

Lab Practical 8 Submission

Roll No. and Name: 21BEI036 Meet Patel

Course Code and Name: 2CSOE53 OS

Aim:

- a) Write a C program to implement FCFS.
- b) Write a C program to implement Round Robin.

code:

a)

```
1  #include <stdio.h>
2  void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
3      wt[0] = 0; // Waiting time for first process is 0
4      for (int i = 1; i < n; i++) {
5          wt[i] = bt[i - 1] + wt[i - 1]; // Waiting time for each process
6      }
7  }
8
9  void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
10     for (int i = 0; i < n; i++) {
11         tat[i] = bt[i] + wt[i]; // Turnaround time = burst time + waiting time
12     }
13 }
14
15 void findAvgTime(int processes[], int n, int bt[]) {
16     int wt[n], tat[n];
17     findWaitingTime(processes, n, bt, wt);
18     findTurnAroundTime(processes, n, bt, wt, tat);
19
20     // Calculate total waiting time and total turnaround time
21     int total_wt = 0, total_tat = 0;
22     for (int i = 0; i < n; i++) {
23         total_wt += wt[i];
24         total_tat += tat[i];
25     }
26
27     // Calculate average waiting time and average turnaround time
28     printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
29     for (int i = 0; i < n; i++) {
30         printf("%d\t%d\t%d\t%d\n", processes[i], bt[i], wt[i], tat[i]);
31     }
32     printf("\nAverage waiting time: %.2f\n", (float)total_wt / n);
33     printf("Average turnaround time: %.2f\n", (float)total_tat / n);
34 }
35
36 int main() {
37     int n;
38
39     // Input the number of processes
40     printf("Enter the number of processes: ");
41     scanf("%d", &n);
42
43     int processes[n];
44     int burst_time[n];
45
46     // Input the burst time for each process
47     printf("Enter burst time for each process:\n");
48     for (int i = 0; i < n; i++) {
```

"FCFS.c" 60L, 1817B

```

10     processes[i] = i + 1; // Assign process IDs (1, 2, 3, ...)
9     printf("Process %d: ", processes[i]);
8     scanf("%d", &burst_time[i]);
7 }
6
5 // Call function to calculate average waiting time and turnaround time
4 findAvgTime(processes, n, burst_time);
3
2 return 0;
1 }
60

```

input:

```

Enter the number of processes: 4
Enter burst time for each process:
Process 1: 13
Process 2: 2
Process 3: 6
Process 4: 10

```

output:

Process	Burst Time	Waiting Time	Turnaround Time
1	13	0	13
2	2	13	15
3	6	15	21
4	10	21	31

Average waiting time: 12.25
 Average turnaround time: 20.00

b)

Code:

```
1  #include <stdio.h>
2  void findWaitingTime(int processes[], int n, int bt[], int wt[], int quantum) {
3      int rem_bt[n]; // Remaining burst time of each process
4      for (int i = 0; i < n; i++) {
5          rem_bt[i] = bt[i];
6      }
7
8      int t = 0; // Current time
9      while (1) {
10         int done = 1;
11         for (int i = 0; i < n; i++) {
12             if (rem_bt[i] > 0) {
13                 done = 0;
14                 if (rem_bt[i] > quantum) {
15                     t += quantum;
16                     rem_bt[i] -= quantum;
17                 } else {
18                     t += rem_bt[i];
19                     wt[i] = t - bt[i]; // Calculate waiting time for the process
20                     rem_bt[i] = 0;
21                 }
22             }
23         }
24         if (done) break; // Exit the loop when all processes are completed
25     }
26 }
27
28 void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
29     for (int i = 0; i < n; i++) {
30         tat[i] = bt[i] + wt[i]; // Turnaround time = burst time + waiting time
31     }
32 }
33
34 void findAvgTime(int processes[], int n, int bt[], int quantum) {
35     int wt[n], tat[n];
36     findWaitingTime(processes, n, bt, wt, quantum);
37     findTurnAroundTime(processes, n, bt, wt, tat);
38
39     // Calculate total waiting time and total turnaround time
40     int total_wt = 0, total_tat = 0;
41     for (int i = 0; i < n; i++) {
42         total_wt += wt[i];
43         total_tat += tat[i];
44     }
45
46     // Calculate average waiting time and average turnaround time
47     printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
48     for (int i = 0; i < n; i++) {
```

```

31     printf("%d\t%d\t%d\t%d\n", processes[i], bt[i], wt[i], tat[i]);
30 }
29 printf("\nAverage waiting time: %.2f\n", (float)total_wt / n);
28 printf("Average turnaround time: %.2f\n", (float)total_tat / n);
27 }
26
25 int main() {
24     int n, quantum;
23
22     // Input the number of processes and time quantum
21     printf("Enter the number of processes: ");
20     scanf("%d", &n);
19     printf("Enter the time quantum: ");
18     scanf("%d", &quantum);
17
16     int processes[n];
15     int burst_time[n];
14
13     // Input the burst time for each process
12     printf("Enter burst time for each process:\n");
11     for (int i = 0; i < n; i++) {
10         processes[i] = i + 1; // Assign process IDs (1, 2, 3, ...)
9         printf("Process %d: ", processes[i]);
8         scanf("%d", &burst_time[i]);
7     }
6
5     // Call function to calculate average waiting time and turnaround time
4     findAvgTime(processes, n, burst_time, quantum);
3
2     return 0;
1 }

```

input:

```

Enter the number of processes: 4
Enter the time quantum: 2
Enter burst time for each process:
Process 1: 10
Process 2: 5
Process 3: 23
Process 4: 5

```

output:

Process	Burst Time	Waiting Time	Turnaround Time
1	10	18	28
2	5	14	19
3	23	20	43
4	5	17	22

Average waiting time: 17.25
 Average turnaround time: 28.00