Round Robin CPU Scheduling Project Report

Aminuz Zaman

1. *Introduction to CPU Scheduling*

CPU scheduling is a fundamental concept in operating systems that manages how processes are assigned to the CPU for execution. Effective scheduling ensures that CPU resources are optimized, balancing factors like CPU utilization, throughput, waiting time, turnaround time, and response time.

2. *Understanding the Round Robin (RR) Scheduling Algorithm*

Round Robin (RR) is a preemptive scheduling algorithm widely used in time-sharing systems. Each process is assigned a fixed time slice, called a **time quantum**. When a process's time quantum expires, it is placed back in the ready queue if it hasn't completed, allowing other processes to access the CPU. This cycle continues until all processes complete. Round Robin's time quantum size significantly impacts performance; a small quantum improves response time but increases context switching, while a large quantum reduces overhead but may lead to poorer response time.

3. *Project Implementation*

This project simulates Round Robin scheduling in Python, using a CSV file for input. Each process in the file has a **pid** (Process ID), **arrival time** (when the process enters the queue), and **burst time** (total CPU time required). The program accepts two parameters:

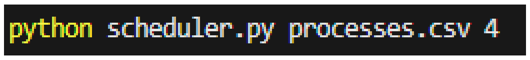
- **File Path**: Path to the CSV file with the processes.
- **Time Quantum**: The time slice assigned to each process per cycle.

The program includes several components:

- **Clock**: Timestamps events.
- **Process Creator**: Reads processes from the file and adds them to the queue based on arrival times.
- **CPU**: Executes each process for the given time quantum.
- **Queue**: Maintains the FIFO order of processes.
- **Performance Metrics**: After running, the program calculates CPU utilization, throughput, average waiting time, and average turnaround time.

4. *Instructions for Running the Program*

To run the program, use the following command in the terminal:



```
python scheduler.py processes.csv 4
```

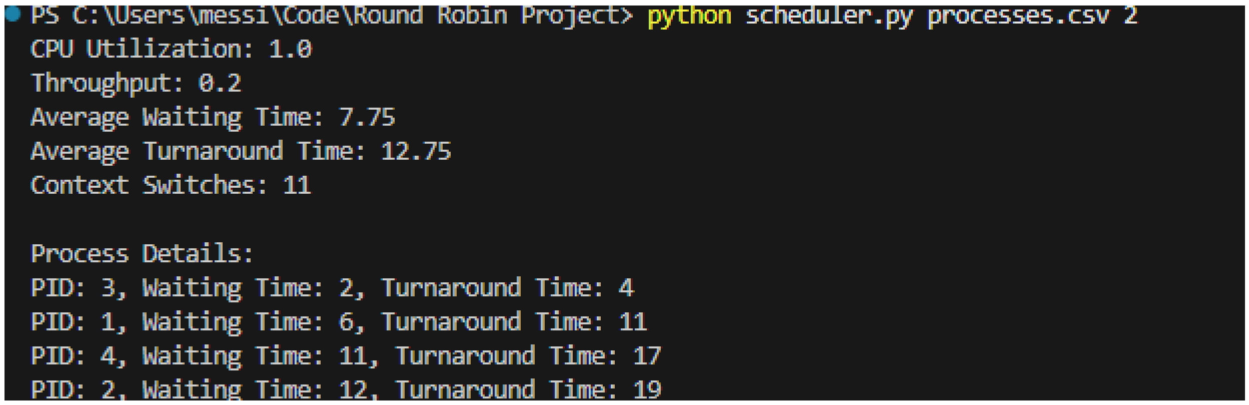
In this case, processes.csv refers to your process list file with the desired time quantum at the end (e.g., 2, 4, etc).

5. *Program Output*

The program outputs performance metrics for the scheduling, including:

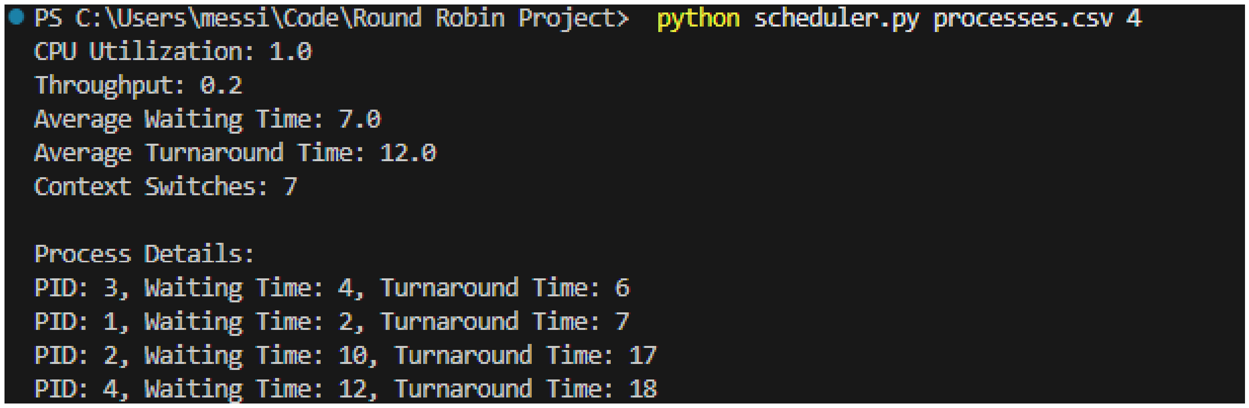
- **CPU Utilization**: The percentage of time the CPU is active.
- **Throughput**: Number of processes completed per unit time.
- **Average Waiting Time**: The average time processes spend waiting in the queue.
- **Average Turnaround Time**: The average time from arrival to completion for each process.

6. *Analysis of Results with Different Time Quantum Values*



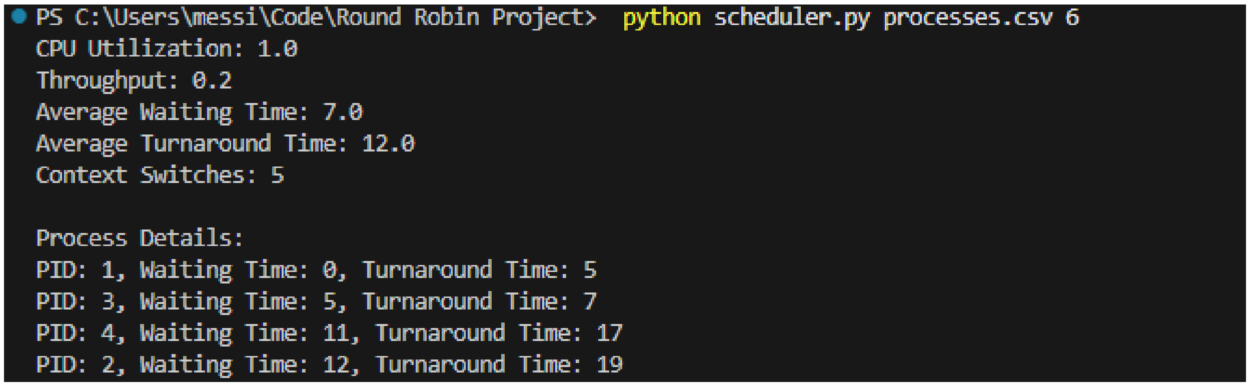
```
PS C:\Users\messi\Code\Round Robin Project> python scheduler.py processes.csv 2
CPU Utilization: 1.0
Throughput: 0.2
Average Waiting Time: 7.75
Average Turnaround Time: 12.75
Context Switches: 11

Process Details:
PID: 3, Waiting Time: 2, Turnaround Time: 4
PID: 1, Waiting Time: 6, Turnaround Time: 11
PID: 4, Waiting Time: 11, Turnaround Time: 17
PID: 2, Waiting Time: 12, Turnaround Time: 19
```



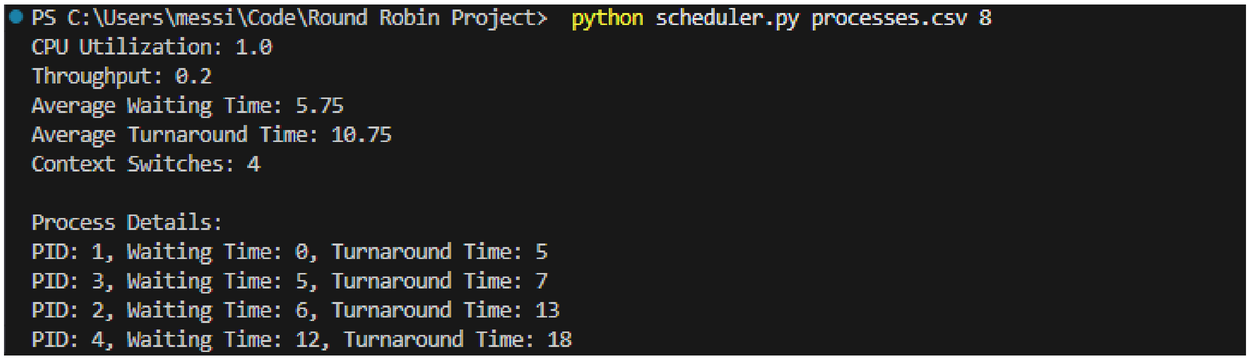
```
PS C:\Users\messi\Code\Round Robin Project>  python scheduler.py processes.csv 4
CPU Utilization: 1.0
Throughput: 0.2
Average Waiting Time: 7.0
Average Turnaround Time: 12.0
Context Switches: 7

Process Details:
PID: 3, Waiting Time: 4, Turnaround Time: 6
PID: 1, Waiting Time: 2, Turnaround Time: 7
PID: 2, Waiting Time: 10, Turnaround Time: 17
PID: 4, Waiting Time: 12, Turnaround Time: 18
```



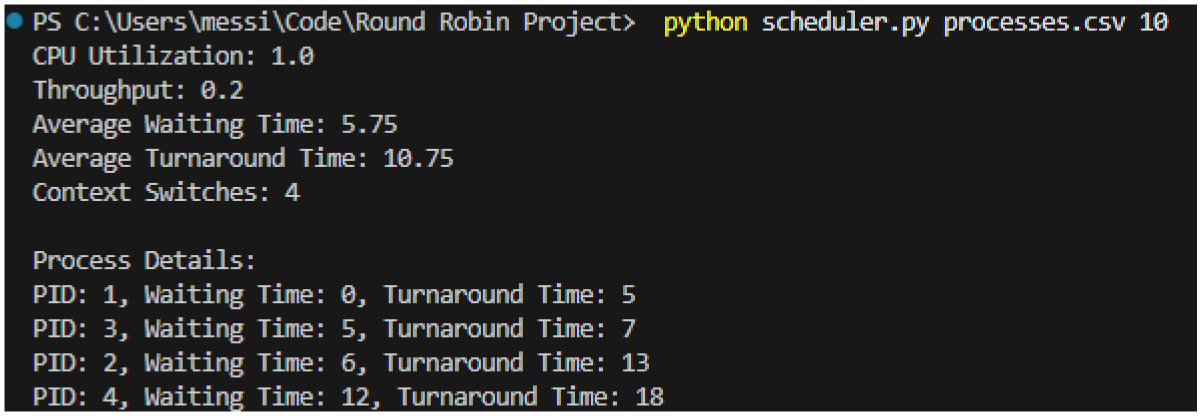
```
PS C:\Users\messi\Code\Round Robin Project>  python scheduler.py processes.csv 6
CPU Utilization: 1.0
Throughput: 0.2
Average Waiting Time: 7.0
Average Turnaround Time: 12.0
Context Switches: 5

Process Details:
PID: 1, Waiting Time: 0, Turnaround Time: 5
PID: 3, Waiting Time: 5, Turnaround Time: 7
PID: 4, Waiting Time: 11, Turnaround Time: 17
PID: 2, Waiting Time: 12, Turnaround Time: 19
```



```
PS C:\Users\messi\Code\Round Robin Project>  python scheduler.py processes.csv 8
CPU Utilization: 1.0
Throughput: 0.2
Average Waiting Time: 5.75
Average Turnaround Time: 10.75
Context Switches: 4

Process Details:
PID: 1, Waiting Time: 0, Turnaround Time: 5
PID: 3, Waiting Time: 5, Turnaround Time: 7
PID: 2, Waiting Time: 6, Turnaround Time: 13
PID: 4, Waiting Time: 12, Turnaround Time: 18
```



```
PS C:\Users\messi\Code\Round Robin Project>  python scheduler.py processes.csv 10
CPU Utilization: 1.0
Throughput: 0.2
Average Waiting Time: 5.75
Average Turnaround Time: 10.75
Context Switches: 4

Process Details:
PID: 1, Waiting Time: 0, Turnaround Time: 5
PID: 3, Waiting Time: 5, Turnaround Time: 7
PID: 2, Waiting Time: 6, Turnaround Time: 13
PID: 4, Waiting Time: 12, Turnaround Time: 18
```

From the analysis, it's clear that the choice of time quantum significantly affects performance metrics in Round Robin scheduling. Here are some observations and recommendations based on the results:

- **Small Time Quantum**: A smaller time quantum (e.g., 2) leads to more frequent context switches, which makes the system highly responsive. However, the increased context-switching overhead can decrease CPU efficiency. This setting may be beneficial for interactive systems where responsiveness is critical, such as in real-time applications.
- **Moderate Time Quantum**: A moderate time quantum (e.g., 4–6) strikes a balance between response time and CPU utilization. With this setting, processes wait a reasonable amount of time, and context-switching overhead is reduced compared to a smaller quantum. This is often a good choice for general-purpose systems.
- **Large Time Quantum**: A large time quantum (e.g., 8 or 10) reduces context switches further, which can be beneficial for CPU-bound processes that require longer, uninterrupted execution. However, this setting may cause delays for shorter or interactive processes, as they must wait longer for CPU access.