance when compared with FCFS, Priority and RR Scheduling.

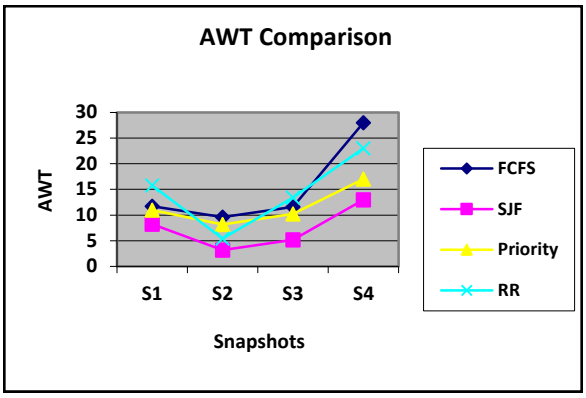

Fig. 24 Graph representing the Comparison of AWT for different Snapshots.

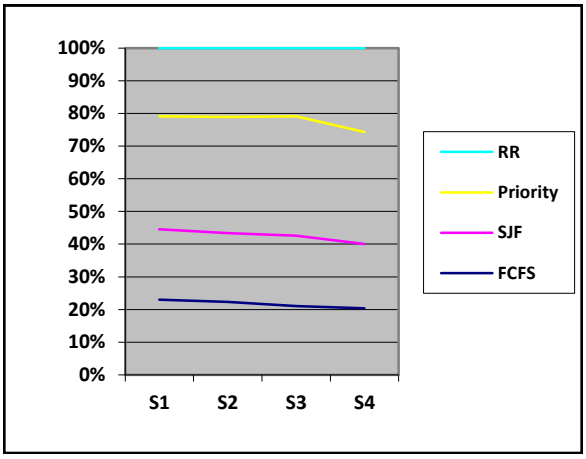

Fig. 25 Graph representing the Comparison of Execution Times of the various Simulations for different Snapshots.

XI. Conclusion

From Table I, II, III and IV the results obtain for the given snapshot shows that for ATAT the scheduling algorithm gives: Priority(Preemtive)>RR>Priority(Nonpreemptive)=SJF(Nonpreemptive)=FCFS>SJF (Preemptive). For AWT the result is as: Priority(Preemtive)>RR>Priority(Nonpreemptive)=SJF(Nonpreemptive)=FCFS>SJF (Preemptive) and for ART values are as follows:Priority(NonPreemtive)=SJF(Nonpreemptive)=FCFS> Priority(Preemptive)>RR>SJF (Preemptive).

The conclusion which we can draw from the two approaches whether it is Simulation or Deterministic modeling. The best result is given by SJF (Preemptive). Thus, the work that we did propose the usage of Shortest Job First for having optimum result. From Table VI we make a conclusion that Round Robin algorithm for its simulation requires maximum space for the data structure. Also the program complexity in terms of lines of codes is maximum when compared from other three approaches.

Acknowledgment

The work is done in Guru Ghasidas Vishwavidyalaya. I place my thanks to my mother, wife, brothers, sister and my student Ashesh for creating an environment that helped me in making the findings which I make during my research work..

References

[1] T.E. Anderson, B.N. Bershad, E.D. Lazowska and H.M. Levy, "Scheduler Activations: Effective kernel support for the user level Management of Parallelism," *ACM Transactions on Computer Systems*, vol 10, pp. 53-79, Feb, 1992.

[2] G.R. Andrews and F.B. Schneider, "Concepts and notations for concurrent programming," *Computing Surveys*, vol. 15, pp. 3-43, March 1983.

[3] R.D. Blumofe and C.E. Leiserson, "Scheduling Multithreaded Computations by Work Stealing," *Proc. 35th Annual Symp. on Foundations of Computer Science, IEEE,* pp. 356-368, Nov 1994.

[4] M. Buchanan and A. Chien, "Coordinated Thread Scheduling for Workstation Clusters under Windows NT," *The USENIX Windows NT Workshop, USENIX*, pp-47-54, 1997.

[5] A. Chandra, M. Adler, P. Goyal and P. Shenoy, "Surplus Fair SCheduling: A Proportional-share CPU Scheduling Algorithm for Symmetric Multiprocessors," *Proc. Fourth Symp. on Operating Systems Design and Implementation, USENIX,* pp. 45-58, 2000.

[6] K.J. Duda and D.R. Cheriton, "Borrowed-Virtual time (BVT) Scheduling: Supporting Latency-Sensitive Threads in a General-Purpose Scheduler," *Proc. 17th Symp. on Operating Systems Principles, ACM,* pp. 261-276, 1999.

[7] W.A. Wulf, "Performance monitors for Multiprogramming Systems," *Proceeding of the Second ACM Symposium on Operating System Principles*, pp. 175-181, October 1969.

[8] J.I. Schwartz and C. Weissman, "The SDC Time-sharing system revisited," *Proceeding of the ACM National Meeting*, pp. 263-271, Aug 1967.

[9] G. Henry, "The Fair Share Scheduler," *AT&T Bell Laboratories Technical Journal*, October 1984.

[10] E.W. Dijkstra, "Solution of a problem in Concurrent Programming Control," Communications of the ACM, vol. 8, no. 9, pp. 569, Sep 1965.