International Journal of Computer Science & Communication (ISSN: 0973-7391)
Volume 12• Issue 1 pp. 27-31 Sept 2020 - March 2021 www.csjournals.com

A New Approach for Shortest Job First (SJF) Simulation in OS

Dr. Satish Chandra Pandey[1], Ved Prakash[2]
Assistant Professor[1] ,Assistant Professor[2]
Department of Computer Science & Engineering Pratap Unniversity,Jaipur (Raj.) ,India
pandey.satishchandra@gmail.com, vedprakash10@gmail.com

Abstract: In a multiprogramming classification, various processes be present at the same time as in main memory. each one process alternates stuck between using a processor and waiting for a quantity of event to occur, such as the conclusion of an I/O operation. The processor or processors are kept busy by executing one process even as the others wait. The key to multiprogramming is scheduling. CPU scheduling deals with the complexity of deciding which of the processes in the set queue is to be allocated the CPU. Scheduling affects the performance of the structure since it determines which processes will wait and which will steps forward. Average Waiting time and Average Turnaround time scheduling criteria were used to evaluate the algorithms and it was discovered that Shortest Job First scheduling algorithm gives more optimal performance of scheduling processes than others scheduling algorithm. In this paper, simulation of scheduling algorithm Shortest Job First (SJF) is done over C.
Key Words: Simulation, Scheduling Algorithm, Multiprogramming, I/O Operation.

Introductions

CPU scheduling is vital because when we have numerous runnable processes, it can have a huge consequence on resource utilization and the generally performance of the system [1]. We have three types of schedulers:
Long-term scheduler – This kind of scheduler decides which jobs or processes would be admitted to the set queue. Also this Scheduler dictates what processes are to run on a scheme.
Mid-term Scheduler - One next type scheduler it's mid-term scheduler who removes process from main memory and moves on resulting memory.
Short-term Scheduler (also known as dispatcher) - The decision as to which available process will be executed by the processor. Dispatcher unit gives control of the CPU to the process preferred by the short-term scheduler.
In a scheduling process is the liability of scheduler to determine when a process moves from successively state to waiting state also scheduler passes a process from the ready state to the execution state [2].
CPU Scheduler at whatever time the CPU becomes vacant, the operating system must select one of the processes in the ready queue to be execute. The selection process is carried out the tiny term scheduler or CPU scheduler. The CPU scheduler selects a process from the processes in memory that are prepared to complete and allocates the CPU to that process. All the processes in the ready queue are lined up before you for a chance to run on the CPU. The records in the queues are normally process control blocks (PCB) of the processes.

Related Work

The scheduler algorithms suggest an never-ending field of lessons. Another study that is very interesting is [3] , Jerry Breecher describes the way that we can get a process attached to a processor .
Jain et al. [4] offered a Linear Data Model Based Study of Improved Round Robin CPU Scheduling algorithm with features of Shortest Job First scheduling with changeable time quantum. Abdulrahim et al. [5] projected algorithm compared with the other algorithms, produces negligible average waiting time (AWT), average turnaround time (ATAT), and number of context switches that adopt RR CPU scheduling.
Suranauwarat [6] use simulator in operating system to study CPU scheduling algorithms in an easier and a more efficient way. Sindhu et al. [7] proposed an algorithm which can handle all types of process with optimum scheduling criteria. Terry Regner & Craig Lacey [8] has introduced the concepts and essentials of the construction and functionality of operating systems. The idea of this object was to examine dissimilar scheduling algorithms in a simulated system.

Basic Concepts

In a sole processor system, only one process can run at a instance; any others have to wait until the CPU is free and can be reschedule. Then intention of multiprogramming is to have some processes operation at all instance, to maximize CPU deployment. The idea is moderately simple. A processor is executed until it must wait naturally for the completion of some I/O request. In a simple computer system, the CPU then just sits inactive. All this waiting time is wasted no useful work is accomplished.. When one process has to wait, the operating system takes the CPU missing from that process and gives the CPU to a different process. The basic idea is to keep the CPU busy as much as feasible by execute a process and then switch to an additional process. The CPU is, of course, one of the prime computer resources. Thus, its scheduling is innermost to operating system blueprint.

Primary CPU Scheduling Algorithms

CPU preparation deals with the problem of deciding which of the processes in the set queue is to be allocated the CPU. The Basic CPU Scheduling algorithms are First-Come, First Served, Round Robin Shortest Job First,.

First-Come-First-Served:

The simplest scheduling policy is first come first served (FCFS), also known as first in first out (FIFO) or a strict queuing scheme. As each process becomes ready, it joins the ready queue. When the currently running process causes to execute, the process that has been in the ready queue the longest is selected for running [9].

Round Robin:

A uncomplicated method to reduce the consequence that short jobs suffer with FCFS is to use preemptive based on a timepiece. The simplest such strategy is round robin. A clock disrupt is generated at interrupted intervals. When the interrupt occurs, the presently running process is positioned in the ready queue and the next ready job is selected on a FCFS origin. This technique is also known as time slicing, because each process is given a slice of time before being preempted [9].

Shortest Job First:

The process is owed to the CPU which has least burst time. A scheduler arranges the processes with the least burst time in head of the queue and longest burst time in tail of the queue. This requires advanced knowledge or estimations about the time required for a process to complete. This idea is illustrated in the Following Figure-1.

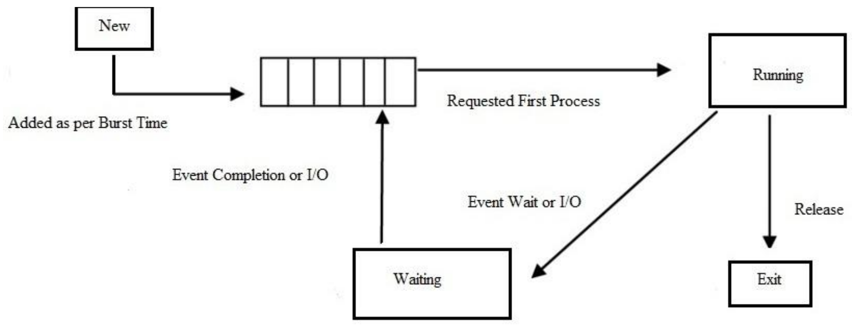


Figure-1 Process Diagram of Shortest Job First (SJF) Scheduling

Characteristics of Various Scheduling Policies

The routine of scheduling is linked to numerous parameters:

- ❖ Selection Function: Determines which process, among ready processes is selected next for execution.
- ❖ Decision Mode: It specifies the instant in time at which the variety function is exercised.
- ❖ Throughput: The scheduling strategy should attempt to maximize the number of processes accomplished per unit of time.
- ❖ Starvation: A condition in which a process is in definitely late since other processes are always given performance.
- ❖ CPU Usage: CPU should be kept full of activity at 100% of time.
- ❖ Turnaround Time: Time which is necessary for the completing of a process.
- ❖ Waiting Time: It is time that a process must wait in queue ready to be executed.
- ❖ Response Time : It is the time between the reception of the request made , to the first response. In order to have an optimal scheduling should be completed following conditions.

- ❖ Latency Time: That is the time it takes for the dispatcher to stop one procedure and start another running [10].

Experimental Phase

SJF is another scheduling algorithm in batch, which is based on the logic that a shorts process executed first. This category of algorithm is alienated into two types in preemptive and non-preemptive.

We have two types of scheduling algorithms preemptive and non-preemptive. It's preemptive in those cases where the implementation of a process can be interrupted by a different process (which may have higher priority), while non-preemptive when a process takes control of the CPU and do not leave it until the end of execution [11]. Yet the process which is ready on the queue has a inferior execution time he expects completion of the executions process.

The real difficulty with the SJF algorithm is, to know the length of the next CPU request. Scheduler adds on top of the ready queue the process which has the shortest burst time and the process with greater at the end.

A scheduler adds on the top of the queue a process who has a short execution time and those who have longer execution time into the tail of the queue. This requires highly developed knowledge or assumptions about the time desired to complete the process.

At the flow chart of SJF figure 5 is especially clear the logic, processes will be executed after they are selected preliminarily.

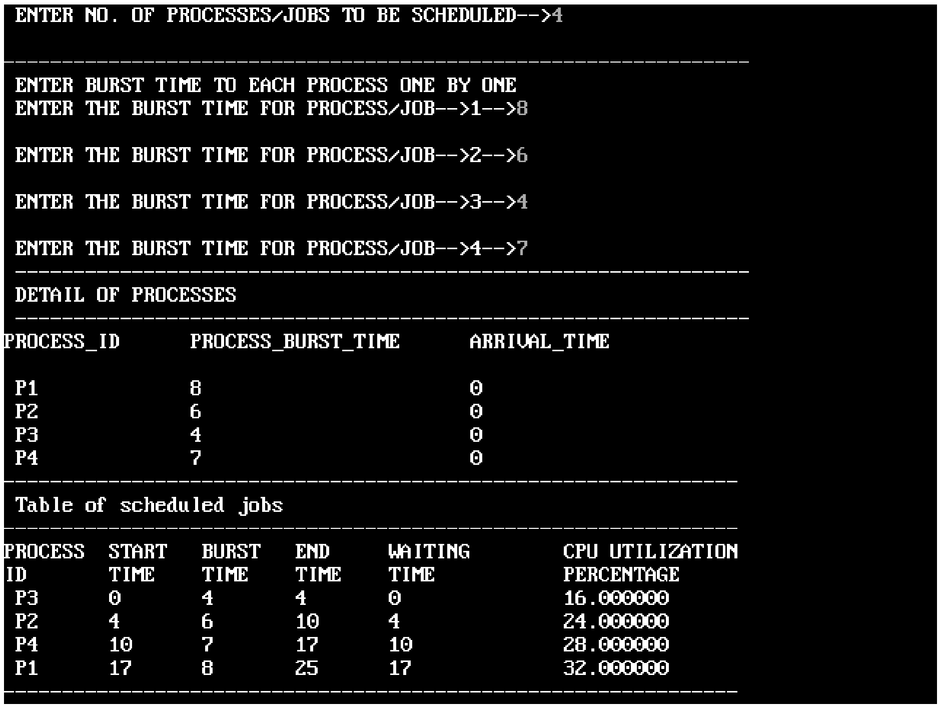
On the phase of the experiment we can calculate the average waiting time for 5 processes .Names and duration of execution for each process taken from the keyboard. I choose C language because it is closer to the assembler and to CPU this for having a parallelism with this topic.

Environment

In relation to the environment in which the experiment takes place i chose windows 7. Mentions that the type of OS does not involve the output of this experiment, it is an simulation on C we have not an interaction with the kernel or CPU.

Test Phase

The results received after execution of 4 processes with SJF are shown in Figure 2.



```
 ENTER NO. OF PROCESSES/JOBS TO BE SCHEDULED-->4

 ------------------------------------------------------------
 ENTER BURST TIME TO EACH PROCESS ONE BY ONE
 ENTER THE BURST TIME FOR PROCESS/JOB-->1-->8

 ENTER THE BURST TIME FOR PROCESS/JOB-->2-->6

 ENTER THE BURST TIME FOR PROCESS/JOB-->3-->4

 ENTER THE BURST TIME FOR PROCESS/JOB-->4-->7
 ------------------------------------------------------------
 DETAIL OF PROCESSES
 ------------------------------------------------------------
PROCESS_ID      PROCESS_BURST_TIME      ARRIVAL_TIME

 P1             8                       0
 P2             6                       0
 P3             4                       0
 P4             7                       0
-----------------------------------------------------------
 Table of scheduled jobs
-----------------------------------------------------------
PROCESS  START   BURST   END     WAITING        CPU UTILIZATION
ID       TIME    TIME    TIME    TIME           PERCENTAGE
 P3      0       4       4       0              16.000000
 P2      4       6       10      4              24.000000
 P4      10      7       17      10             28.000000
 P1      17      8       25      17             32.000000
-----------------------------------------------------------
```

Figure-2 Results of Various Process in Shortest Job First Scheduling (SJF)

Figure 2 shows the results of the execution of the SJF code, noticed an improvement in the average waiting time this because short processes executed at the beginning this brings. a short time waiting for long processes at the bottom of the queue.

Shortest Job First (SJF) Source Code:

SJF Simulator (Code Implementation)

```
#include<stdio.h>
 #include<conio.h>
   void main()    {
     int i,j,n,st[10],et[10],av[10],bt[10],wt[10],p[10],t,pos,tot=0,k=20;
     float avg_t, avg_total,pr[10],t1;
     clrscr();
     printf("\n ENTER NO. OF PROCESSES/JOBS TO BE SCHEDULED-->");
     scanf("%d",&n);
     printf("\n-----------------------------------------------------------------");
     printf("\n ENTER BURST TIME TO EACH PROCESS ONE BY ONE ");
    for(i=0;i<n;i++)      {
     printf("\n ENTER THE BURST TIME FOR PROCESS/JOB-->%d-->",i+1);
     scanf("%d",&bt[i]);
     p[i]=i+1;
     av[i]=0;
     tot=tot+bt[i];     }
     printf(" ---------------------------------------------------------------- \n");
     printf(" DETAIL OF PROCESSES \n");
     printf(" ---------------------------------------------------------------- \n");
     printf("PROCESS_ID      PROCESS_BURST_TIME      ARRIVAL_TIME\n");
    for(i=0;i<n;i++)
     printf("\n P%d      \t%d               \t%d",i+1,bt[i],av[i]);
    for( i=0;i<n; i++)     {
     for(j=i+1;j<n;j++)     {
     if(bt[i]>bt[j])     {
      t= bt[i];
      bt[i]=bt[j];
      bt[j]=t;
      t=p[i];
      p[i]=p[j];
      p[j]=t;      }      }
     if(i==0)      {
       st[0]=0;
       et[0]=bt[0]+st[0];
       wt[0]=0;      }
     else      {
       st[i]=et[i-1];
       et[i]=st[i]+bt[i];
       wt[i]=st[i];     }      }
     for(i=0;i<n;i++)
      pr[i]=bt[i]*(100.0/tot);
      printf("\n-----------------------------------------------------------------" );
      printf("\n Table of scheduled jobs" );
      printf("\n-----------------------------------------------------------------" );
      printf("\nPROCESS\t START \t BURST \t END  \t WAITING\tCPU UTILIZATION ");
      printf("\nID    \t TIME  \t TIME  \t TIME \t TIME    \tPERCENTAGE ");
     for(i=0;i<n;i++)
      printf("\n P%d \t %d    \t %d     \t %d  \t %d   \t \t%f  ", p[i] , st[i] , bt[i] , et[i] , wt[i], pr[i]);
      printf("\n-----------------------------------------------------------------" );
      getch();     }
```

Advantages of SJF

One of the major advantages of this algorithm is:

- SJF Is a good algorithm for the process that has a small execution time.
- SJF is normally used for long term scheduling.
- It reduces the average waiting time in excess of FIFO (First in First Out) algorithm.

- SJF method gives the lowest average waiting time for a specific set of processes.
- It is appropriate for the jobs running in batch, where run times are known in advance.

Disadvantages of SJF

One of the major disadvantages of this algorithm is:

- A process must wait in the queue if he has a large time of execution although he may be in the queue for a long time if on the queue come processes with short execution time
- It cannot be implemented at the level of short term CPU scheduling.
- Job completion time must be known earlier, but it is hard to predict.
- SJF can't be implemented for CPU scheduling for the short term. It is because there is no
- It is hard to know the length of the upcoming CPU request.

Conclusion & Future Scope

The present paper shows analysis on simulation of CPU scheduling policy via SJF.After execution of simulation codes of SJF we conclude that SJF algorithm is more competitive than FCFS If we talk about performance because it has minimal average waiting that is very important. After running the waiting times, turnaround time and average waiting time of this scheduling algorithm result has noticed minimal average waiting time. Simulation results shows that the proposed SJF scheduling algorithm is always giving better performance than FCFS. This algorithm can be implemented to improve the performance in the system in which SJF is a preferable choice.
Results concludes that the proposed algorithm is superior then commonly used algorithm with less waiting response time, less turnaround time and context switching; The SJF is better if the process comes to processor simultaneously. All algorithms is good, but the speed of the process depends on the processor load. Taking the base of proposed algorithm more improvement can be made the future.

References

[1] A Comparison between FCFS and Mixed Scheduling Alka Pant.
[2] A Comparative Study of CPU Scheduling Algorithms : Dr. R.B. Garg.
[3] Operating Systems , Scheduling : Jerry Breecher.
[4] Jain, S., Shukla, D. and Jain, R. Linear Data Model Based Study of Improved Round Robin CPU Scheduling algorithm. International Journal of Advanced Research in Computer and Communication Engineering, June 2015, Vol. 4, No. 6, pp.562-564.
[5] Abdulrahim, A., Abdullahi, S., E. and Sahalu J., B. A New Improved Round Robin (NIRR) CPU Scheduling. International Journal of computer Application. March 2014, Vol. 4, No.90, pp. 27-33.
[6] Suranauwarat, S. A CPU scheduling algorithm simulator. IEEE Proceedings (Frontiers in Education Conference - Global Engineering: Knowledge without Borders, Opportunities without Passports, 2007. FIE '07. 37th Annual), pp. 19 – 24.
[7] Sindhu, M., Rajkamal, R. and Vigneshwaran, P. An Optimum Multilevel CPU Scheduling Algorithm. IEEE (International Conference on Advances in Computer Engineering (ACE)), 2010, pp. 90 – 94.
[8] Terry Regner, craig lacy; An Introductory Study of Scheduling Algorithms; Feb 2005
[9] William Stallings, Operating Systems Internals and Design Principles, 5th Edition.
[10] Complex Scheduling (GOR-Publications) :Peter Brucker
[11] Modern Operating Systems : Andrey Tanenbaum.