**Data File 1 – Analysis**

The first come first serve algorithm has the highest waiting and turnaround time of all the algorithms which is due to the fact that all the processes have the same arrival time. Although the algorithm is not known for starving processes, however, it does starve process’ that are at the end of the ready queue as a result of the same arrival time. Process 1 gets a reasonable turnaround time, but remaining processes have an unreasonable turnaround time due to the long service time of process 1. Round Robin algorithm performs reasonably well as the time slice is adequate in completing most of the process in only one time slice. The response time for all processes is greater than that of the first come first serve algorithm. All process’ receive fair treatment and there is no starvation of processes.

The feedback constant algorithm produces the same output as Round Robin algorithm as a result of the processes having the same arrival time. This means that the processes move through the priority queues in order and the algorithm performs exactly like a Round Robin algorithm. Narrow Round Robin performs slight better than the RR and FB algorithm which is due to the algorithm cutting down the execution time allowed for the process after each processing. This results in the process 1 being executed for a CPU time of 3 when it is processed for the second time meaning that process 5 has to wait of one less CPU time period. Which in turn reduces the turnaround and waiting time of process 5 slightly making it smaller than that of the RR and FB algorithm.

**Data File 2 – Analysis**

The FCFS algorithm performs reasonably well for the processes as a result of different arrival times meaning that the turnaround time and waiting time for the processes is a lot smaller unlike the data file 1 processes. Short jobs have an unreasonable turn around and waiting time but this is expected as the algorithm is known for penalizing short processes. RR Algorithm produces a reasonable average turnaround time but the average waiting time for processes is the third largest. The arrival time of process 3 and 4 comes slightly after the completion of the first processing of process 1 at which process 2 is added to the queue and process 1 is loaded back onto the queue. Only at the dispatcher stage (time period 6) does process 3 is added to the ready queue meaning that it must wait for the process 1 and 2 to be processed before it can be process hence the large turnaround and waiting time of process 3 and 4.

FB algorithm produces the best turnaround and waiting time for the data set. The short jobs have the shortest waiting and turnaround time of all the algorithms which is expected as the algorithm processes short process’ very quickly. Process 1 has the largest waiting time and turnaround time which is caused by the long service time of the process. The steady stream of higher priority process’ has caused the process to move down to low-priority queue and waiting indefinitely until the other process’ are processed. This is expected as the algorithm is known its starvation of long jobs if there are many new jobs. NRR algorithm performs very poorly for this data set. The waiting and turnaround time for short process’ is reasonable but this is not the case for long process’. Process 1 has second highest turn around and waiting time as well as Process 5 having the highest turnaround and waiting time. The wasted resources of about 32% (9/28) for context switch explains the high turnaround and waiting time of the algorithm.

Relative performance of Algorithms

The implemented version of the algorithms performs exactly as the algorithm is expected, as it produces the expected results. The algorithm was run with the data set presented in the lecture slides and it produced the expected results. It was also tested with other data sets and produced the expected results. Implementation of the algorithms can definitely be improved through further testing of algorithm with various data sets to find any unexpected results that reveal bugs/errors. The performance of the algorithm depends mainly on data set that is being processed. For example, data set 1 seems to favour the Narrow Round Robin algorithm but if there was 2 long processes then the algorithm would perform poorly as it would keep context switching between the two.