COMP 3411 - Assignment 4
Kevin Kim
T00201473
2017/03/29

Analysis of scheduling algorithms

Sample Generation Parameters

Minimum Priority	40
Maximum CPU Burst?	70
Simulation Time?	3600000
Average Arrival Time?	35
Quantim?	50

First Come First Serve without preemption:

Processes are processed in the order in which they arrive. CPU is only released once the process is done. Had the second best performance.

Shortest Job FIrst without preemption:

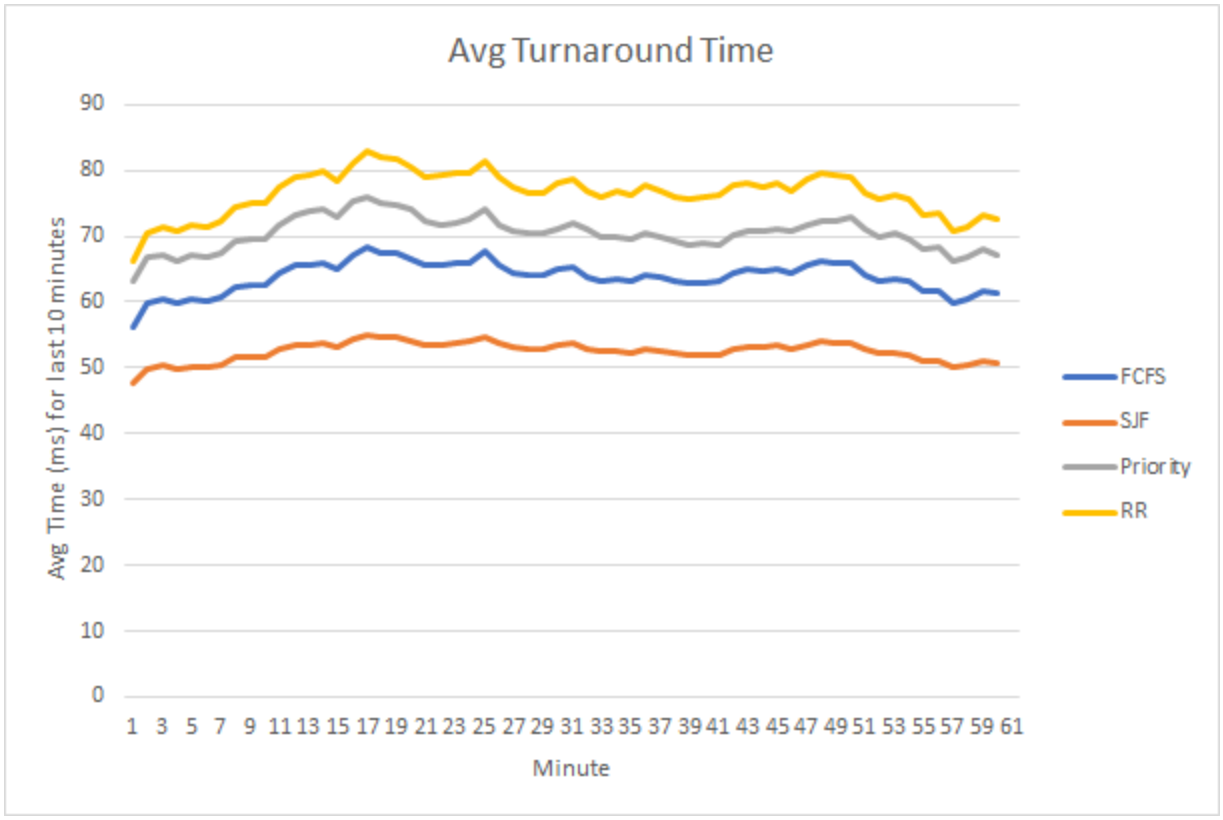
Processes are queued in the order of shortest CPU burst time but once that process occupies the CPU, it is not released until the process is done. Had the best performance.

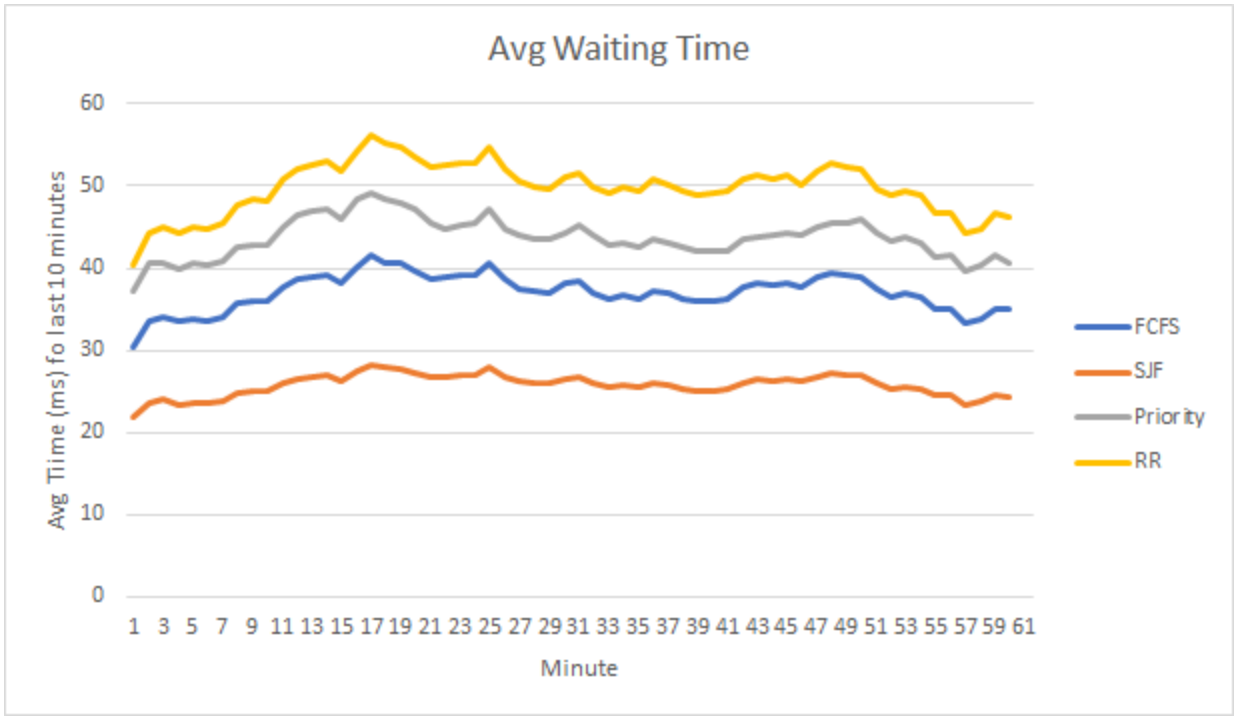
Priority with preemption:

Processes are queued and ordered by priority. If a queued process has higher priority than current process, the current process is preempted. Can prevent lower priority processes from running indefinitely if higher priority processes keep arriving. Didn’t have the best performance since priority is not related to CPU burst time.

Round Robin without preemption:

Processes are queued in the order they arrive but each process either fully processed by the CPU or runs for a given time slice (quantum). This ensures all processes in the queue get equal share of the CPU. Had the worst performance of all 4 scheduling algorithms.


Avg Turnaround Time
Avg Time (ms) for last 10 minutes
90
80
70
60
50
40
30
20
10
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61
Minute
FCFS
SJF
Priority
RR


Avg Waiting Time
Avg Time (ms) fo last 10 minutes
60
50
40
30
20
10
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61
Minute
FCFS
SJF
Priority
RR

Instructions

make sample - generates sample
make fcfs
make sfj
make priority
make rr
make output - Outputs and writes data to OUTPUT_*.txt

Notes:

- Modified ProcessControlBlock to include running time parameter
- Opted to java.util.PriorityQueue rather than given PriorityQueue class because too slow when test data is big.