CS520-A: Introduction to Operating Systems

The Simulation Programming Project

Process Schedulers

Final Report

By -
Gautam Agrawal
CWID: 20005250

The project specified implementing a Process Scheduler Simulator which simulates various Process Scheduling Algorithms, specifically, First Come First Serve (FCFS), Shortest Job First (SJF) and Round Robin (RR) algorithms.

The above three mentioned have been implemented successfully in the submitted Program files.

The FCFSScheduler.java implements the FCFS scheduling algorithm. It initializes 10 processes with different burst times which are uniformly generated. And then adds the processes to the ready queue at the initialization.

The SJFScheduler.java implements the SJF scheduling algorithm. It also initializes 10 processes with different burst times but the processes are added in the ready queue based on the original burst times generated. The SJF algorithm implemented here is a non-preemptive SJF algorithm.

The RRScheduler.java implements the RR scheduling algorithm. The program selects the quantum value from the file initial_values.java.

The values in the file initial_values.java can be changed safely as long as they make logical sense.

The values initially set for the variables in initial_values.java are:

- No. Of Processes = 10
- Base inter I/O arrival value = 30
- Increment inter I/O arrival value = 5
- I/O Time = 60

- Snapshot Time = 60000
- Quantum Value = 120

The CPU Utilization is between 50 and 90 percent for all process schedulers with the above values.

The average waiting time increases with decreasing quantum time in Round Robin when the time taken to complete an I/O decreases. This is because the Processes finish the I/O fast and then have to wait longer in the ready queue to occupy CPU thus increasing waiting time and the decreasing quantum causes them to occupy the CPU for less time and thus wait in the I/O Queue due to an I/O Interrupt. So they are either waiting in the Ready queue due to decreasing quantum or in I/O queue due to decreasing I/O Time.

Quantum	I/O Time	Average Waiting Time	Log File	Snapshot File	CPU Utilization in %
120	60	1332614.5	log_RR.txt	Snapshot_RR.txt	79.18882169754
110	55	1343306.1	log_RR_2.txt	Snapshot_RR_2.txt	78.41085827060
100	50	1506870.8	log_RR_3.txt	Snapshot_RR_3.txt	80.47447979035
90	45	1539661.9	log_RR_4.txt	Snapshot_RR_4.txt	78.20471931109
80	40	1602520.6	log_RR_5.txt	Snapshot_RR_5.txt	81.42272398149

As can be seen, the CPU utilization for all the processes are within 50-90% and 10 processes have been considered in each run. The Burst time for all the processes are between 2 and 4 minutes as mentioned.

The Gantt charts have been provided in the snapshot files in the form of snapshots with all the required information. The Snapshot files also contain the final state of the processes.

The Outputs and Snapshots for the FCFS algorithm, SJF Algorithm and RR Algorithm are in the main directory with the above mentioned initial values. For FCFS Algorithm, the log file is named “log_FCFS.txt” and the snapshot file is named “Snapshot_FCFS.txt”. For SJF Algorithm, the log file is named “log_SJF.txt” and the snapshot file is named “Snapshot_SJF.txt”. Similarly, for RR Algorithm, the log file is named “log_RR.txt” and the snapshot file is named “Snapshot_RR.txt”.

To run the Process Schedulers, all the files need to be compiled together and the process scheduler selected needs to be run.

If there already exists a log or snapshot file of the scheduler being run, the program will create another file with the name “log_[process scheduler algorithm]_[copy number].txt” if it is a log file or “Snapshot_[process scheduler algorithm]_[copy number].txt” if it is a snapshot file. For eg, “log_RR_2.txt” if there already exists a log file for RR.

The VSCode Workspace for the complete project has been provided for extra help.