Jeffrey Aguilar

CSCI 300 – M01

Final Project

CPU Scheduling is a process that allows a process to use the CPU while another process being executed is put on hold or in a waiting state. This is due to the unavailability of resources such as I/O. The goal of CPU scheduling is to make the system efficient, fast, and fair. The Round Robin CPU scheduling algorithm assigns a set of processes a fixed amount of time in a cyclic way. This algorithm is one of the most common CPU scheduling techniques used and it helps deal with the issue of starvation since all processes get a fair share of CPU. Round Robin is also preemptive as processes are assigned CPU at a fixed amount of time. Some disadvantages of Round Robin are that there is a larger waiting time and response time, there is low throughput, and there are context switches.

For this project, there are three classes in order to simulate the Round Robin scheduling algorithm. These classes are “Process.java”, “RRScheduler.java”, and “RRScheduling.java”. The “Process.java” class was created in order to read the processes and the other information that comes from the desired CSV file. The three variables that are included in the Process constructor that are related to the CSV file are the “PID”, “ArrivalTime”, and “BurstTime”.

The “RRScheduler.java” class is mainly where the RoundRobin simulation takes place. There is a processes list, ready queue, timer, time quantum, CPU, context switch, and a counter. The timer serves as the clock for the running program, the CPU only holds one process at a time, and the context switch tracks the switches of each process. The RoundRobin() method found in the class contains a while loop that checks if the ready queue, processes list, or CPU isn’t empty. This ensures that each process in the processes list finishes. When the loop is running, a for loop iterates through the processes list and, using an if statement, verifies which processes’ arrival time is equal to the timer. Once a process is successfully verified, it will be loaded onto the ready queue. The CPU is checked to see if it’s empty or not. If the CPU is empty, then the ready queue loads a process onto the CPU. Next, the counter records the current process of the CPU and the process’s service time. Then, an if statement confirms whether the current process’s burst time and service time are equal. If they are confirmed to be equal, then the process is done and can be removed from the CPU.

The program will record the processes’ completion time before it’s removed from the CPU. Then, the process will be added to the end processes list. This helps the program track how many processes have been completed. Next, the CPU becomes equals null which will enable it to accept the subsequent waiting process. The context switch is corrected to match the info of the program and the counter will become 0 which will trigger a reset. The else if statement found in the method determines whether the process is greater than the value of the time quantum. If it’s true, the process returns to the ready queue to be loaded again, the CPU will be updated back to null, and the context switch will also be corrected.

This process is the CPU’s way of effectively being occupied and Round Robin ensures this to be the case. The program will then compute and display CPU utilization, throughput, average waiting time, and average turnaround time. The program can perform these calculations with the following equations:

Average Turnaround Time = The total of all Turnaround times / Processes

Average waiting time = The total of all waiting times / Processes

Throughput = Processes / Timer

CPU utilization = (Burst time – (Context time \* 0.01)) / Processes

Finally, the method will print out the CPU utilization, throughput, average waiting time, and average turnaround time.

When you run the program “RRScheduling.java”, you’ll be asked to input the path to the CSV file you wish the program to read. Next, you’ll be asked to enter the time quantum you want the program to run at. Then, the program will read the list of processes stored in the CSV file. Then, the program will store the contents of the file in an ArrayList. Afterwards, the RoundRobin() method in “RRScheduler.java” will run. Finally, the program will run the calculations and when it’s finished, the console will display the calculations for the user to read after all processes have been completed. Examples are shown below.

Table

Description automatically generated

The CSV Process List

Text

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated

Text, letter

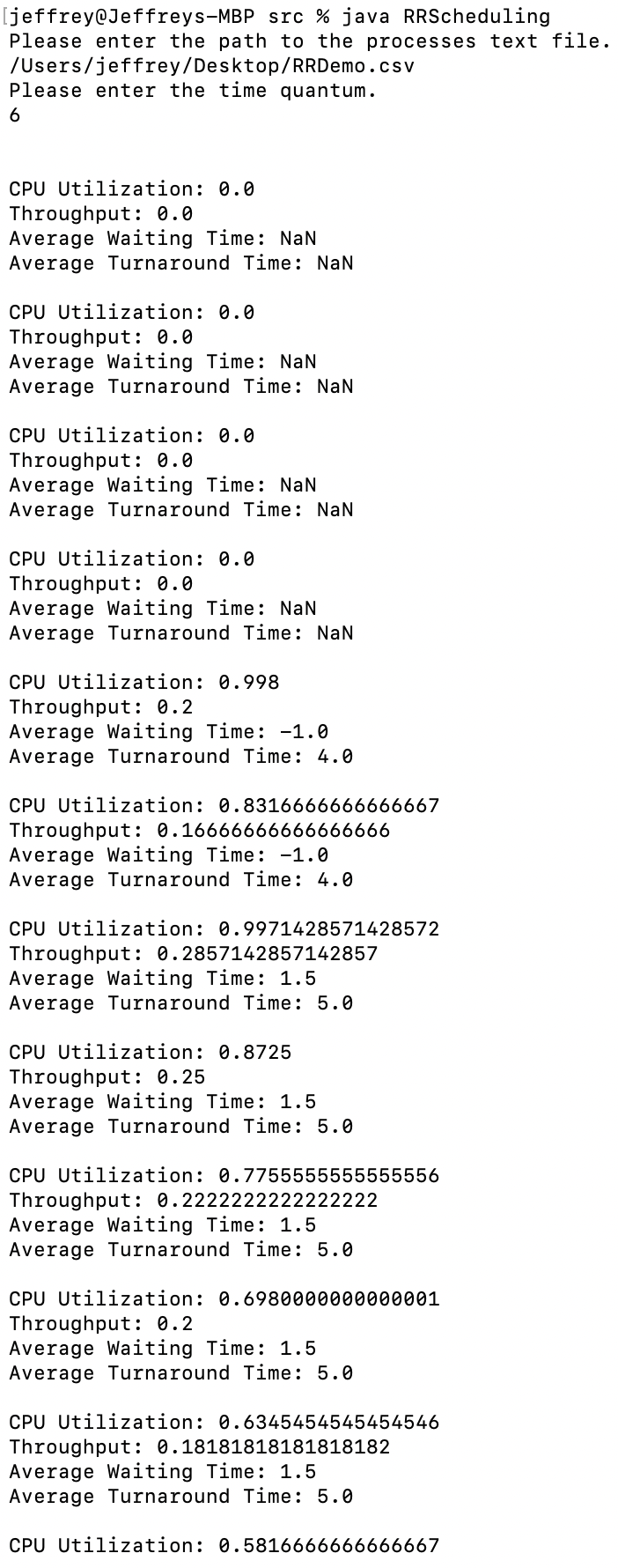
Description automatically generated

Text, letter

Description automatically generated

A picture containing text, newspaper, receipt

Description automatically generated



Text, letter

Description automatically generated

Sources:

<https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/>

<https://www.studytonight.com/operating-system/cpu-scheduling>