Project 1

Genevieve Leach

CS 431

Professor Atanasio

11 February 2018

For this project, I implemented simulations for an operating system scheduler for the following algorithms: First Come First Serve, Shortest Job First, Round Robin, and Lottery. I wrote a Process class to hold relevant information to the processes such as PID, Burst Time, and Priority. I wrote a SchedulerAlg class to hold the methods each algorithm needs, as well as write the run() method that FCFS and SJF shares. Meanwhile, RoundRobin and Lottery classes overwrite the run() method as they are different implementations. Each algorithm is in its own class; in the main method I create an object of the algorithm class and call the run() method to perform the simulation.

For FCFS and SJF these were the average completion times:

Test File #	FCFS	SJF
1	1963	1518
2	1913	1370
3	7013	4935
4	8770	5685

At smaller test data with shorter burst times, the difference between FCFS and SJF isn't too bad. However, at larger data with higher burst times, SJF is more efficient than FCFS. This makes sense; FCFS will have to wait for longer processes to finish while SJF gets it done as quick as it can. FCFS's shorter processes will have to wait longer, increasing the total/average completion time. If the jobs are the same length, FCFS and SJF act the same. But with SJF, short jobs do not get stuck under long jobs. If short jobs are constantly arriving though, it may keep longer jobs from running.

For RR25 and RR50 these were the average completion times:

Test File #	RR25	RR50
1	2913	2738
2	2603	2491
3	10323	9827
4	12030	11419

According to my test data, all-around RR50 is the better option for the processes. I assume this is because each process runs for a longer time quantum, especially if there are higher burst times; this cuts down on the time it takes to switch processes, while still maintaining the quicker response time that Round Robin is known for. Additionally, the time quantum should be

chosen carefully, because a quantum that is too long may make the scheduler almost FCFS which is less efficient with longer processes. But it still ensures CPU is shared equally.

For RR50 and Lottery50 these were the average completion times:

Test File #	RR50	Lottery50
1	2738	2563
2	2491	2637
3	9827	8424
4	11419	9803

Lottery is also another approach that lets CPU be shared; though not equally as Round Robin, because processes are picked by the random ticket and their priorities, but on average the allocated CPU time is proportional to the number of tickets given to each job. Lottery can be the same as Round Robin, assuming equal priority. This run displayed in the table is shorter than RR50 all around. However, this may not always be the case, as processes are picked at random in Lottery and test results may change among different runs, as I make it random every ticket drawn.

In conclusion, the scheduler algorithm you want to implement depends on what you want it to do; each one has pros and drawbacks. Round Robin and Lottery are good for CPU sharing and response time, while SJF may process all the short ones while ignoring the longer ones. FCFS is best for equal or short burst times. They all have been used in operating systems or processors at some point, and have their uses depending on the situation. Based on the test data I obtained, SJF was overall the lowest average completion time, but again it depends on what you want to prioritize for performance or response time or other issues.