

Operating System

(4ITRC2)

IT IV Semester

Submitted by

HIMANSHU GAWALI

23I4025

Information Technology -A

Submitted to

Jasneet kaur

Department of Information Technology

Institute of Engineering and
Technology

Devi Ahilya Vishwavidhyalaya, Indore (M.P.) India

(www.iet.dauniv.ac.in)

Session jan- may, 2025

Lab Assignment 5

Aim: To create C programs for different scheduling algorithms.

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

1. First Come First Serve (FCFS).

FCFS is a non-preemptive scheduling algorithm where the process that arrives first gets executed first.

C Program for FCFS:

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

```
void findWaitingTime(int processes[], int n, int bt[], int wt[]) {  
    wt[0] = 0;  
    for (int i = 1; i < n; i++) {  
        wt[i] = bt[i - 1] + wt[i - 1];  
    }  
}
```

```
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])  
{  
    for (int i = 0; i < n; i++) {  
        tat[i] = bt[i] + wt[i];  
    }  
}
```

```
void findAvgTime(int processes[], int n, int bt[]) {  
    int wt[n], tat[n];  
    findWaitingTime(processes, n, bt, wt);  
    findTurnAroundTime(processes, n, bt, wt, tat);  
}
```

```

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

```

```

int main() {
    int processes[] = {1, 2, 3};
    int n = sizeof processes / sizeof processes[0];
    int burst_time[] = {10, 5, 8};
    findAvgTime(processes, n, burst_time);
    return 0;
}

```

Expected Output:

Processes	Burst Time	Waiting Time	Turnaround Time
1	10	0	10
2	5	10	15
3	8	15	23

2. Shortest Job First (SJF) Scheduling

SJF selects the process with the shortest burst time first, reducing the average waiting time.

C Program for SJF:

```
#include <stdio.h>

void findWaitingTime(int n, int bt[], int wt[]) {
    wt[0] = 0;
    for (int i = 1; i < n; i++) {
        wt[i] = bt[i - 1] + wt[i - 1];
    }
}

void findTurnAroundTime(int n, int bt[], int wt[], int tat[]) {
    for (int i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
    }
}

void findAvgTime(int n, int bt[]) {
    int wt[n], tat[n], total_wt = 0, total_tat = 0;
    findWaitingTime(n, bt, wt);
    findTurnAroundTime(n, bt, wt, tat);

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

int main() {
    int bt[] = {6, 8, 7, 3};
    int n = sizeof bt / sizeof bt[0];
```

```

for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - i - 1; j++) {
        if (bt[j] > bt[j + 1]) {
            int temp = bt[j];
            bt[j] = bt[j + 1];
            bt[j + 1] = temp;
        }
    }
}

findAvgTime(n, bt);
return 0;
}

```

Expected Output:

Processes	Burst Time	Waiting Time	Turnaround Time
4	3	0	3
1	6	3	9
3	7	9	16
2	8	16	24

HIMANSHU GAWALI-40/2025

3. Round Robin Scheduling

Round Robin (RR) scheduling assigns a fixed time quantum and cycles through all processes.

C Program for Round Robin:

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

```
void roundRobin(int processes[], int n, int bt[], int quantum) {  
    int rem_bt[n], t = 0;
```

```
    for (int i = 0; i < n; i++) {  
        rem_bt[i] = bt[i];  
    }
```

```
    while (1) {  
        int done = 1;  
        for (int i = 0; i < n; i++) {  
            if (rem_bt[i] > 0) {  
                done = 0;  
                if (rem_bt[i] > quantum) {  
                    t += quantum;  
                    rem_bt[i] -= quantum;  
                } else {  
                    t += rem_bt[i];  
                    rem_bt[i] = 0;  
                }  
                printf("Process %d executed till time %d\n", i + 1, t);  
            }  
        }  
        if (done == 1) break;  
    }  
}
```

```
int main() {  
    int processes[] = {1, 2, 3, 4};  
    int n = sizeof processes / sizeof processes[0];  
    int burst_time[] = {8, 4, 9, 5};  
    int quantum = 3;  
  
    roundRobin(processes, n, burst_time, quantum);  
    return 0;  
}
```

Expected Output:

Process 1 executed till time 3
Process 2 executed till time 6
Process 3 executed till time 9
Process 4 executed till time 12
Process 1 executed till time 15
Process 3 executed till time 18
Process 4 executed till time 19
Process 1 executed till time 20
Process 3 executed till time 21