4ITRC2 Operating System Lab

Lab Assignment 5

Aim: To create C programs for the different scheduling algorithms.

To perform: Create and execute C programs for following CPU Scheduling Algorithms:

1. First Come First Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin Scheduling

To Submit: C Codes for the above scheduling algorithms with their outputs.

1. First Come First Serve (FCFS) Scheduling

```
#include <stdio.h>
void main() {
    int n, i, wt[10], bt[10], tat[10];
    float avg_wt = 0, avg_tat = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter burst times:\n");
    for (i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &bt[i]);
    }

    wt[0] = 0;
    tat[0] = bt[0];
    avg_tat += tat[0];

    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = wt[i] + bt[i];
```

```
        avg_tat += tat[i];
        avg_wt += wt[i];
    }

    avg_wt /= n;
    avg_tat /= n;

    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        printf("P%d\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }

    printf("Average waiting time: %.2f\n", avg_wt);
    printf("Average turnaround time: %.2f\n", avg_tat);
}
```

Sample Output:

Enter the number of processes: 3

Enter burst times:

Process 1: 24

Process 2: 3

Process 3: 3

Process	Burst Time	Waiting Time	Turnaround Time
P1	24	0	24
P2	3	24	27
P3	3	27	30

2. Shortest Job First (SJF) Scheduling

```
#include <stdio.h>
void main() {
    int n, i, j, temp, bt[10], wt[10], tat[10];
    float avg_wt = 0, avg_tat = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter burst times:\n");
    for (i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &bt[i]);
    }
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (bt[j] > bt[j + 1]) {
                temp = bt[j];
                bt[j] = bt[j + 1];
                bt[j + 1] = temp;
            }
        }
    }

    wt[0] = 0;
    tat[0] = bt[0];
    avg_tat += tat[0];

    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = wt[i] + bt[i];
```

```
        avg_tat += tat[i];
        avg_wt += wt[i];
    }

    avg_wt /= n;
    avg_tat /= n;

    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        printf("P%d\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }

    printf("Average waiting time: %.2f\n", avg_wt);
    printf("Average turnaround time: %.2f\n", avg_tat);
}
```

Sample Output:

Enter the number of processes: 3

Enter burst times:

Process 1: 6

Process 2: 8

Process 3: 7

Process	Burst Time	Waiting Time	Turnaround Time
P1	6	0	6
P3	7	6	13
P2	8	13	21

Average waiting time: 6.33

Average turnaround time: 13.33

3. Round Robin Scheduling

```
#include <stdio.h>
void main() {
    int n, i, time_quantum, bt[10], wt[10] = {0}, tat[10] = {0};
    int remaining_bt[10], time = 0, remain;

    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter burst times:\n");
    for (i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &bt[i]);
        remaining_bt[i] = bt[i];
    }

    printf("Enter time quantum: ");
    scanf("%d", &time_quantum);
    remain = n;

    while (remain > 0) {
        for (i = 0; i < n; i++) {
            if (remaining_bt[i] > 0) {
                if (remaining_bt[i] > time_quantum) {
                    time += time_quantum;
                    remaining_bt[i] -= time_quantum;
                } else {
                    time += remaining_bt[i];
                    remaining_bt[i] = 0;
                    tat[i] = time;
                    wt[i] = tat[i] - bt[i];
```

```
                remain--;
            }
        }
    }
}

    float avg_wt = 0, avg_tat = 0;
    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        printf("P%d\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);
        avg_wt += wt[i];
        avg_tat += tat[i];
    }
    avg_wt /= n;
    avg_tat /= n;

    printf("Average waiting time: %.2f\n", avg_wt);
    printf("Average turnaround time: %.2f\n", avg_tat);
}
```

Sample Output:

Enter the number of processes: 3

Enter burst times:

Process 1: 10

Process 2: 5

Process 3: 8

Enter time quantum: 2

Process Burst Time Waiting Time Turnaround Time

P1 10 13 23

P2	5	4	9
P3	8	10	18

Average waiting time: 9.00

Average turnaround time: 16.67