REPORT

CPU SCHEDULING SIMULATOR

Group 7

Group Members

Aayushi Agarwal(Y13UC007)

Anshika Mittal(Y13UC036)

Apurva Singh (Y13UC043)

Bhawna Rawat(Y13UC068)

Diksha Bhateja(Y13UC090)

The project has been done under the guidance of Dr. Subrat K. Dash and Prof Sunil Kumar.

Abstract

CPU scheduling falls under the **process management of Operating System**. Under this various process (programs in execution) utilize the CPU time and other resources by switching through some pre-given condition. By switching the CPU among processes, the operating system can make the computer more productive. The objective of multiprogramming is to have some process running at all times, in order to maximize CPU utilization.

Through this project we want to present a simulator to our users, **that uses graphical representation to convey the concepts of various scheduling algorithms for a single CPU**. It allows the user to test and analyze the various scheduling algorithm, through graphical user interface of the simulator and increase their understanding of the concepts and decide which scheduling algorithm suits their requirement.

The simulator graphically depicts each process from the starting point to the end time and what the process is doing against time.

INTRODUCTION

Scheduling is the method by which work specified by some means is assigned to resources that complete the work. The operating system has two scheduler, the short term Scheduler selects from among the processes n ready queue and allocates the CPU to one of them .the following are the algorithm that are implemented by the system

First Come First Serve (FCFS)

This non-preemptive scheduling algorithm follows the first-in, first-out (FIFO) policy. As each process becomes ready, it joins the ready queue. When the current running process finishes execution, the oldest process in the ready queue is selected to run next.

Round Robin

This scheduling policy (aka time-slicing) gives each process a slice of time before being preempted. As each process becomes ready, it joins the ready queue. A clock interrupt is generated at periodic intervals. When the interrupt occurs, the currently running process is preempted, and the oldest process in the ready queue is selected to run next. The time interval between each interrupt may vary.

Shortest Job First (Shortest Process Next)

This non-preemptive scheduling algorithm favors processes with the shortest expected process time. As each process becomes ready, it joins the ready queue. When the current running process finishes execution, the process in the ready queue with the shortest expected processing time (aka service time) is selected to run next.

Shortest Job First (Remaining Time)

This preemptive scheduling algorithm favors processes with the shortest remaining expected process time. As each process becomes ready, it joins the ready queue. This triggers an interrupt which preempts the current running process back into the ready queue. The process in the ready queue with the shortest remaining service time is selected to run next.

Feedback

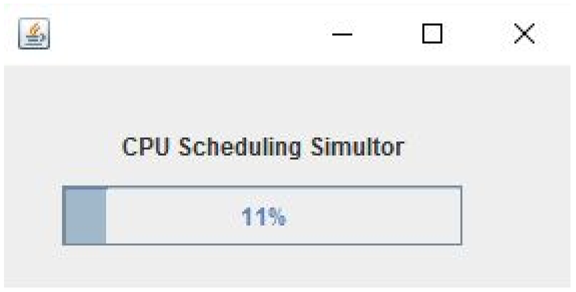
This preemptive scheduling algorithm favors shorter jobs by penalizing processes that have been running longer. It achieves this by using a multi level priority queues. When a process becomes ready, it joins the first level ready queue. After each subsequent execution interval, it is relegated to the next lower-priority queue until it joins the bottom-most level queue. At each interrupt, the process in the highest priority queue is selected to run next; the processes in the lowest priority queue are scheduled using round robin algorithm. The time interval between interrupts may vary, and the levels of ready queues may also vary.

<u>DESIGN SYSTEM AND IMPLEMENTATION:</u>

The software User Interface (front end) has been created with GUI java Swings, the output file comprises of an animated Gantt chart, pie chart created using applets and an integration between matlab and java (matlab control function).

The software has been developed on the following platform Eclipse and Matlab.

First of all, a progress bar is viewed showing that the simulator has started. Followed by a Welcome Screen.



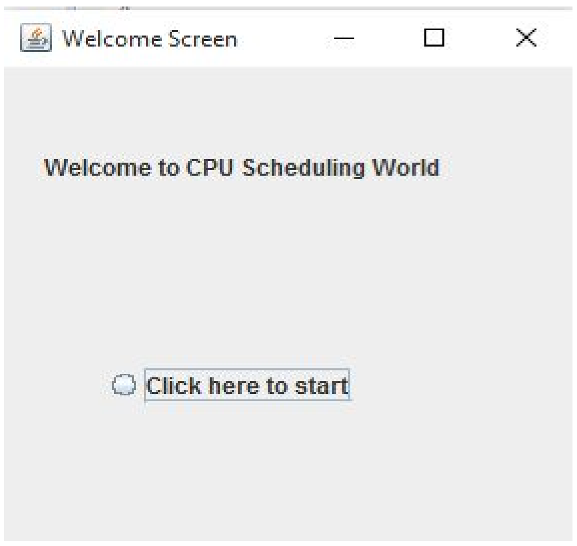


Figure 1 and 2 : Welcome Screens

<u>Main Input Window:</u>

The graphical user interface will include a main window that holds scheduling algorithm selection from an option list and a command for launching the sub window which holds the methods to read specific data (inputs) in the subsequent window the system allows the user to either choose a file(through browsing) or enter the test cases in a table provided by the user interface. The file selected through browsing should be a text file. The test cases generated through table are transferred into a file. The files in both the cases are read. All this have been implemented using **file handling.** The test cases are then scheduled through various scheduling algorithm:

-First Come First Serve

- Shortest Job Priority (Primitive)

-Shortest Job Priority (Non Primitive)

-Round Robin

-Feedback

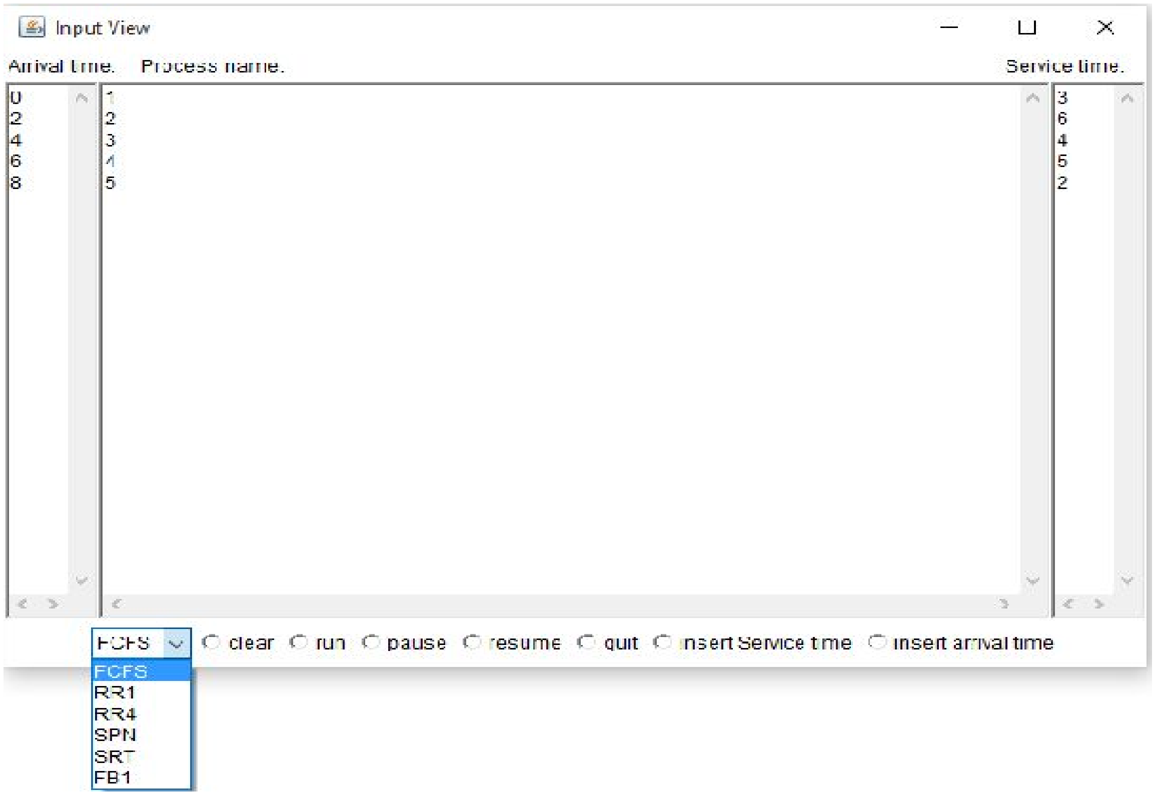
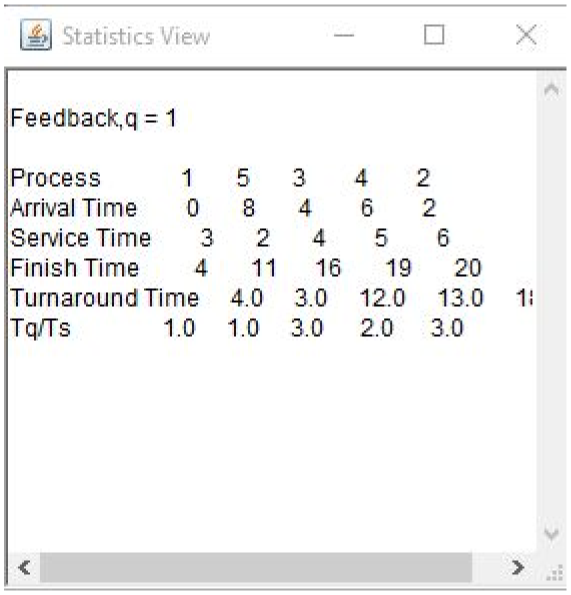


Figure 3: The Input Window

Once the scheduling algorithm starts the user is provided with various states of the processes such as start, suspend, dispatch, terminate and resume. If a user selects one of the above states in middle of execution then the system will behave affect the scheduling with the command and the visualization will change according to it.

When the user wants to restart the scheduling he has only to press the restart command then the system will end the current scheduling and set it to the beginning of a new scheduling, when a user presses the exit command all the processes and threads will terminate and the system will release all resources. When the user wants to create a new process the user selects the add command and a new process is created. The user may even create a thread for a certain process by selecting the process and pressing create thread.

 The software analysis the given test cases and will give the user Individual turnaround time , average turnaround time, for each of the desired scheduling. Furthermore if a single test case is run under more than one schedule algorithms then it will display the optimized scheduling algorithm according to the average turnaround time.



Statistics View

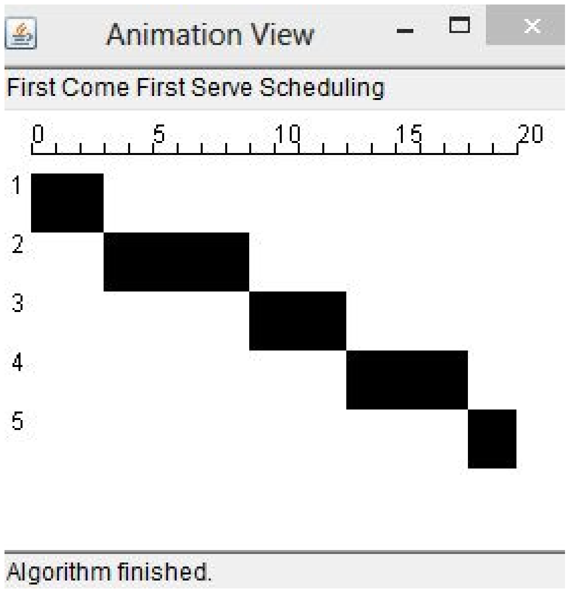
```
Feedback,q = 1

Process          1     5     3     4     2
Arrival Time     0     8     4     6     2
Service Time       3     2     4     5     6
Finish Time        4     11    16    19    20
Turnaround Time  4.0   3.0   12.0   13.0   1
Tq/Ts          1.0   1.0   3.0   2.0   3.0
```

Figure 2: The Statistics View Window

A Real time animated Gantt chart dynamically plots the bar graph for the given test cases.

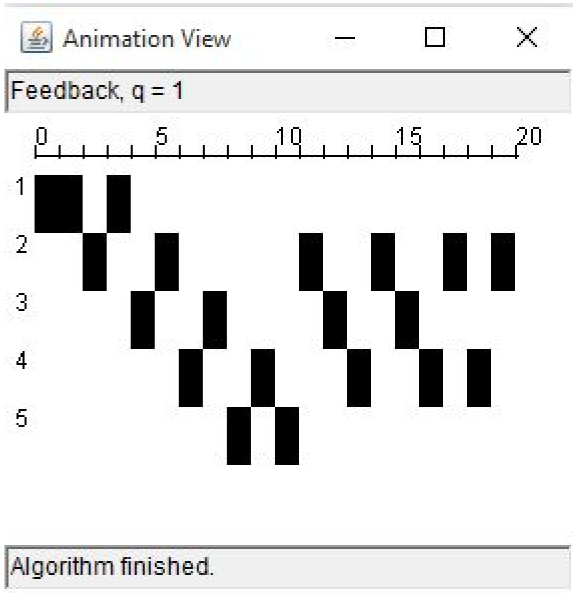
In the Gantt chart the title of the currently/previously running CPU algorithm is displayed across the top panel. The bottom panel traces the clock time and the status of the algorithm. The main panel performs the animation by establishing a timeline as the top horizontal axis and the process names as the vertical-left axis, and dynamically plots the bar graph. The entire Animation view is un editable since user input is not needed.



Animation View

First Come First Serve Scheduling

Algorithm finished.



Animation View

Feedback, q = 1

Algorithm finished.

Figure 4 and 5: Real time animated Gantt Chart

A pie chart is also constructed with the burst time of each process as its inputs and the ratio of above input as output. The chart has been constructed in Matlab using the matlab control function in java.

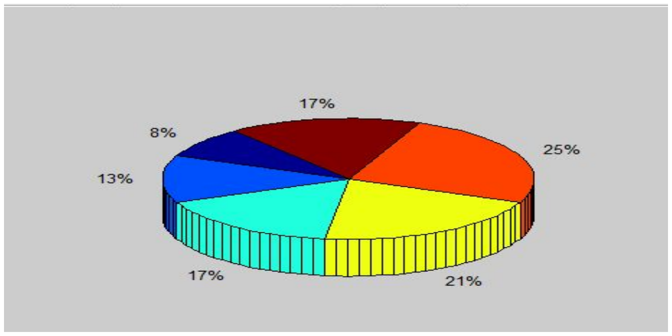


Figure 6: Pie chart describing the Burst Time of each processes

Inference

When a number of test case have been run, following inferences were made

1. Various scheduling algorithm according to their average waiting time in the ascending order:

- Shortest job First(SJF) (minimum)
- Round robin
- Multilevel Queue(Feedback)
- First come First serve (maximum)

2. Shortcomings of the various Scheduling Algorithms

In SJF the smallest job always get executed first which also leads starvation for large jobs and also the resources are underutilized.

IN FCFS the smallest jobs waits in the queue for a long time till the large job finishes and so the resources also underutilized.

In Round robin the jobs are not finished within the scheduled time which also leads to performance degradation and increases cost.

3. The order of complexity of the system depends upon the burst time of process.

CONCLUSIONS

1. This system is built to show how scheduling algorithms are work in visual way and show how process are executing.

2. This system show process acts in different process states and show the difference between schedule algorithms.

3. From this system we obtain easy tools for the user to easy manage process and easy tools to the user to add or remove process and show the effect.

4. Easy tools to show how different process scheduling are working and the user have better understand the process schedule.

REFERENCES

1. http://www.iasj.net/iasj?func=fulltext&aId=37803
2. http://courses.cs.vt.edu/~cs3204/spring2000/henry/projects/Project1/Spec1.PDF