

Experiment No: 01

Experiment Name: Experiment on implementation of first come first serve (FCFS) CPU scheduling algorithm.

Objectives:

- To implement the FCFS CPU scheduling algorithm.
- Calculate waiting time & turnaround time.

Theory: CPU scheduling is the process of determining which process in the ready queue is to be allocated the CPU. One of the simplest CPU scheduling algorithms is First-Come, First-Serve (FCFS).

- It is the simplest scheduling algorithm.
- The process that arrives first in the ready queue is executed first.
- It is a non-preemptive scheduling algorithm.

Source Code:

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    int arrival[n], burst[n], completion[n], waiting[n], turnaround[n];
    // Input: Arrival time and Burst time
    printf("Enter Arrival Time and Burst Time for each process:\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d %d", &arrival[i], &burst[i]);
    }
    // Sorting based on Arrival Time (Bubble Sort)
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arrival[j] > arrival[j + 1]) {
                // Swap arrival time
                int temp = arrival[j];
                arrival[j] = arrival[j + 1];
                arrival[j + 1] = temp;
            }
            // Swap burst time
            temp = burst[j];
            burst[j] = burst[j + 1];
            burst[j + 1] = temp;
        }
    }
    int time = 0; // Tracks CPU execution time
```

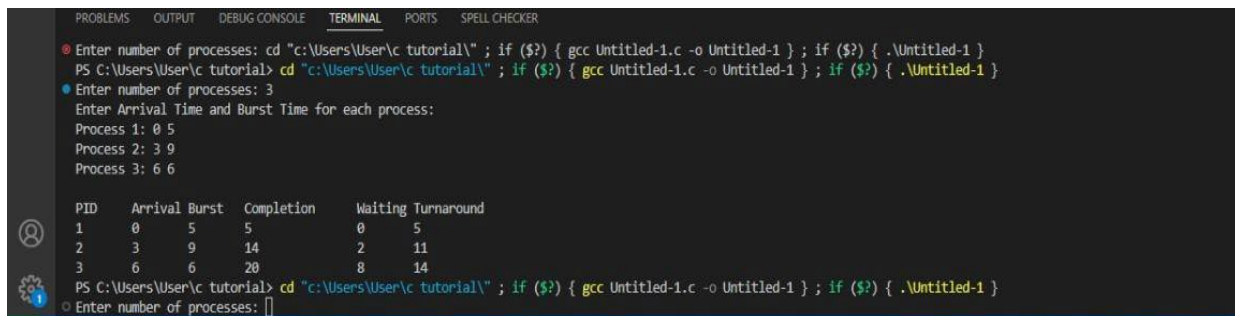
```

// Calculate Completion, Turnaround, and Waiting times
for (int i = 0; i < n; i++) {
    if (time < arrival[i]) {
        time = arrival[i]; // If CPU is idle, start when process arrives
    }
    completion[i] = time + burst[i]; // Completion Time
    turnaround[i] = completion[i] - arrival[i]; // Turnaround Time
    waiting[i] = turnaround[i] - burst[i]; // Waiting Time
    time = completion[i]; // Update CPU time
}
// Output Process Table
printf("\nPID\tArrival\tBurst\tCompletion\tWaiting\tTurnaround\n");
for (int i = 0; i < n; i++) {
    printf("%d\t%d\t%d\t%d\t%d\t%d\n", i + 1, arrival[i], burst[i], completion[i], waiting[i],
turnaround[i]);
}

return 0;
}

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER
Enter number of processes: cd "c:\Users\User\c tutorial\" ; if ($?) { gcc Untitled-1.c -o Untitled-1 } ; if ($?) { .\Untitled-1 }
PS C:\Users\User\c tutorial> cd "c:\Users\User\c tutorial\" ; if ($?) { gcc Untitled-1.c -o Untitled-1 } ; if ($?) { .\Untitled-1 }
Enter number of processes: 3
Enter Arrival Time and Burst Time for each process:
Process 1: 0 5
Process 2: 3 9
Process 3: 6 6

PID  Arrival  Burst  Completion  Waiting  Turnaround
1    0         5       5           0         5
2    3         9      14           2        11
3    6         6      20           8        14

PS C:\Users\User\c tutorial> cd "c:\Users\User\c tutorial\" ; if ($?) { gcc Untitled-1.c -o Untitled-1 } ; if ($?) { .\Untitled-1 }
Enter number of processes: 

```

Discussion: In this lab, we implemented the First-Come, First-Served (FCFS) CPU scheduling algorithm, which is one of the simplest and most intuitive scheduling methods. The algorithm schedules processes in the order they arrive, making it straightforward to implement. Despite some issues, FCFS can be effective when all processes have similar burst times. The successful implementation of the algorithm in this experiment clearly demonstrated how the scheduling order influences the waiting time and turnaround time of processes, reinforcing key theoretical concepts in CPU scheduling.