Assignment - 3

OPERATING SYSTEM

TOPIC: Process Scheduling, - PART1

---

1. Write a C programme to simulate the following **non-preemptive** CPU scheduling algorithms to find the turnaround time and waiting time for the above problem.
    a. FCFS
    b. SJF
    c. Priority

★ FCFS CPU SCHEDULING ALGORITHM

- For the FCFS scheduling algorithm, read the number of processes/jobs in the system, and their CPU burst times.
- The scheduling is performed based on the arrival time of the processes, irrespective of their other parameters.
- Each process will be executed according to its arrival time.
- Calculate the waiting time and turnaround time of each of the processes accordingly.

CODE:-

```
#include <stdio.h>
struct Process {
    int processID;
    int burstTime;
    int arrivalTime;
    int waitingTime;
    int turnaroundTime;
};
void findWaitingTime(struct Process processes[], int n) {
    processes[0].waitingTime = 0; // First process has no waiting time

    for (int i = 1; i < n; i++) {
        processes[i].waitingTime = processes[i - 1].waitingTime + processes[i - 1].burstTime;
    }
}
void findTurnaroundTime(struct Process processes[], int n) {
    for (int i = 0; i < n; i++) {
        processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;
    }
}
void findAverageTimes(struct Process processes[], int n) {
    int totalWaitingTime = 0, totalTurnaroundTime = 0;
    findWaitingTime(processes, n);
    findTurnaroundTime(processes, n);
    printf("Processes\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        totalWaitingTime += processes[i].waitingTime;
        totalTurnaroundTime += processes[i].turnaroundTime;
        printf("%d\t\t%d\t\t%d\t\t%d\n", processes[i].processID, processes[i].burstTime,
processes[i].waitingTime, processes[i].turnaroundTime);
```

```
    }
    printf("Average waiting time = %.2f\n", (float)totalWaitingTime / n);
    printf("Average turnaround time = %.2f\n", (float)totalTurnaroundTime / n);
}
int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    struct Process processes[n];
    for (int i = 0; i < n; i++) {
        processes[i].processID = i + 1;
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &processes[i].burstTime);
        processes[i].arrivalTime = i;
}
    findAverageTimes(processes, n);
    return 0;
}
```

OUTPUT :-

```
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment3$ gcc fcfs.c -o fcfs
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment3$ ./fcfs
Enter the number of processes: 8
Enter burst time for process 1: 5
Enter burst time for process 2: 9
Enter burst time for process 3: 1
Enter burst time for process 4: 7
Enter burst time for process 5: 3
Enter burst time for process 6: 15
Enter burst time for process 7: 3
Enter burst time for process 8: 8
Processes       Burst Time      Waiting Time    Turnaround Time
1               5               0               5
2               9               5               14
3               1               14              15
4               7               15              22
5               3               22              25
6               15              25              40
7               3               40              43
8               8               43              51
Average waiting time = 20.50
Average turnaround time = 26.88
```

★ SJF CPU SCHEDULING ALGORITHM

- For the SJF scheduling algorithm, read the number of processes/jobs in the system, and their CPU burst times.
- Arrange all the jobs in order with respect to their burst times.
- Two jobs may be in queue with the same execution time, and then the FCFS approach will be performed.
- Each process will be executed according to the length of its burst time.
- Then calculate each process's waiting time and turnaround time accordingly.

CODE:-

```
#include <stdio.h>

struct Process {
    int processID;
    int burstTime;
    int waitingTime;
    int turnaroundTime;
```

```
};
void sortProcessesByBurstTime(struct Process processes[], int n) {
    struct Process temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (processes[i].burstTime > processes[j].burstTime) {
                temp = processes[i];
                processes[i] = processes[j];
                processes[j] = temp;
            }
        }
    }
}

void findWaitingTime(struct Process processes[], int n) {
    processes[0].waitingTime = 0; // First process has no waiting time

    for (int i = 1; i < n; i++) {
        processes[i].waitingTime = processes[i - 1].waitingTime + processes[i - 1].burstTime;
    }
}

void findTurnaroundTime(struct Process processes[], int n) {
    for (int i = 0; i < n; i++) {
        processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;
    }
}

void findAverageTimes(struct Process processes[], int n) {
    int totalWaitingTime = 0, totalTurnaroundTime = 0;

    findWaitingTime(processes, n);
    findTurnaroundTime(processes, n);

    printf("Processes\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        totalWaitingTime += processes[i].waitingTime;
        totalTurnaroundTime += processes[i].turnaroundTime;
        printf("%d\t\t%d\t\t%d\t\t%d\n", processes[i].processID, processes[i].burstTime,
processes[i].waitingTime, processes[i].turnaroundTime);
    }

    printf("Average waiting time = %.2f\n", (float)totalWaitingTime / n);
    printf("Average turnaround time = %.2f\n", (float)totalTurnaroundTime / n);
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
```

```
    struct Process processes[n];
    for (int i = 0; i < n; i++) {
        processes[i].processID = i + 1;
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &processes[i].burstTime);
    }

    sortProcessesByBurstTime(processes, n);
    findAverageTimes(processes, n);

    return 0;
}
```

OUTPUT :-

```
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment3$ gcc sjf.c -o sjf
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment3$ ./sjf
Enter the number of processes: 6
Enter burst time for process 1: 5
Enter burst time for process 2: 1
Enter burst time for process 3: 9
Enter burst time for process 4: 8
Enter burst time for process 5: 4
Enter burst time for process 6: 8
Processes       Burst Time      Waiting Time    Turnaround Time
2               1               0               1
5               4               1               5
1               5               5               10
4               8               10              18
6               8               18              26
3               9               26              35
Average waiting time = 10.00
Average turnaround time = 15.83
```

★ PRIORITY CPU SCHEDULING ALGORITHM

- For the priority scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the priorities.
- Arrange all the jobs in order with respect to their priorities.
- There may be two jobs in queue with the same priority, and then FCFS approach will be performed.
- Each process will be executed according to its priority.
- Calculate the waiting time and turnaround time of each of the processes accordingly.

CODE:-

```
#include <stdio.h>
struct Process {
    int processID;
    int burstTime;
    int priority;
    int waitingTime;
    int turnaroundTime;
};
void sortProcessesByPriority(struct Process processes[], int n) {
    struct Process temp;
```

```
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (processes[i].priority > processes[j].priority) {
                temp = processes[i];
                processes[i] = processes[j];
                processes[j] = temp;
            }
        }
    }
}
void findWaitingTime(struct Process processes[], int n) {
    processes[0].waitingTime = 0; // First process has no waiting time
    for (int i = 1; i < n; i++) {
        processes[i].waitingTime = processes[i - 1].waitingTime + processes[i - 1].burstTime;
    }
}
void findTurnaroundTime(struct Process processes[], int n) {
    for (int i = 0; i < n; i++) {
        processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;
    }
}
void findAverageTimes(struct Process processes[], int n) {
    int totalWaitingTime = 0, totalTurnaroundTime = 0;

    findWaitingTime(processes, n);
    findTurnaroundTime(processes, n);
    printf("Processes\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        totalWaitingTime += processes[i].waitingTime;
        totalTurnaroundTime += processes[i].turnaroundTime;
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].processID, processes[i].burstTime,
processes[i].priority, processes[i].waitingTime, processes[i].turnaroundTime);
    }
    printf("Average waiting time = %.2f\n", (float)totalWaitingTime / n);
    printf("Average turnaround time = %.2f\n", (float)totalTurnaroundTime / n);
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    struct Process processes[n];
    for (int i = 0; i < n; i++) {
```

```
        processes[i].processID = i + 1;
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &processes[i].burstTime);
        printf("Enter priority for process %d: ", i + 1);
        scanf("%d", &processes[i].priority);
    }
    sortProcessesByPriority(processes, n);
    findAverageTimes(processes, n);

    return 0;
}
```

OUTPUT :-

```
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment3$ gcc priorityScheduling.c -o priorityScheduling
namrata@NamraRio:~/MCA2023/Namrata_B_34/assignment3$ ./priorityScheduling
Enter the number of processes: 6
Enter burst time for process 1: 5
Enter priority for process 1: 2
Enter burst time for process 2: 3
Enter priority for process 2: 4
Enter burst time for process 3: 1
Enter priority for process 3: 4
Enter burst time for process 4: 2
Enter priority for process 4: 8
Enter burst time for process 5: 9
Enter priority for process 5: 3
Enter burst time for process 6: 7
Enter priority for process 6: 10
Processes       Burst Time      Priority        Waiting Time    Turnaround Time
1               5               2               0               5
5               9               3               5               14
3               1               4               14              15
2               3               4               15              18
4               2               8               18              20
6               7               10              20              27
Average waiting time = 12.00
Average turnaround time = 16.50
```