ISSN:0975-9646

Mona Saini / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 2918-2923

Designing Various CPU Scheduling Techniques using SCILAB

Mona Saini

Department of Computer Science and Engineering
Krishna Institute of Engineering and Technology, Ghaziabad

1.ABSTRACT

Operating system's performance and throughput are highly affected by CPU scheduling. CPU Scheduling is the basis of multi-programmed operating system. The most important aspect of this paper is to represent scheduling algorithms in SCILAB language. Scilab is an open source, cross-platform numerical computational package and a high-level, numerically oriented programming language.

There are many scheduling algorithms available for a multi-programmed operating system. In this paper we are representing following scheduling algorithm, programmed using SCILAB. FCFS, SJFS, Round-Robin and deadlock bankers algorithm.

Keywords: CPU Scheduling, SCILAB, CPU Utilization, Throughput, Waiting Time, Response Time, Priority, SCFS, FJFS, Round Robin.

2. INTRODUCTION

In a single-processor system, only one process can run at a time; any others must wait until the CPU is free and can be rescheduled. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization [1]. Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources.

multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time-multiplexing. Part of the reason for using multiprogramming is that the operating system itself is implemented as one or more processes, so there must be a way for the operating system and application processes to share the CPU.

In this project we are using SCILAB for designing scheduling techniques because it is a open source, cross platform computational, high-level, numerically oriented programming language. The language provides an interpreted programming environment, with matrices as the main data type. By utilizing matrix-based computation, dynamic typing, and automatic memory management, many numerical problems may be expressed in a reduced number of code lines, as compared to similar solutions using traditional languages, such as Fortran, C, or C++. This allows users to rapidly construct models for a range of mathematical problems. While the language provides simple matrix operations such as multiplication, the Scilab package also provides a library of high-level operations such ascorrelation and complex multidimensional arithmetic. The software can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, and numerical optimization.

3. GOALS FOR SCHEDULING:

A CPU Scheduling scheme should be quick enough and satisfy following criteria.

3.1 CPU Utilization: It is the average fraction of time, during which the processor is busy It should keeps the CPU busy 100% of the time with useful work.

3.2 Throughput: It refers to the amount of work completed in a unit of time. The number of processes the system can execute in a period of time. It should maximizes the number of jobs processed per hour.

3.3 Waiting Time: The average period of time a process spends waiting. Waiting time may be expressed as turnaround time less the actual execution time For good scheduling technique, it should be as minimum as possible.

3.4 Turnaround time: The interval from the time of submission of a process to the time of completion is the turnaround time It should be minimum.

3.5 Response time: A scheduling technique is said to be good if its response time is quick. Response time is the time from submission of a request until the first response is produced

3.6 Priority: give preferential treatment to processes with higher priorities

3.7 Fairness: The primary function of any scheduling strategy is to make sure each process

4. VARIOUS SCHEDULING ALGORITHMS

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU.

4.1 First In First Serve Scheduling (FCFS): Just a FIFO queue, like customers waiting in line at the bank or the post office or at a copying machine. Dispatcher selects first job in queue and this job runs to completion of CPU burst. The advantages of FCFS is that it is simple and has low overhead. And has disadvantages of inappropriate for interactive systems and large fluctuations in average turnaround time are possible.

Process	Burst Time
P1	24
P2	3
P3	3

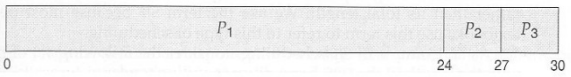


Figure-1 Process parameters and grant chart for FCFS.

In the first Gantt chart, process P1 arrives first. The average waiting time for the three processes is (0 + 24 +27) / 3 = 17.0 ms.

4.2 Shortest Job First Scheduling(SJFS) : The idea behind the SJF algorithm is to pick the quickest fastest little job that needs to be done, get it out of the way first, and then pick the next smallest fastest job to do next. arriving jobs inserted at proper position in queue, dispatcher selects shortest job (1st in queue) and runs to completion. Its advantage is that it is provably optimal from turnaround/waiting point of view. The disadvantages of SJF are that in general, it cannot be implemented, also starvation is possible, Can do it approximately: use exponential averaging to predict length of next CPU burst

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

Figure-2: Process parameters and grant chart for SJFS.

In the case above the average wait time is (0 + 3 + 9 + 16) / 4 = 7.0 ms, (as opposed to 10.25 ms for FCFS for the same processes.)

4.3 Priority Scheduling: Priority scheduling is a more general case of SJF, in which each job is assigned a priority and the job with the highest priority gets scheduled first. (SJF uses the inverse of the next expected burst time as its priority - The smaller the expected burst, the higher the priority.) Priorities can be assigned either internally or externally. Internal priorities are assigned by the OS using criteria such as average burst time, ratio of CPU to I/O activity, system resource use, and other factors available to the kernel. External priorities are assigned by users, based on the importance of the job, fees paid, politics, etc.

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

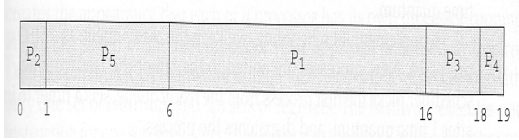


Figure-3: Process parameters and grant chart for Parity Scheduling.

The following Gantt chart is based upon these process burst times and priorities, and yields an average waiting time of 8.2 ms.

4.4 Round Robin Scheduling: Round robin scheduling is similar to FCFS scheduling, except that CPU bursts are assigned with limits called ***time quantum***. treat ready queue as circular, arriving jobs are placed at the end, dispatcher selects first job in queue and runs until completion of CPU burst, or until time quantum expires if quantum expires, job is again placed at end. The advantages of Round Robin are that it is simple, low overhead, works for interactive systems and has the following disadvantages if quantum is too small, there will be too much time wasted in context switching and if too large (i.e., longer than mean CPU burst), it approaches FCFS.

4.5 Banker's Algorithm (Deadlock Detection): In an operating system, a deadlock is a situation which occurs when a process or thread enters a waiting state because a resource requested by it is being held by another waiting process, which in turn is waiting for another resource. If a process is unable to change its state indefinitely because the resources requested by it are being used by another waiting process, then the system is said to be in a deadlock.

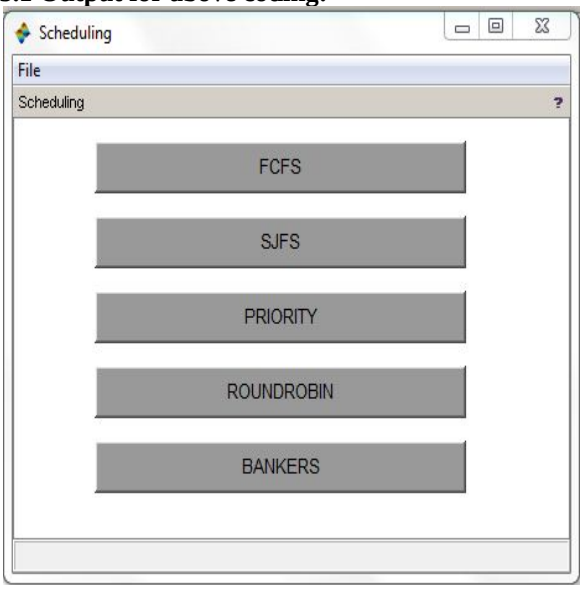
The Banker's algorithm is run by the operating system whenever a process requests resources. The algorithm **avoids** deadlock by denying or postponing the request if it determines that accepting the request could put the system in an unsafe state (one where deadlock could occur). When a new process enters a system, it must declare the maximum number of instances of each resource type that may not exceed the total number of resources in the system. Also, when a process gets all its requested resources it must return them in a finite amount of time.

5. CODING:

This coding helps to implement all these algorithms in one frame and every button have inbuilt coding for a particular algorithm.

```
function Schedule()

   defaultfont       = "arial";
   font_size = 12;

   // Figure creation
   //
=====================================
=====================================
===
   axes_w      = 450;
   axes_h      = 250;

   Schedule = scf(100001);

   // Remove Scilab graphics menus & toolbar
   delmenu(Schedule.figure_id, gettext("&File"));
   delmenu(Schedule.figure_id, gettext("&Tools"));
   delmenu(Schedule.figure_id, gettext("&Edit"));
   delmenu(Schedule.figure_id, gettext("&?"));
   toolbar(Schedule.figure_id, "off");

   Schedule.background      = -2;
   Schedule.color_map       = jetcolormap(128);
   Schedule.figure_position = [300 100];
   Schedule.figure_name     = gettext("Scheduling");

   // New menu
   h = uimenu( "parent",Schedule, "label",gettext("File"));
   uimenu( "parent",h, "label",gettext("Close"),
"callback","Schedule=get_figure_handle(100001);delete(Sc
hedule);", "tag","close_menu");

   sleep(500);
   Schedule.axes_size       = [axes_w axes_h];

button1 = uicontrol("parent",Schedule,
"style","pushbutton", "fontsize",13, "string","FCFS",
"units","pixels", "position",[65 200 300 35],
"callback","FCFS();", "tag","FCFS");
button1 = uicontrol("parent",Schedule,
"style","pushbutton", "fontsize",13, "string","SJFS",
"units","pixels", "position",[65 150 300 35],
"callback","SJFS();", "tag","SJFS");
button1 = uicontrol("parent",Schedule,
"style","pushbutton", "fontsize",13, "string","PRIORITY",
"units","pixels", "position",[65 100 300 35],
"callback","PRIORITY();", "tag","PRIORITY");
button1 = uicontrol("parent",Schedule,
"style","pushbutton", "fontsize",13,
"string","ROUNDROBIN", "units","pixels", "position",[65
50 300 35], "callback","ROUNDROBIN();",
"tag","ROUNDROBIN");
button1 = uicontrol("parent",Schedule,
"style","pushbutton", "fontsize",13, "string","BANKERS",
"units","pixels", "position",[65 0 300 35],
"callback","BANKERS();", "tag","BANKERS");

endfunction
```

5.1 Output for above coding:



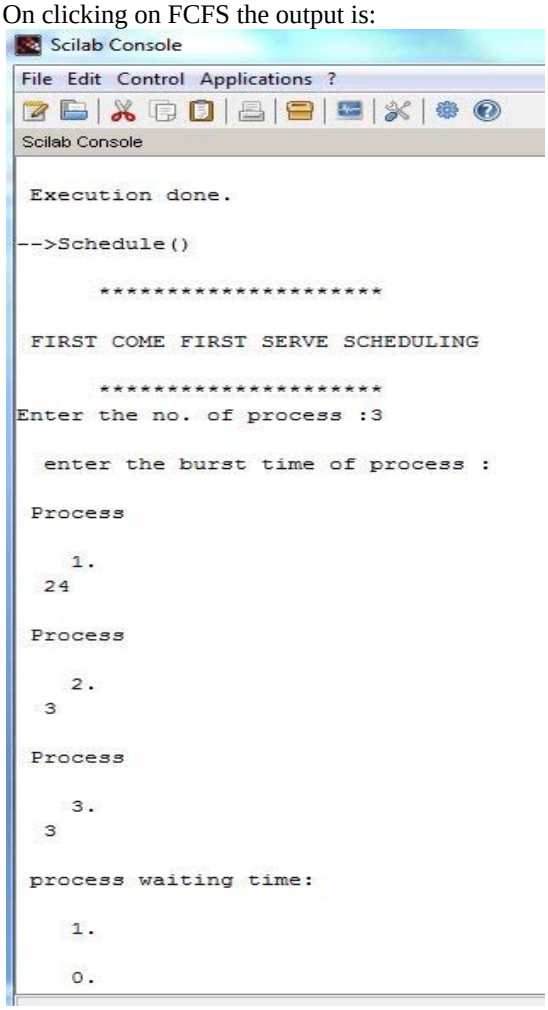
On Clicking at any of the scheduling technique, function corresponding to it is called. For example if we click of FCFS then following function will be initiated in SCILAB Interface.

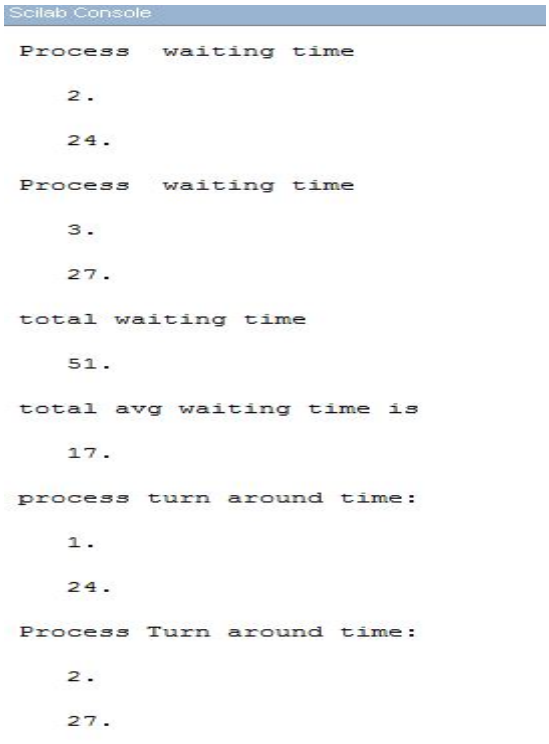
5.2 First Come First Serve:

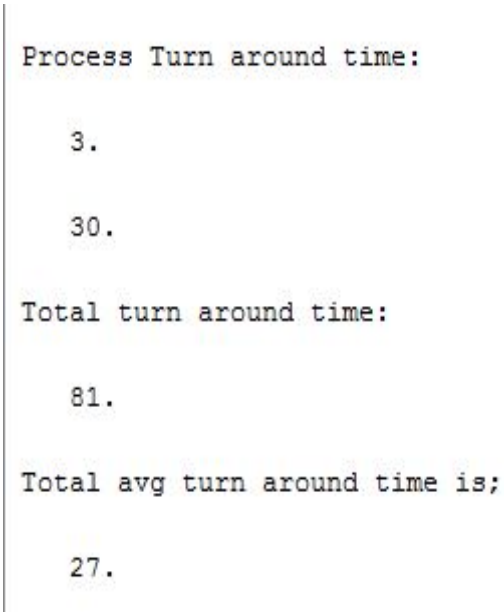
```
function FCFS()

n=input("Enter the no. of process :")
disp(" enter the burst time of process :")
for i=1:n
   disp(i,"Process")
   b(i)=input("  ")
         a(i)=i
        end
w(1)=0
avg=0
disp(w(1),a(1),"process waiting time:")
for i= 2:n
w(i)=b(i-1)+w(i-1)
disp(w(i),a(i),"Process  waiting time")
avg=avg+w(i)
end
disp(avg,"total waiting time")
disp(avg/n,"total avg waiting time is")
tat(1)=b(1)
avg1=b(1)
disp(tat(1),a(1),"process turn around time:")
for k= 2:n
tat(k)=tat(k-1)+b(k)
disp(tat(k),a(k),"Process Turn around time:")
avg1=avg1+tat(k)
end
disp(avg1,"Total turn around time: ")
disp(avg1/n,"Total avg turn around time is; ")
exec('C:\Users\Lovepreet\Desktop\mona saini prjct\new
prg\fcfs.sci', -1)
 endfunction
```

FCFS OUTPUT:

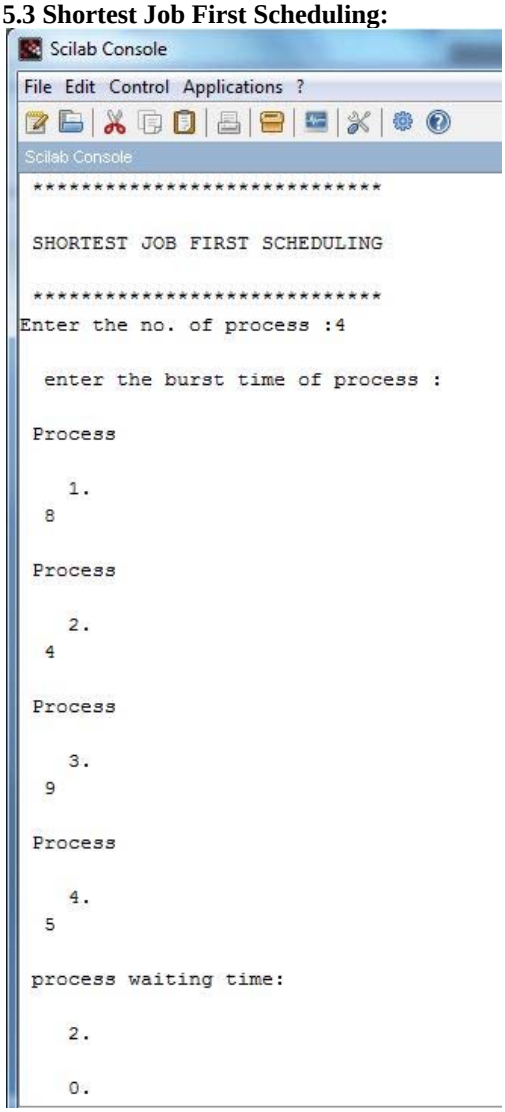
On clicking on FCFS the output is:





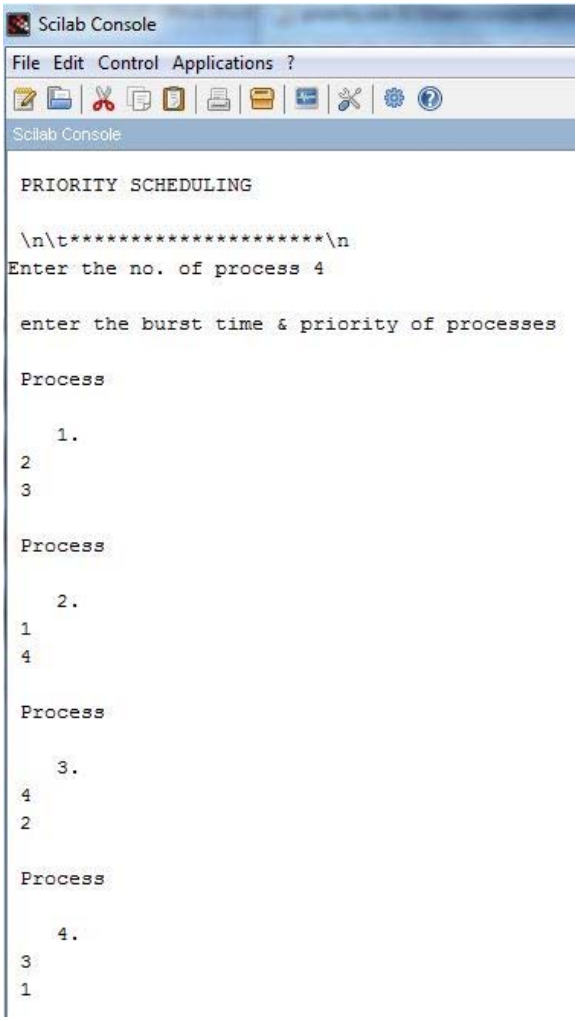


5.3 Shortest Job First Scheduling:



```
Scilab Console
Process  waiting time
   4.
   4.
Process  waiting time
   1.
   9.
Process  waiting time
   3.
   17.
total waiting time
   30.
total avg waiting time is
   7.5
process turn around time:
   2.
   4.
Process Turn around time:
   4.
   9.
Process Turn around time:
   1.
   17.
Process Turn around time:
   3.
   26.
Total turn around time:
   56.
Total avg turn around time is;
   14.
```

5.4 PARITY SCHEDULING:




```
Scilab Console
process waiting time
    4.
    0.
Process waiting time
    3.
    3.
Process waiting time
    1.
    7.
Process waiting time
    2.
    9.
total waiting time
    19.
total avg waiting time is
    4.75
```

```
Scilab Console
process turn around time

   3.

   4.

Process turn around time

   7.

   3.

Process turn around time

   9.

   1.

Process turn around time

   10.

   2.

total turn around time

   29.

total avg turn around time is

   7.25
```

6. CONCLUSION AND FUTURE WORK:

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. 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. This paper describes non-primitive scheduling and our future work is to work on primitive scheme of CPU scheduling in Scilab.

REFRENCES:

Neetu Goel, R.B. Garg| A Comparative Study of CPU Scheduling Algorithms, International Journal of Graphics & Image Processing |Vol 2|issue 4|November 2012

M. Spuri and G. Buttazzo. Efficient aperiodic service under earliest deadline scheduling. In IEEE Real-Time Systems Symposium, December 1994.

Z. Deng, J. W. S. Liu, and J. Sun. A scheme for scheduling hard real-time applications in open system environment,1997.

J. Banks, J. S. Carson J.S., Nelson B. L., Nicol D.M., (2005). Discrete-Event System Simulation Fourth Edition.

M. Cosnard and M. Loi. Automatic Task Graph GenerationTechniques. Parallel Processing Letters, 5(4):527–538,1995.

http://en.wikipedia.org./wiki/scheduling.

Silberchatz, Galvin and Gagne, 2003. Operating systems concepts.

D.M. Dhamdhere operating Systems A Concept Based Approach, Second edition, Tata McGraw-Hill, 2006.

Y. Zhang and R. West. Process-aware interrupt scheduling and accounting. In Proceedings of the 27th IEEE Real Time Systems Symposium, December 2006.