Operating System Lab

(4ITRC2)

IT IV Semester

Submitted by

Bhumika Badoniya

23I4124

Information Technology - B

Submitted to

MRS.JASNEET KAUR

Department of Information Technology

Institute of Engineering and Technology

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

(www.iet.dauniv.ac.in)

Session JAN-APRIL, 2025

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 the 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)

```
#include <stdio.h>

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

    int bt[n], wt[n], tat[n];
    printf("Enter burst time for each process:\n");
    for (i = 0; i < n; i++) {
```

```
        printf("P%d: ", i + 1);
        scanf("%d", &bt[i]);
    }



    wt[0] = 0; // First process has 0 waiting time
    for (i = 1; i < n; i++)
        wt[i] = wt[i - 1] + bt[i - 1];

    printf("\nProcess\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        tat[i] = wt[i] + bt[i];
        printf("P%d\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }

    return 0;
}
```

Sample Output:

```
Enter number of processes: 3
Enter burst time for each process:
P1: 5
P2: 3
P3: 8

Process BT  WT  TAT
```

P1 5 0 5

P2 3 5 8

P3 8 8 16

2. Shortest Job First (SJF - Non-preemptive)

```
#include <stdio.h>



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

   int bt[n], p[n], wt[n], tat[n], temp;

   printf("Enter burst time for each process:\n");
   for (i = 0; i < n; i++) {
      printf("P%d: ", i + 1);
      scanf("%d", &bt[i]);
      p[i] = i + 1;
   }

   // Sort processes by burst time
   for (i = 0; i < n - 1; i++) {
      for (j = i + 1; j < n; j++) {
```

```
            if (bt[i] > bt[j]) {
                temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
                temp = p[i]; p[i] = p[j]; p[j] = temp;
            }
        }
    }

    wt[0] = 0;
    for (i = 1; i < n; i++)
        wt[i] = wt[i - 1] + bt[i - 1];



    printf("\nProcess\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        tat[i] = wt[i] + bt[i];
        printf("P%d\t%d\t%d\t%d\n", p[i], bt[i], wt[i], tat[i]);
    }

    return 0;
}
```

Sample Output:

Enter number of processes: 3

Enter burst time for each process:

P1: 6

P2: 8

P3: 7

Process BT WT TAT

P1 6 0 6

P3 7 6 13

P2 8 13 21

3. Round Robin Scheduling

```
#include <stdio.h>

int main() {
   int n, i, time_quantum, total = 0;
   printf("Enter number of processes: ");



   scanf("%d", &n);

   int bt[n], rt[n], wt[n], tat[n];
   printf("Enter burst time for each process:\n");
   for (i = 0; i < n; i++) {
     printf("P%d: ", i + 1);
      scanf("%d", &bt[i]);
      rt[i] = bt[i];
   }

   printf("Enter time quantum: ");
```

```
scanf("%d", &time_quantum);


int t = 0, done;
do {
   done = 1;
   for (i = 0; i < n; i++) {
      if (rt[i] > 0) {
         done = 0;
         if (rt[i] > time_quantum) {
            t += time_quantum;
            rt[i] -= time_quantum;
         } else {
            t += rt[i];
            wt[i] = t - bt[i];
            rt[i] = 0;
         }
      }



   }
} while (!done);

printf("\nProcess\tBT\tWT\tTAT\n");
for (i = 0; i < n; i++) {
   tat[i] = bt[i] + wt[i];
   printf("P%d\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
```

```
    }

    return 0;
}
```

Sample Output:

```
Enter number of processes: 3
Enter burst time for each process:
P1: 10
P2: 5
P3: 8
Enter time quantum: 2

Process BT  WT  TAT
P1      10  13  23
P2      5   10  15
P3      8   13  21
```