Thomas Blandino
Jahaan Jain

CSCI 330-M03
Dr. Susan Gass

Operating Systems Project:
CPU Scheduling Implementing the Round Robin (RR) Algorithm

Introduction:

CPU Scheduling is a process of determining which process will use CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU is idle, the operating system selects one of the processes available in the ready queue. This allows for maximum CPU utilization, a phrase that was said in class many times. This allows the operating system to make full use of the CPI, thus making scheduling as fast as possible as well as efficient. There are two different types of CPU Scheduling, preemptive and nonpreemptive. In preemptive scheduling, the tasks are mostly assigned with their priorities, whereas in non preemptive scheduling the CPU is allocated to a specific process. Moreover, there are mainly six types of process scheduling algorithms, but the one focused on in this project is Round Robin Scheduling (RR). Round Robin is the most efficient preemptive process scheduling algorithm out of them all. Each process is provided a fixed time to execute, which can be called time quantum. Once a process is executed for a given time period, it is preempted, and another process executes for a given time period. Moreover, context switching is something used to save states of preempted processes. Therefore, when changing the time quantum value, the response time of the processes can be increased and could be too large, so one should pick a value that is not too large and not too small.

Implementation of Round Robin Algorithm:

For the implementation of this Round Robin algorithm, the following were steps taken for execution of the program:

1).

In our Simulator.java class, a simulator object was created that has the parameters of time quantum and context switch time. The initial time and context switches were set to zero. The string filePath as well as the csv file name (process.csv), were parameters also included in the similar object.

2).

In the simulator object, each parameter was initialized as some were stated above, and the process.csv file was parsed based on the three parameters in the csv file (process id, arrival time, and burst time).

3).

Then, it was all added to the ready queue if the burst time was not equal to zero. In each row of the process.csv file, a process object is created and stored into the process list (ArrayList<Process> processes).

4). Lastly, each process is executed based off of the time quantum value imputed and the context switch value. It switches with the next process when it has finished its turn.

Java Implementation:

The RR CPU scheduling algorithm was implemented in the following 3 Java classes:

1). Simulator.java:

- This class was created to define the Round Robin scheduling algorithm.

  The following parameters were used in this Process.java class:

  a). *time* - The total time value for all the finished processes.

  b). *contextSwitchTime* - Value for how long it will take a content switch for completion.

  c). *contextSwitches* - The total number of context switches.

  d). *timeQuantum* - How long a process can spend in execution.

  e). *numberOfProcesses* - The total number of processes that will be processed for completion.

  f). *totalBurstTime* - The total burst time of all the processes.

  g). *gantt* - This is for showing the sequence of the processes that are being executed.

  h). *ArrayList<Process>processes* - List of processes.

  i). *Queue<Process>ready* - A FIFO ready queue for the process to determine which process should be executed on the CPU next.

The most essential methods used in our Simulator.java class is of the following:

a). *public void addToReadyQueue( )* - This was used when the processes burst time was not equal to zero. If the "if" statement was true, it would add process to the ready queue. Without this method, the process id could not be added to the ready queue, thus not making our program run.

b). *public void contextSwitch( )* - Increases the process context switch value by one and adds it to the process queue.

c). *public void processQueue( )* - Used to show the arrival times before and after added into the ready queue and sets the completed and waiting time. This is essential in our program because this is how the processes and the times are added based on the various time quantum values.

2). Process.java:

- This class was created to define process objects such as the id, waiting time, arrival time and to make setters and getters for each parameter to be used in the Simulator and Main class.

  The following parameters were used in this Process.java class:

  a). *pid* - ID of the process loaded from the csv file.

  b). *burst* - The processes burst time loaded from the csv file.

  c). *initialBurst* - The process' initial burst time.

  d). *arrival* - The arrival time of the process while arriving in the ready queue.

  e). *waitingTime* - The number for how long the process has been kept waiting.

  f). *completedTime* - The total time for the process to finish its execution.

  g). *completed* - Used to check if the process is completed or not.

  h). *ready* - Checks to see if the process is in the ready queue.

**Every parameter is important in this class because it displays the setters and getters for each parameter. Without this class, nothing would run because in the Simulator and Main class, the get methods are called. Without these get methods, nothing would run.

3). Main.java:

- This class contains the main method and runs our simulation.

How to Run the Program for Demonstration of the RR Scheduling:

In order to get an output for the program, simply run these commands in console:

```
java c Main.java
java Main processes.csv <time_quantum>
```

If for some reason this does not work, please edit the following line in Main.java

```
Simulator rr = new Simulator(args[0], Integer.parseInt(args[1]), 1);
```

To this:

```
Simulator rr = new Simulator("<path_to_csv>", <time_quantum>, 1);
```

**Therefore, by completing this line of code, the information pulled from the "processes.csv" file will be displayed in the console in its respective categories of calculation.

Process Table (.csv file where the information is being held):

The following table is a process table that consists of the following three elements: Process ID, Arrival Time, and Burst Time.

**This is where our information was pulled out to display all of the parameters in our output.

Process ID	***Arrival Time***	***Burst Time***
1	0	4
2	6	17
3	9	9
4	0	3
5	1	2
6	13	15

7	7	7

Screenshots of Output with Different Time Quantum (5 Total Test):

Test #1: Time Quantum = 3

```
+-+-+-+-+-+-+-+-+-+ Round Robin Simulator +-+-+-+-+-+-+-+-+-+

    Processes: 7 | Time Quantum: 3 | Context Switch Time: 1

+-+-+-+-+-+ Authors: Jahaan Jain & Thomas Blandino +-+-+-+-+-+

CPU Utilization (Expected / Actual): 0.7403
Throughput (Processes / Total Time): 0.0909
Average Turnaround Time (Avg Completion Time / Processes): 35.7143
Average Waiting Time ((Turnaround / Initial) / Processes): 27.5714
Gantt Chart:  -> P1 -> P2 -> P3 -> P4 -> P5 -> P6 -> P7 -> P1 -> P2 -> P3 -> P6 -> P7 -> P2 -
> P3 -> P6 -> P7 -> P2 -> P6 -> P2 -> P6 -> P2
Total Context Switches: 20
Total Completion Time: 77

ID: 01 | Turnaround Time: 28 | Waiting Time: 24 | Completion Time: 28
ID: 02 | Turnaround Time: 65 | Waiting Time: 48 | Completion Time: 71
ID: 03 | Turnaround Time: 34 | Waiting Time: 25 | Completion Time: 43
ID: 04 | Turnaround Time: 15 | Waiting Time: 12 | Completion Time: 15
ID: 05 | Turnaround Time: 16 | Waiting Time: 14 | Completion Time: 17
ID: 06 | Turnaround Time: 48 | Waiting Time: 33 | Completion Time: 61
ID: 07 | Turnaround Time: 44 | Waiting Time: 37 | Completion Time: 51
```

Test #2: Time Quantum = 6

```
+-+-+-+-+-+-+-+-+-+ Round Robin Simulator +-+-+-+-+-+-+-+-+-+

    Processes: 7 | Time Quantum: 6 | Context Switch Time: 1

+-+-+-+-+-+ Authors: Jahaan Jain & Thomas Blandino +-+-+-+-+-+

CPU Utilization (Expected / Actual): 0.8261
Throughput (Processes / Total Time): 0.1014
Average Turnaround Time (Avg Completion Time / Processes): 31.7143
Average Waiting Time ((Turnaround / Initial) / Processes): 23.5714
Gantt Chart:  -> P1 -> P2 -> P3 -> P4 -> P5 -> P6 -> P7 -> P2 -> P3 -> P6 -> P7 -> P2 -> P6
Total Context Switches: 12
Total Completion Time: 69

ID: 01 | Turnaround Time: 04 | Waiting Time: 00 | Completion Time: 04
ID: 02 | Turnaround Time: 53 | Waiting Time: 36 | Completion Time: 59
ID: 03 | Turnaround Time: 32 | Waiting Time: 23 | Completion Time: 41
ID: 04 | Turnaround Time: 22 | Waiting Time: 19 | Completion Time: 22
ID: 05 | Turnaround Time: 23 | Waiting Time: 21 | Completion Time: 24
ID: 06 | Turnaround Time: 43 | Waiting Time: 28 | Completion Time: 56
ID: 07 | Turnaround Time: 45 | Waiting Time: 38 | Completion Time: 52
```

Test #3: Time Quantum = 1

```
+-+-+-+-+-+-+-+-+-+ Round Robin Simulator +-+-+-+-+-+-+-+-+-+

    Processes: 7 | Time Quantum: 1 | Context Switch Time: 1

+-+-+-+-+-+ Authors: Jahaan Jain & Thomas Blandino +-+-+-+-+-+

CPU Utilization (Expected / Actual): 0.5044
Throughput (Processes / Total Time): 0.0619
Average Turnaround Time (Avg Completion Time / Processes): 57.8571
Average Waiting Time ((Turnaround / Initial) / Processes): 49.7143
Gantt Chart:  -> P1 -> P2 -> P3 -> P4 -> P5 -> P6 -> P7 -> P1 -> P2 -> P3 -> P4 -> P5 -> P6 -
> P7 -> P1 -> P2 -> P3 -> P4 -> P6 -> P7 -> P1 -> P2 -> P3 -> P6 -> P7 -> P2 -> P3 -> P6 -> P
7 -> P2 -> P3 -> P6 -> P7 -> P2 -> P3 -> P6 -> P7 -> P2 -> P3 -> P6 -> P2 -> P3 -> P6 -> P2 -
> P6 -> P2 -> P6 -> P2 -> P6 -> P2 -> P6 -> P2 -> P6 -> P2 -> P6 -> P2 -> P2
Total Context Switches: 56
Total Completion Time: 113

ID: 01 | Turnaround Time: 41 | Waiting Time: 37 | Completion Time: 41
ID: 02 | Turnaround Time: 101 | Waiting Time: 84 | Completion Time: 107
ID: 03 | Turnaround Time: 65 | Waiting Time: 56 | Completion Time: 74
ID: 04 | Turnaround Time: 35 | Waiting Time: 32 | Completion Time: 35
ID: 05 | Turnaround Time: 21 | Waiting Time: 19 | Completion Time: 22
ID: 06 | Turnaround Time: 83 | Waiting Time: 68 | Completion Time: 96
ID: 07 | Turnaround Time: 59 | Waiting Time: 52 | Completion Time: 66
```

Test #4: Time Quantum = 2

```
+-+-+-+-+-+-+-+-+-+ Round Robin Simulator +-+-+-+-+-+-+-+-+-+

    Processes: 7 | Time Quantum: 2 | Context Switch Time: 1

+-+-+-+-+-+ Authors: Jahaan Jain & Thomas Blandino +-+-+-+-+-+

CPU Utilization (Expected / Actual): 0.6552
Throughput (Processes / Total Time): 0.0805
Average Turnaround Time (Avg Completion Time / Processes): 41.8571
Average Waiting Time ((Turnaround / Initial) / Processes): 33.7143
Gantt Chart:  -> P1 -> P2 -> P3 -> P4 -> P5 -> P6 -> P7 -> P1 -> P2 -> P3 -> P4 -> P6 -> P7 -
> P2 -> P3 -> P6 -> P7 -> P2 -> P3 -> P6 -> P7 -> P2 -> P3 -> P6 -> P2 -> P6 -> P2 -> P6 -> P
2 -> P6 -> P2
Total Context Switches: 30
Total Completion Time: 87

ID: 01 | Turnaround Time: 23 | Waiting Time: 19 | Completion Time: 23
ID: 02 | Turnaround Time: 75 | Waiting Time: 58 | Completion Time: 81
ID: 03 | Turnaround Time: 47 | Waiting Time: 38 | Completion Time: 56
ID: 04 | Turnaround Time: 31 | Waiting Time: 28 | Completion Time: 31
ID: 05 | Turnaround Time: 12 | Waiting Time: 10 | Completion Time: 13
ID: 06 | Turnaround Time: 59 | Waiting Time: 44 | Completion Time: 72
ID: 07 | Turnaround Time: 46 | Waiting Time: 39 | Completion Time: 53
```

Test #5: Time Quantum = 8

```
+-+-+-+-+-+-+-+-+ Round Robin Simulator +-+-+-+-+-+-+-+-+

    Processes: 7 | Time Quantum: 8 | Context Switch Time: 1

+-+-+-+-+-+ Authors: Jahaan Jain & Thomas Blandino +-+-+-+-+-+

CPU Utilization (Expected / Actual): 0.8507
Throughput (Processes / Total Time): 0.1045
Average Turnaround Time (Avg Completion Time / Processes): 31.7143
Average Waiting Time ((Turnaround / Initial) / Processes): 23.5714
Gantt Chart:  -> P1 -> P2 -> P3 -> P4 -> P5 -> P6 -> P7 -> P2 -> P3 -> P6 -> P2
Total Context Switches: 10
Total Completion Time: 67

ID: 01 | Turnaround Time: 04 | Waiting Time: 00 | Completion Time: 04
ID: 02 | Turnaround Time: 55 | Waiting Time: 38 | Completion Time: 61
ID: 03 | Turnaround Time: 39 | Waiting Time: 30 | Completion Time: 48
ID: 04 | Turnaround Time: 26 | Waiting Time: 23 | Completion Time: 26
ID: 05 | Turnaround Time: 27 | Waiting Time: 25 | Completion Time: 28
ID: 06 | Turnaround Time: 39 | Waiting Time: 24 | Completion Time: 52
ID: 07 | Turnaround Time: 32 | Waiting Time: 25 | Completion Time: 39
```

Analysis/Conclusion:

As shown in the console output, when the time quantum value was changed five different times, all the parameters, such as the CPU utilization, average turnaround time, and so on were also changed. Therefore, it can be determined that in the below categories, the time quantum was performed the best.

CPU Utilization:

Best performed when time quantum was equal to 1.

Rank #2: Time Quantum = 2

Rank #3: Time Quantum = 3

Rank #4: Time Quantum = 6

Rank #5: Time Quantum = 8

Throughout:

Best performed when time quantum was equal to 1.

Rank #2: Time Quantum = 2

Rank #3: Time Quantum = 3

Rank #4: Time Quantum = 6

Rank #5: Time Quantum = 8

Average Turnaround Time:

Best performed when time quantum was 8 and 6 (Same Value).

Rank #3: Time Quantum = 3

Rank #4: Time Quantum = 2

Rank #5: Time Quantum = 1

Average Waiting Time:

Best performed when time quantum was 8 and 6 (Same Value).

Rank #3: Time Quantum = 3

Rank #4: Time Quantum = 2

Rank #5: Time Quantum = 1

Total Context Switches:

Best performed when time quantum was equal to 8.

Rank #2: Time Quantum = 6

Rank #3: Time Quantum = 3

Rank #4: Time Quantum = 2

Rank #5: Time Quantum = 1

Total Completion Time:

Best performed when time quantum was equal to 8.

Rank #2: Time Quantum = 6

Rank #3: Time Quantum = 3

Rank #4: Time Quantum = 2

Rank #5: Time Quantum = 1

What can be concluded from these results is that the higher the time quantum, the quicker the average turnaround time and the average waiting time. However, when the time quantum was equal to 8, it did have the highest CPU utilization number as well as the throughput time.