

OPERATING SYSTEM (4ITRC2) 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:

To Submit:

1.First come First serve (FCFS).

2.Shortest job First (SJF).

3.Round Robin Scheduling.

```
#include <stdio.h>
```

// 1. First Come First Serve (FCFS) Scheduling Algorithm

```
void fcfs(int processes[], int n, int burst_time[]) {
    int wait_time[n], turnaround_time[n];
    wait_time[0] = 0;
```

```
for (int i = 1; i < n; i++)
    wait_time[i] = wait_time[i - 1] + burst_time[i - 1];
```

```
for (int i = 0; i < n; i++)
    turnaround time[i] = wait time[i] + burst time[i];
```

```
printf("\nFCFS Scheduling:\nProcess \t Burst Time \t Waiting Time \t  
Turnaround Time\n");
```

```
for (int i = 0; i < n; i++)
    printf("P%d \t %d \t\t %d \t\t %d\n", processes[i], burst_time[i], wait_time[i],
        turnaround_time[i]);
}
```

// 2. Shortest Job First (SJF) Scheduling Algorithm

```
void sjf(int processes[], int n, int burst_time[]) {
    int temp, i, j, wait_time[n], turnaround_time[n], pos;

    for (i = 0; i < n; i++) {
        pos = i;
        for (j = i + 1; j < n; j++)
            if (burst_time[j] < burst_time[pos])
                pos = j;
        temp = burst_time[i];
        burst_time[i] = burst_time[pos];
        burst_time[pos] = temp;
        temp = processes[i];
        processes[i] = processes[pos];
        processes[pos] = temp;
    }

    wait_time[0] = 0;
    for (i = 1; i < n; i++)
        wait_time[i] = wait_time[i - 1] + burst_time[i - 1];

    for (i = 0; i < n; i++)
        turnaround_time[i] = wait_time[i] + burst_time[i];

    printf("\nSJF Scheduling:\nProcess \t Burst Time \t Waiting Time \t Turnaround\nTime\n");
    for (i = 0; i < n; i++)
        printf("P%d \t %d \t %d \t %d\n", processes[i], burst_time[i], wait_time[i], turnaround_time[i]);
}
```

// 3. Round Robin Scheduling Algorithm

```
void round_robin(int processes[], int n, int burst_time[], int quantum) {
    int remaining_time[n], wait_time[n], turnaround_time[n], t = 0;

    for (int i = 0; i < n; i++)
        remaining_time[i] = burst_time[i];

    int done;
```

```

do {
    done = 1;
    for (int i = 0; i < n; i++) {
        if (remaining_time[i] > 0) {
            done = 0;
            if (remaining_time[i] > quantum) {
                t += quantum;
                remaining_time[i] -= quantum;
            } else {
                t += remaining_time[i];
                wait_time[i] = t - burst_time[i];
                remaining_time[i] = 0;
            }
        }
    }
} while (!done);

for (int i = 0; i < n; i++)
    turnaround_time[i] = burst_time[i] + wait_time[i];

printf("\nRound Robin Scheduling:\nProcess \t Burst Time \t Waiting Time \t\n\t Turnaround Time\n");
for (int i = 0; i < n; i++)
    printf("P%d \t %d \t %d \t %d\n", processes[i], burst_time[i], wait_time[i], turnaround_time[i]);
}

int main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    int processes[n], burst_time[n];

    printf("Enter burst times:\n");
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("P%d: ", i + 1);
        scanf("%d", &burst_time[i]);
    }
}

```

```
fcfs(processes, n, burst_time);  
sjf(processes, n, burst_time);  
  
int quantum;  
printf("Enter time quantum for Round Robin: ");  
scanf("%d", &quantum);  
round_robin(processes, n, burst_time, quantum);  
  
return 0;  
}
```

Milind Sharma2414193