

B.Siva Shirish -192324016

3. Design a CPU scheduling program with C using First Come First Served technique with the following considerations.

- a. All processes are activated at time 0.
- b. Assume that no process waits on I/O devices

Aim:

To design a CPU scheduling program using the First Come First Served (FCFS) technique where all processes are activated at time 0, and no process waits on I/O devices.

Algorithm:

1. Start the program.
2. Input the number of processes and their burst times.
3. Calculate the waiting time for each process:
 - Waiting time for the first process is 0.
 - For subsequent processes, $\text{Waiting Time}[i] = \text{Waiting Time}[i-1] + \text{Burst Time}[i-1]$.
4. Calculate the turnaround time for each process:
 - $\text{Turnaround Time}[i] = \text{Waiting Time}[i] + \text{Burst Time}[i]$.
5. Display the process details, including their burst time, waiting time, and turnaround time.
6. Compute the average waiting time and turnaround time.
7. End the program.

Procedure:

1. Include necessary headers: `<stdio.h>`.
2. Define arrays for burst times, waiting times, and turnaround times.
3. Compute waiting times and turnaround times iteratively.
4. Calculate and display average waiting and turnaround times.

CODE:

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

```
int main() {
```

```
    int n, i;
```

```
    float avg_wait = 0, avg_turnaround = 0;
```

```
    printf("Enter the number of processes: ");
```

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

```
    int burst_time[n], waiting_time[n], turnaround_time[n];
```

```
    printf("Enter the burst times for each process:\n");
```

```
    for (i = 0; i < n; i++) {
```

```
        printf("Process %d: ", i + 1);
```

```
        scanf("%d", &burst_time[i]);
```

```
    }
```

```
    waiting_time[0] = 0;
```

```
    for (i = 1; i < n; i++) {
```

```
        waiting_time[i] = waiting_time[i - 1] + burst_time[i - 1];
```

```
    }
```

```
    for (i = 0; i < n; i++) {
```

```
        turnaround_time[i] = waiting_time[i] + burst_time[i];
```

```
        avg_wait += waiting_time[i];
```

```
        avg_turnaround += turnaround_time[i];
```

```

}

avg_wait /= n;
avg_turnaround /= n;

printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
for (i = 0; i < n; i++) {
    printf("%d\t%d\t\t%d\t\t%d\n", i + 1, burst_time[i], waiting_time[i],
turnaround_time[i]);
}

printf("\nAverage Waiting Time: %.2f\n", avg_wait);
printf("Average Turnaround Time: %.2f\n", avg_turnaround);

return 0;
}

```

OUTPUT:

The screenshot shows the OnlineGDB interface. The left sidebar contains navigation links: Welcome, Siva Shirish, Create New Project, My Projects, Classroom (new), Learn Programming, Programming Questions, Upgrade, and Logout. The main area displays the execution of a C++ program. The program prompts for the number of processes (3) and their burst times (2, 3, 4). It then outputs a table of process metrics and the final average values.

Process	Burst Time	Waiting Time	Turnaround Time
1	2	0	2
2	3	2	5
3	4	5	9

Average Waiting Time: 2.33
Average Turnaround Time: 5.33