IJCSN - International Journal of Computer Science and Network, Volume 3, Issue 6, December 2014
ISSN (Online): 2277-5420 **www.IJCSN.org**
Impact Factor: 0.274

Simulation of First Come First Served (FCFS) and Shortest Job First (SJF) Algorithms

[1] Nevila Xoxa, [2] Marjo Zotaj, [3] Igli Tafa, [4] Julian Fejzaj

[1, 4] University of Tirana, Faculty of Natural Science

[2, 3] Polytechnic University of Tirana, Faculty of Information and Technology

Abstract - Development of scheduling algorithms is directly related with the development of operating system which brings difficulties in implementation. Any modification on the scheduling algorithm will appear as modification on the operating system kernel code. Processor is an important source of cpu scheduling process, so it becomes very important on accomplishing of the operating system design goals. A delicate problem of the well-functioning of SO is the case when in CPU comes two or more processes which wants to be executed. Scheduling includes a range of mechanisms and policies that SO has to follows in order that all processes take the service. In this Paper we will discuss about two main batches algorithms, such as FCFS and SJF, and i will show a manner how to improve these algorithms in the future work.

Keywords - *CPU-Scheduling, Scheduler, FCFS, SJF.*

1. Introduction

CPU scheduling is important because when we have multiple runable processes, it can have a big effect on resource utilization and the overall performance of the system [2].

We have three types of schedulers:

A-***Long-term scheduler*** – This type of scheduler decides which jobs or processes would be admitted to the ready queue. Also this Scheduler dictates what processes are to run on a system.

B-*Mid-term Scheduler* - One second type scheduler it's mid-term scheduler who removes process from main memory and moves on secondary memory.

C-***Short-term Scheduler (also known as dispatcher)* -** Dispatcher module gives control of the CPU to the process selected by the short-term scheduler. Characteristic for dispatcher is the.

C.1 Latency time: that is the time it takes for the dispatcher to stop one process and start another running [8].

CPU scheduling deals with the problem of choosing a process from the ready queue to be executed by the CPU. In a scheduling process is the responsibility of scheduler to determine when a process moves from running state to waiting state also scheduler passes a process from the ready state to the execution state[3]. In general waiting queues use FIFO and LIFO policies. We have two types of scheduling algorithms preemptive and non-preemptive. It's preemptive in those cases where the execution of a process can be interrupted by another 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 [1].

The performance of scheduling is linked to several parameters:

1. ***CPU Usage****:* cpu should be kept busy at 100% of time .
2. ***Throughput*** : Number of process that typically ends executing in the given moment of time .
3. ***Turnaround time***: time which is necessary for the execution of a process.
4. ***Waiting time*****:** it is time that a process must wait in queue ready to be executed.
5. ***Response time*** **:** 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:

1.CPU-usage - MAX
2.Throughput - Max
3.Turnaround time- MIN
4.Waiting time - MIN
5.Response time – MIN[6].

2. Related Works

The scheduler algorithms offer an endless field of study. I am focused on algorithms batch, I chose these algorithms because windows is very prevalent in my country and I think that this paper will help those who study windows as OS .What that I concretely will deal in this paper is the simulation of two algorithms (in order to compare them) this topic well has studied by [2] on paper published in 2011, it has become an excellent study from both theory and practice , Alka Pant has achieved interesting conclusions regarding the: turnaround time, waiting time and response time time which has a great importance in batch systems. Another study that is very interesting is [13] , Jerry Breecher describes the way that we can get a process attached to a processor .

Another study in which I based my paper is [12] . In this paper Insup Lee explains the problems that appear during scheduling process ,he treats the scheduling process improvement, reducing turnaround time ,waiting time response time and. According to the [12] different applications require different optimization criteria as example : batch systems (throughput, turnaround time) , interactive system (response time, fairness, user expectation) . Dr. R.B. Garg in his study [3] explains very clearly idea of scheduling through figures. Others studies that I have studied very carefully are [1] , [4] and [6] which helped me to reach a clearer conclusion of waiting time which is important in batches sistmet. This simulation will be carried through a C code. After I realize this simulation and I calculated the time needed for comparison I have describe a way to improve these algorithms by performance.

3. Theory of Experiment

Below will give an overview of algorithms for both construction and operation. This part will be accompanied by tables and figures to make clear how the functionality of these two algorithms are.

3.1 FCFS

FCFS is the simplest algorithm on batch systems as for the building as for the functioning. Policy that uses this algorithm is that FIFO, so the first process which requires CPU takes service independently by the size of the process and this is illustrated in Figure 1 that is a four state diagram of FCFS. Is an algorithm non-preemptive so if process take control of CPU and don't leave it until the end of execution.

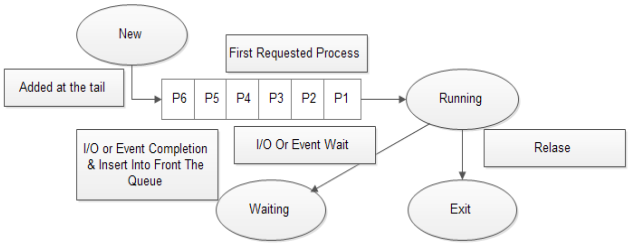

Fig. 1: First Come First Serve Scheduling

Figure 2 gives the flow chart of FCFS in which the C code is supported , which we will use for simulation[4]. There is a simple structure that represents the algorithm from the functional and construction. Processes are added one after another in the ready queue and executed in sequential order in time independently by the burst time they have [2].

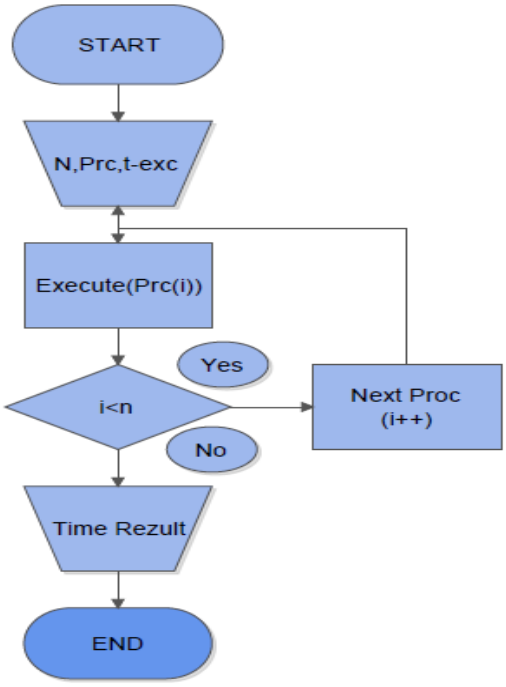

Fig. 2: First Come First Serve flow chart

Below I present the Gantt diagram for the three processes for which we have calculated average time.

Table .1 Process Execution

Process	Duration	Order	Arrival Time
P1	20	1	0
P2	7	2	0
P3	4	3	0

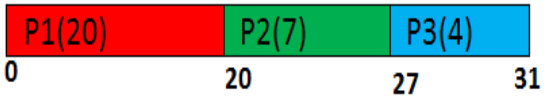

Fig. 3: Gantt chart for First Come First Serve

As we see from the Gantt Chart : P1 waiting time = 0
P2 waiting time = 20
P2 waiting time = 27
The average waiting time =(0+20+7)/3=9

Advantages
FCFS Is an algorithm relatively easy to understand and build, choosing of process for execution is very simple, enough to take the first in the queue and also the processes are added at the end of the queues.

Disadvantages
Through put is very low this because the long process want a long time to be executed, this leads to the so-called monopolization of cpu. Also turn around time and response time are relatively large for the same reason. Another disadvantage is that the average waiting time is quite large.

3.2 SJF

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

Preemptive : We interrupted the executions of process and executed coming process which has a smaller execution time .

Non-preemptive : Even if the process which is ready on the queue has a lower 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. This idea is illustrated in Figure 5.

A scheduler adds on the top of the queue a process who has a short executation time and those who have longer executation time into the tail of the queue. This requires advanced knowledge or assumptions about the time needed to complete the process [1].

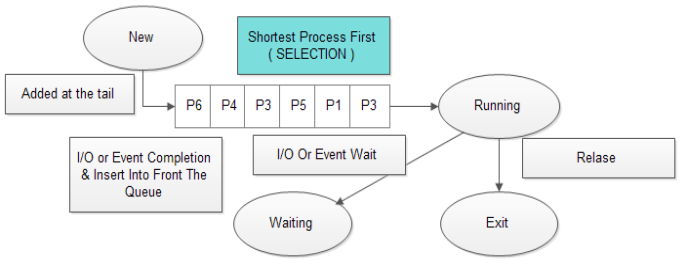

Fig. 4: Shortest Job First scheduling

Basis for part of the experiment will be the flow chart. At the flow chart of SJF figure 5 is very clear the logic, processes will be executed after they are selected preliminarily.

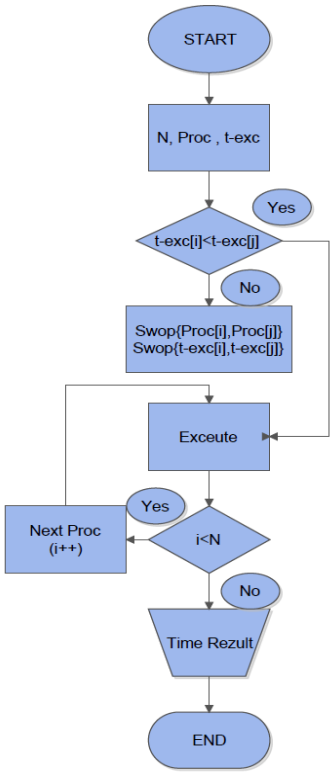

Fig. 5: First Come First Serve flow chart

If we repeat again the experiment that was conducted in case of FCFS we conclude that waiting time of SJF is less than FCFS [12].

Table.2 Process Execution

Process	Duration	Order	Arrival Time
P1	20	3	0
P2	7	2	0
P3	4	1	0

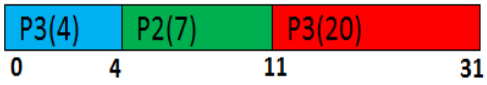

Fig. 6: Gantt Chart for Shortest Job First

As we see from the Gantt Chart : P3 waiting time = 0
P2 waiting time = 4
P1 waiting time = 11
The average waiting time =(0+4+7)/3=3.6]

Advantages
SJF Is a good algorithm for the process that has a small execution time. Regarding the average and waiting time is pretty advantage then FCFS .This because the execution of small processes in start brings a little time waiting for long processes.

Disadvantages
One of the major disadvantage 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 [6]. Complexity is another of bugs of this algorithm which stands in the selection of the next process from the CPU.

4. Experimental Phase

On the phase of the experiment I calculated the average waiting time for 5 processes .Names and duration of execution for each process taken from the keyboard. Both simulation realizes DEV - C + + as we mentioned in c language . I choose C language because it is closer to the assembler and to cpu, this for having a parallelism with the topic .

4.1 Environment

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

4.2 Test Phase

The results received after execution of 5 processes with FCFS are shown in Figure 3. And as mentioned in the theory noticed that the average waiting time is very high .

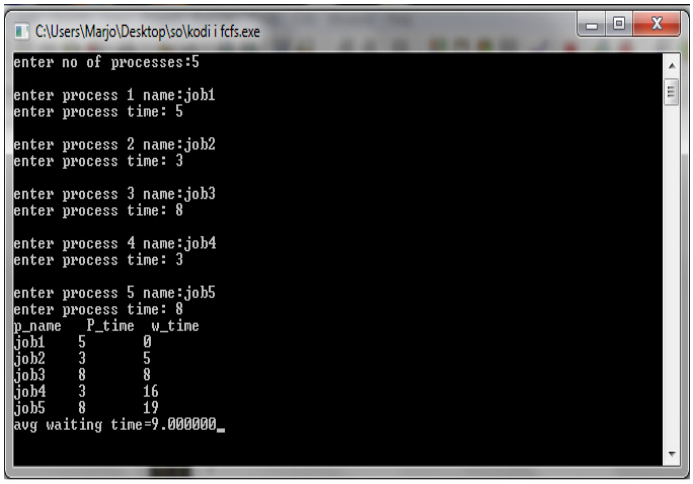

Fig. 7: Screenshot of my application (FCFS)

Figure 8 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.

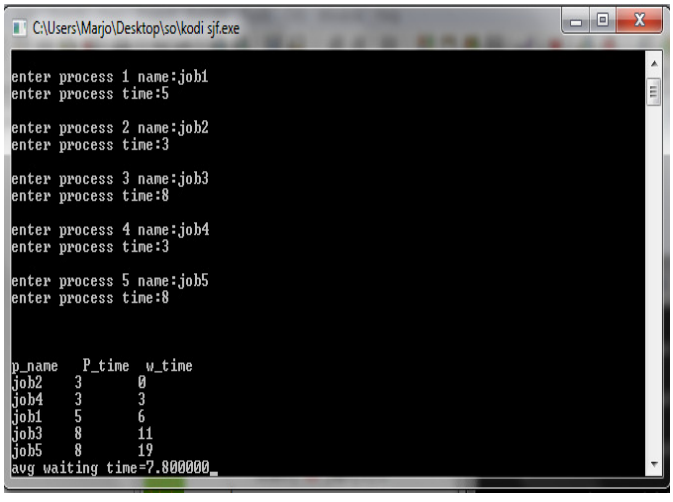

Fig. 8 : Screenshot of my application(SJF)

5. Conclusions

After execution of simulation codes of FCFS and SJF I come to the conclusion that SJF algorithm is more competitive than FCFS If we talk about performance because It has minimal average waiting that is very important.SJF main flaw is : a very long process to be not executed at the time required, thing that does not happen to FCFS algorithm.

6. Future Works

Like future works I am thinking a way to improve these two algorithms. One way is to improve FCFS in terms of average time is : we can select processes that come in queue according to a rule : to all processes who want to be executed waiting in the queue we choose processes combining a process with a long time of execution with another who has a short time of execution . According to some preliminary simulations that have made this leads to a reduction of average time with 10-11% a considerable time if we talk about 100 or more processes waiting in the queue . Related with SJF we can exclude the case: when a very long process on the ready queue is waiting for a long time or not be executed. That we can accomplish by using timers. For example, a long process which has a long time that is waiting his time expires this will certainly bring an interrupt which will then bring the execution process pending. In this way I think to remove disadvantages of FCFS and SJF .

References

[1] Modern Operating Systems : Andrey Tanenbaum.
[2] A Comparison between FCFS and Mixed Scheduling Alka Pant.
[3] A Comparative Study of CPU Scheduling Algorithms : Dr. R.B. Garg.
[4] W4118 Operating Systems : Junfeng Yang.
[5] Theory, Algorithms, and Systems : Michael L. Pinedo.
[6] Introduction to Algorithms : Thomas H. Cormen.
[7] Handbook of Scheduling: Algorithms, Models, and Performance Analysis Joseph Y-T. Leung.
[8] Complex Scheduling (GOR-Publications) :Peter Brucker.
[9] On Job Scheduling for HPC-Clustersand the dynP Scheduler :Achim Streit.
[10] Scheduling Algorithmes: Peter Pruckes.
[11] Theory of Scheduling: R.Conay.
[12] Softuare Systems , os overview-cpu scheduling: Insup Lee.
[13] Operating Systems , Scheduling : Jerry Breecher
[14] Comparative Study of Scheduling Algorithms in Operating System : Ankur Bhardwaj

Appendix

A-First Come First Served (FCFS) Source code .

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
 int main()
{ char pl[10][10];int pf[10]; int tot=0,wt[10],i,n; float avg=0;
printf("enter no of processes:");
scanf("%d",&n);
for(i=0;i<n;i++)
{ printf("\n enter process %d name:",i+1);
scanf("%s",&pl[i]);
 printf("enter process time: ");
 scanf("%d",&pf[i]);
} wt[0]=0;
for(i=1;i<n;i++)
{ wt[i]=wt[i-1]+pf[i-1];
tot=tot+wt[i];
} avg=tot/n;
printf("p_name\t P_time\t w_time\n");
for(i=0;i<n;i++)
printf("%s\t%d\t%d\n",pl[i],pf[i],wt[i]);
printf("avg waiting time=%f",avg);
 getch();
}
```

B-Shortest Job First (SJF) Source code

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{ char pl[10][5],temp[5];int tot=0,wt[10],pf[10],i,j,n,temp1;
float avg=0;
printf("enter no of processes:");
scanf("%d",&n);
 for(i=0;i<n;i++)
{
printf("\nenter process %d name:",i+1);


scanf("%s",&pl[i]);
 printf("enter process time:");
scanf("%d",&pf[i]);
```

```
 }
for(i=0;i<n-1;i++)
{
   for(j=i+1;j<n;j++)
 {
   if(pf[i]>pf[j])
{
temp1=pf[i];
pf[i]=pf[j];
pf[j]=temp1;
strcpy(temp,pl[i]);
strcpy(pl[i],pl[j]);
strcpy(pl[j],temp);
} } }
wt[0]=0;
 for(i=1;i<n;i++)
{ wt[i]=wt[i-1]+pf[i-1];
 tot=tot+wt[i];
 }
avg=(float)tot/n;
printf("\n\n\n");
printf("p_name\t P_time\t w_time\n");
for(i=0;i<n;i++)
printf("%s\t%d\t%d\n",pl[i],pf[i],wt[i]);
printf("avg waiting time=%f",avg);
  getch();}
```