UNIVERSITI MALAYA

WIA2004 Operating Systems

Lab 1: CPU Scheduling Algorithms

Group：F5U

Occ: 10

Lecturer: Dr. Fazidah Othman

GROUP MEMBERS	MATRICS NUMBER
ZHANG ZHIYANG	S2193685
MEILIN LI	S2174975
HUMYRA TASMIA	S2176677
HUSNA NADIAH	22060027
HUSNA BINTI IHSANUDDIN	U2102305

Table of Content

1.0 Introduction

CPU scheduling algorithms are essential for managing process execution in computer systems. Among these algorithms is the First-Come, First-Served (FCFS) scheduling algorithm, which is one of the simplest non-preemptive scheduling algorithms.

FCFS operates on the principle of serving processes in the order they arrive in the ready queue. There is no consideration for the burst time or priority of processes; the CPU serves the process that has been waiting the longest. While FCFS is easy to implement and understand, it may lead to longer waiting times and turnaround times for processes, especially when long-running processes are scheduled before short ones.

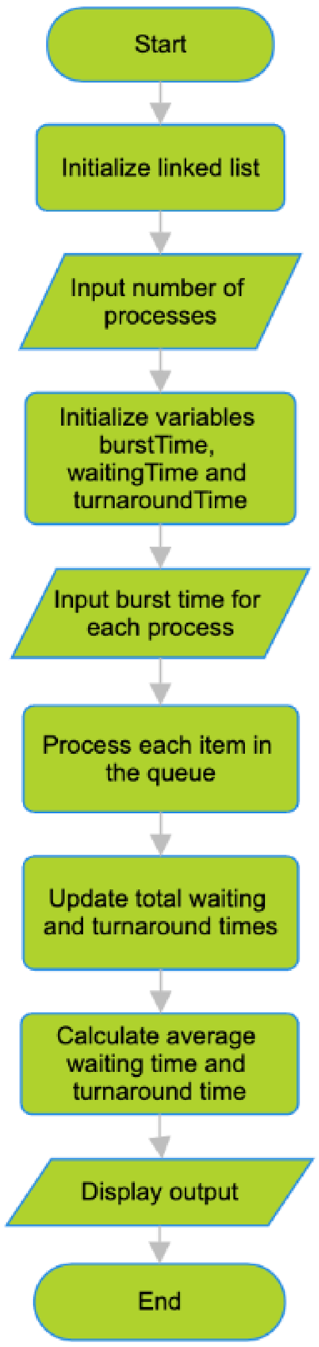
Here we are assuming that the arrival time for all processes is the same, but not necessarily 0. Waiting time for each process is the total time it spends waiting in the queue before getting executed. For the first process, waiting time = 0. For subsequent processes, waiting time = sum of burst times of preceding processes. Turnaround time for each process is the total time it takes from being submitted to the system until it completes execution. Turnaround time = Waiting Time + Burst Time.

If the arrival times for all processes are the same, then the turnaround time and completion time will be the same for all the processes. The waiting time will depend on the burst time of the process. The shorter the burst time, the shorter the waiting time. Therefore, if all the processes have the same arrival time, the order in which they are executed will depend only on their burst times.

In this program, we will simulate the FCFS CPU scheduling algorithm by taking inputs such as the number of processes and their CPU burst times. We will calculate the waiting time and turnaround time for each process based on their arrival order, providing insights into the performance of the FCFS algorithm.

2.0 Methodology

Flowchart

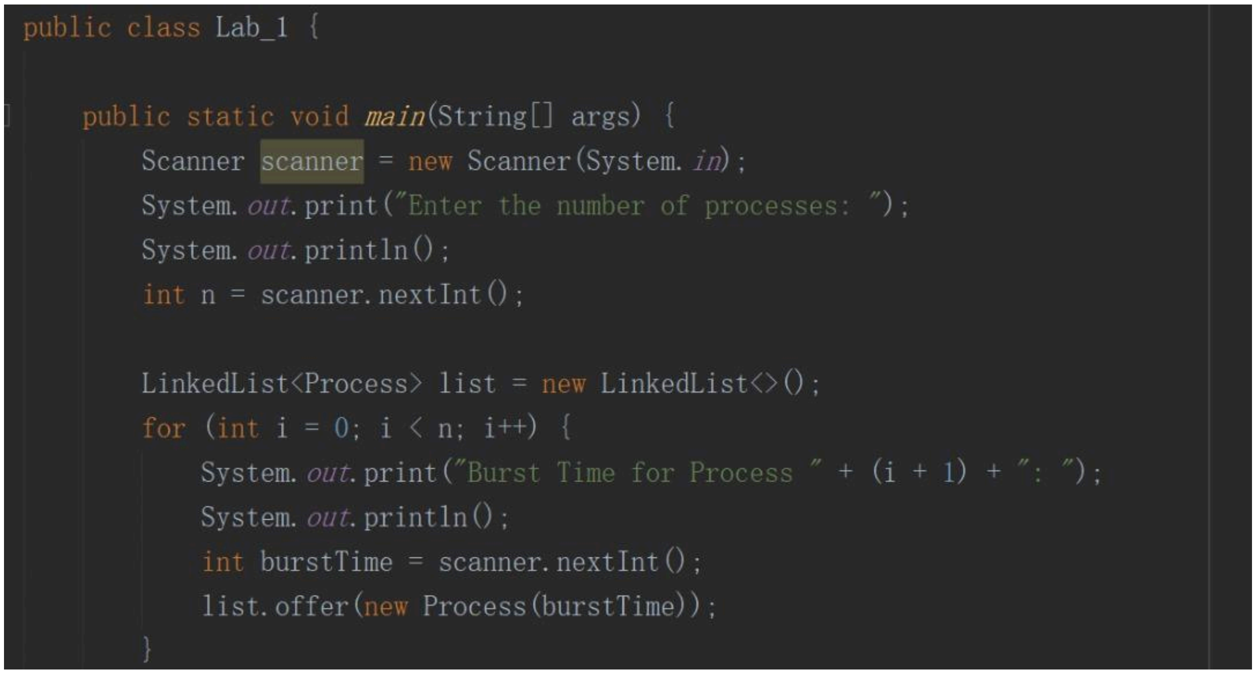

Start
Initialize linked list
Input number of processes
Initialize variables burstTime, waitingTime and turnaroundTime
Input burst time for each process
Process each item in the queue
Update total waiting and turnaround times
Calculate average waiting time and turnaround time
Display output
End

Pseudocode

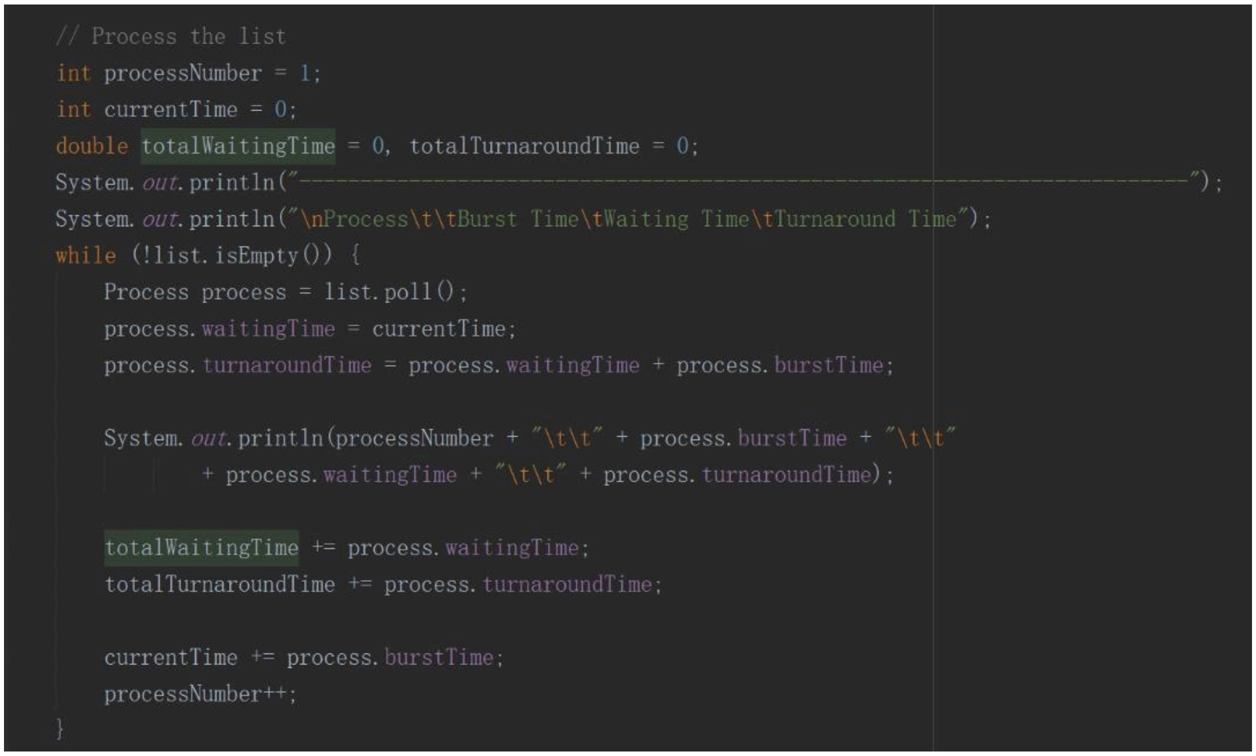
```
Start

Initialize a LinkedList of Process

Prompt the user for the number of processes

For i = 1 to number of processes
    Prompt the user for the burst time of process i
    Create a new Process with the given burst time
    Add the new Process to the LinkedList
End For

Set currentTime to 0
Set totalWaitingTime to 0
Set totalTurnaroundTime to 0
Set processNumber to 1

While the LinkedList is not empty
    Take the first Process from the LinkedList (poll)
    Set its waitingTime to currentTime
    Set its turnaroundTime to its waitingTime plus its burstTime

    Display the processNumber, its burstTime, waitingTime, and turnaroundTime

    Add the Process's waitingTime to totalWaitingTime
    Add the Process's turnaroundTime to totalTurnaroundTime
    Increment currentTime by the Process's burstTime
    Increment processNumber
End While

Calculate avgWaitingTime as totalWaitingTime divided by number of processes
Calculate avgTurnaroundTime as totalTurnaroundTime divided by number of
processes

Display the avgWaitingTime
Display the avgTurnaroundTime

End
```

3.0 Implementation

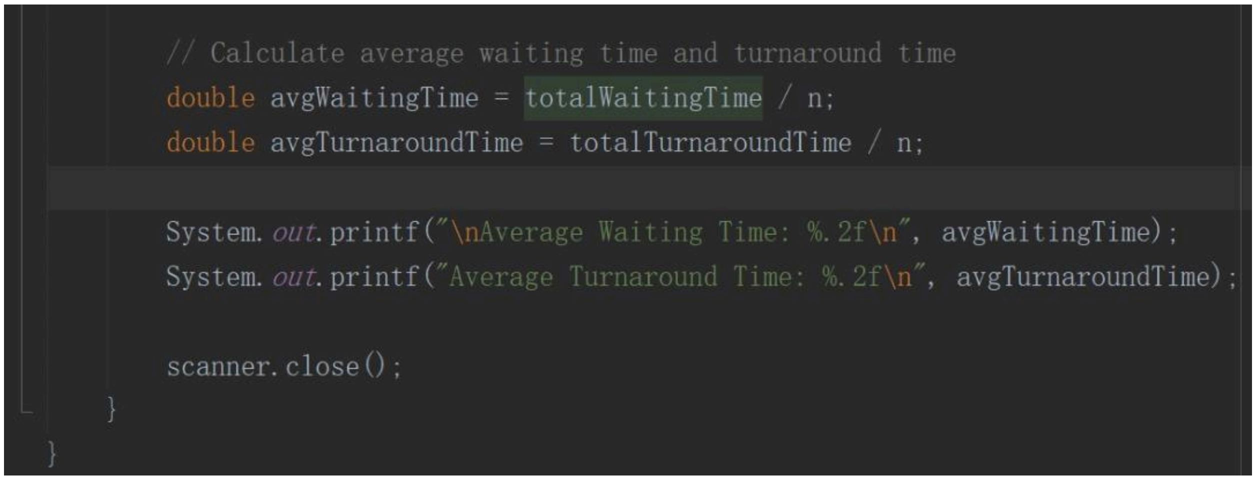
3.1 Coding



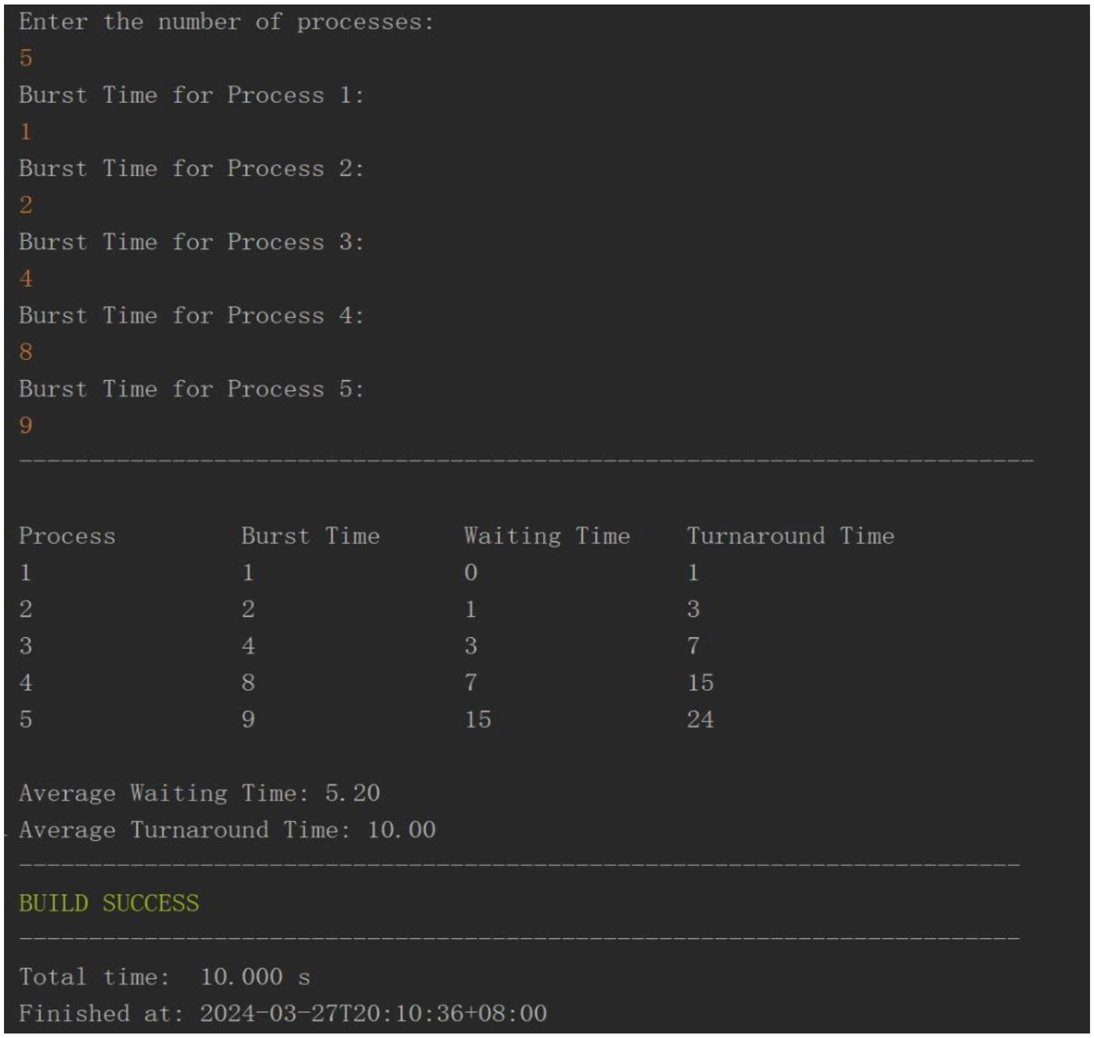
```java
public class Lab_1 {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of processes: ");
        System.out.println();
        int n = scanner.nextInt();

        LinkedList<Process> list = new LinkedList<>();
        for (int i = 0; i < n; i++) {
            System.out.print("Burst Time for Process " + (i + 1) + ": ");
            System.out.println();
            int burstTime = scanner.nextInt();
            list.offer(new Process(burstTime));
        }
```



```java
// Process the list
int processNumber = 1;
int currentTime = 0;
double totalWaitingTime = 0, totalTurnaroundTime = 0;
System.out.println("-------------------------------------------------------------------------");
System.out.println("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
while (!list.isEmpty()) {
    Process process = list.poll();
    process.waitingTime = currentTime;
    process.turnaroundTime = process.waitingTime + process.burstTime;

    System.out.println(processNumber + "\t\t" + process.burstTime + "\t\t"
            + process.waitingTime + "\t\t" + process.turnaroundTime);

    totalWaitingTime += process.waitingTime;
    totalTurnaroundTime += process.turnaroundTime;

    currentTime += process.burstTime;
    processNumber++;
}
```


```java
        // Calculate average waiting time and turnaround time
        double avgWaitingTime = totalWaitingTime / n;
        double avgTurnaroundTime = totalTurnaroundTime / n;

        System.out.printf("\nAverage Waiting Time: %.2f\n", avgWaitingTime);
        System.out.printf("Average Turnaround Time: %.2f\n", avgTurnaroundTime);

        scanner.close();
    }
}
```

3.2 Sample output


```
Enter the number of processes:
5
Burst Time for Process 1:
1
Burst Time for Process 2:
2
Burst Time for Process 3:
4
Burst Time for Process 4:
8
Burst Time for Process 5:
9
------------------------------------------------------------------------


Process         Burst Time      Waiting Time    Turnaround Time
1               1               0               1
2               2               1               3
3               4               3               7
4               8               7               15
5               9               15              24

Average Waiting Time: 5.20
Average Turnaround Time: 10.00
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  10.000 s
Finished at: 2024-03-27T20:10:36+08:00
```

4.0 Conclusion

The First Come First Serve (FCFS) scheduling algorithm is straightforward and guarantees equality by executing the processes in the order they arrive. However, its drawbacks include the potential of long waiting times, especially for processes with long execution times. Additionally, FCFS may not be suitable for systems with varying process burst times, as shorter processes can get delayed behind longer ones. While FCFS is easy to understand and implement, its lack of prioritization based on process characteristics may lead to inefficient resource utilization and unpredictable response times, making it less suitable for time-critical systems. Therefore, while FCFS offers simplicity, its limitations necessitate the consideration of alternative scheduling algorithms for better system performance and responsiveness.

5.0 References

- G. S. (2018, May 19). *L-2.3: First Come First Serve(FCFS) CPU Scheduling Algorithm with Example*. YouTube. https://www.youtube.com/watch?v=MZdVAVMgNpA
- G. (2023, July 21). *Program for FCFS CPU Scheduling Set 1*. GeeksforGeeks. https://www.geeksforgeeks.org/program-for-fcfs-cpu-scheduling-set-1/