Uniprocessor Process Scheduling Simulator

Problem Statement: Develop a uniprocessor scheduling simulator.

Abstract:

When more than one process is runable, the operating system must decide which one first. Scheduling refers to a set of policies and mechanisms built into the operating system that govern the order in which work to be done by a computer is completed. Many objectives must be considered in the design of a scheduling discipline. In particular, a scheduler should consider fairness, efficiency, response time, turnaround time, throughput, etc., Some of these goals depends on the system one is using for example batch system, interactive system or real-time system, etc. but there are also some goals that are desirable in all systems. A scheduling discipline is preemptive if, once a process has been given the CPU can taken away. RR is an example of a preemptive scheduling discipline Among the non preemptive processes the most commonly used scheduling policies are FCFS and SRTF.

Introduction:

We have built a process scheduling simulator when takes into input the jobs and their respective Arrival Time, I/O time and CPU time and returns the output corresponding to the following three strategies:

i) Round Robin Algorithm
ii) First-Come-First-Served Algorithm
iii) Shortest Remaining Time First Algorithm

In the output, the status of each of the queues, i.e. New, Ready, Blocked and Running is displayed at different instance if time after running the code on the input.

A GUI has been made to improve the user experience and make it easier for the user to decide the strategies he/she actually wants to compare for the same input data set. The simulator simulates the functioning of these algorithms inside the Operating System. The algorithms have been modified to work according to the Jobs having different sets of I/O and CPU time. The simulator provides a clear cut comparison between the different strategies on the same input data set.

All the Scheduling Criteria are based on the following methods:

User-oriented Criteria

User-oriented criteria is the behavior of the system as perceived by a single user.

Quantitative:

Turnaround Time – The time from when a process is first added to the group of ready executable processes to the time when it finishes executing and exits.

Response Time – The time from when the request is first made to the time when the response starts to be received. From a user's point of view, response time is a better measure than turnaround time because a process can produce feedback to the user while the process is still executing.

Qualitative:

Predictability – For any given job, it should run in approximately the same amount of time regardless of the load on the system.

System-oriented

System-oriented criteria focus on effective and efficient utilization of the processor.

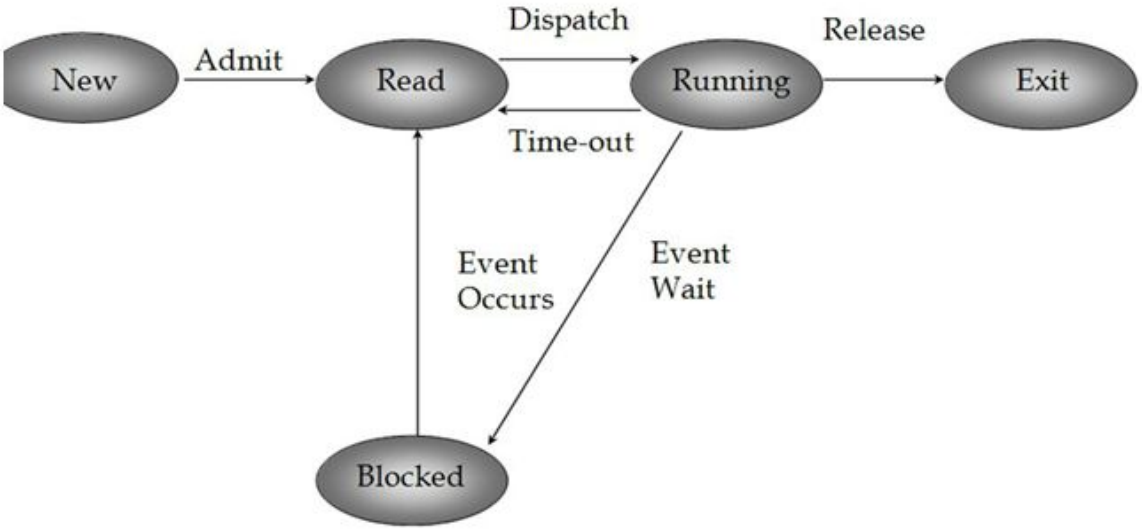
Quantitative:

Throughput – The number of processes completed per unit of time. Depends a lot on the average length of the processes, but is still affected by the scheduling policy.
Processor Utilization – The percentage of time that the processor is busy. More important in multi-user systems than single user systems.

Qualitative:

Fairness – Processes should be treated the same, and no process should suffer starvation.

Five State Process Diagram:



Scheduling Criteria :

While designing this system, we wanted to optimize for some specific system behavior(s).We have compared the performance criteria that are *quantitative* and can be easily measured, and performance criteria that are more *qualitative* and can not be as easily measured.

Round Robin (RR)

Round Robin is a policy that uses preemption based on a clock interrupt at periodic intervals.

Are all types of processes treated fairly?

Yes, all are treated fairly In order to schedule processes fairly, a round-robin scheduler generally employs time-sharing, giving each job a time slot or quantum (its allowance of CPU time), and interrupting the job if it is not completed by then. The job is resumed next time a time slot is assigned to that process. If the process terminates or changes its state to waiting during its attributed time quantum, the scheduler selects the first process in the ready queue to execute. In the absence of time-sharing, or if the quanta were large relative to the sizes of the jobs, a process that produced large jobs would be favored over other processes.

Round Robin algorithm is a pre-emptive algorithm as the scheduler forces the process out of the CPU once the time quota expires.

In the implementation of our Round Robin Algorithm, we have kept the CPU time utilization for any process as 2 seconds, i.e., any process, no matter what its total CPU time is, will remain in the Running State of a maximum of 2 seconds at one go.

First-Come-First-Served (FCFS)

FCFS is a non-preemptive policy that selects and runs a process until completion based on FIFO.

Are all types of processes treated fairly?

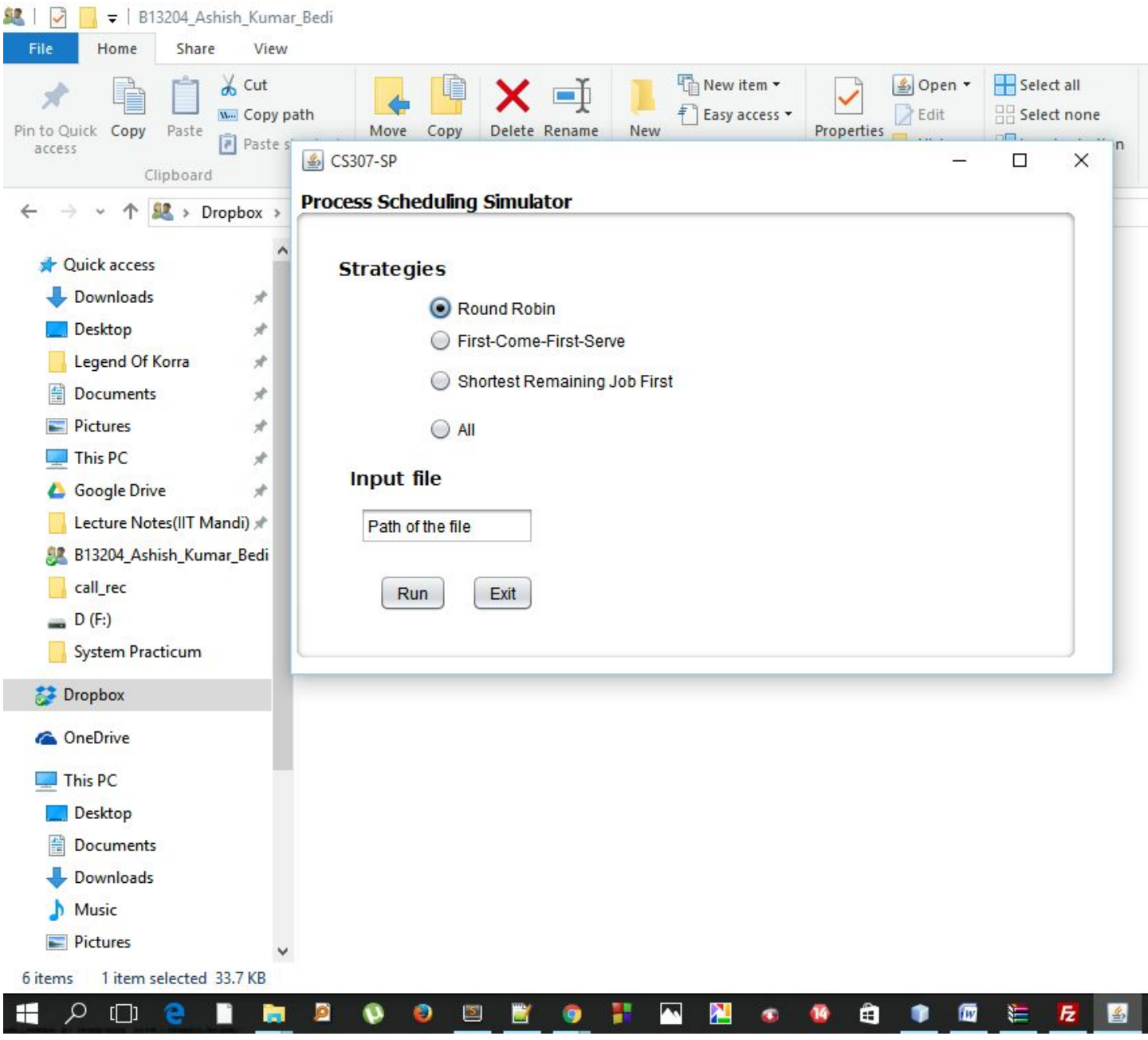
No, they are not. Longer processes are going to fare better than shorter processes, especially in the case when a short process arrives just after a long process.

Processor-bound processes are favored over I/O-bound processes. Suppose we have a collection of processor-bound and I/O-bound processes. When a processor-bound process is running, all I/O-bound processes will have to wait. Some of these I/O-bound processes will be in a blocked state, but could move to the ready state even while the processor-bound process is running once they receive their required input or finish their output operation. When the processor-bound process finally finishes executing, the I/O-bound process will execute only to become quickly blocked by I/O operations again.

Shortest Remaining Time First (SRTF)

Shortest remaining time, also known as **shortest remaining time first (SRTF)**, is a scheduling method that is a preemptive version of shortest job next scheduling. In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a smaller amount of time.

GUI



Are all types of processes treated fairly?

Penalizes long processes. Short processes are able to cut in line ahead of longer processes.Here is a table summarizing everything related to qualitative and quantitative measures of the three algorithms:

	FCFS	**Round Robin**	**SRTF**
Selection Function	max[w]	constant	min[s-e]
Decision Mode	Non-preemptive	Preemptive (at time slice quantum q)	Preemptive(at arrival)
Throughput	Not emphasized	Depends on quantum, could be low if time quantum is small	High
Response Time	May be high, especially if there is a large variance in process execution times	Provides good response time for short processes	Provides good response time to all
Overhead	Minimum	Minimum	Can be high
Effect on Processes	Penalizes short processes; penalizes I/O bound processes	Fair treatment	Penalizes long processes
Starvation	No	No	Possible

So, the above material tells all about the motivation, problem statement, work done and results of the project. Now, coming to the limitations:

Limitations:

The present day Operating Systems have multi-core processor and thus, the process scheduling is much faster and efficient than the one simulated by us.
More strategies can be added.

Learning Outcomes:

We learnt how to use various **technologies** through this project, such as:
i) Eclipse IDE for Java
ii) NetBeans IDE
The entire code was linked and compiled through the Eclipse IDE and the GUI was built in NetBeans. In NetBeans, we used the Swing GUI widget toolkit.

A lot of theoretical concepts were exhausted while building this project. We learnt the three major scheduling algorithms/strategies, namely:
i) Round Robin
ii) First-Come-First-Served
iii) Shortest-Remaining-Time-First
For all these scheduling criteria, we learnt and implement the underlying principles into our code and learnt how to check whether all the processes are treated fairly or not. We also learnt about all the methods a Scheduling Criteria is based upon, which broadly fall into the following two categories:

1) **User Oriented Criteria :**

   Quantitative :

   i) Turnaround Time : The time from when a process is first added to the group of ready executable processes to the time when it finishes executing and exits.
   ii) Response Time : The time from when the request is first made to the time when the response starts to be received. From a user's point of view, response time is a better measure than turnaround time because a process can produce feedback to the user while the process is still executing.

   Qualitative:

   Predictability – For any given job, it should run in approximately the same amount of time, regardless of the load on the system.

2) **System Oriented Criteria :**

Quantitative :

i) Throughput : The number of processes completed per unit of time. Depends a lot on the average length of the processes, but is still affected by the scheduling policy.

ii) Processor Utilization : The percentage of time that the processor is busy. More important in multi-user systems than single user systems.

Qualitative :

Fairness : Processes should be treated the same, and no process should suffer starvation.