Mawlana Bhashani Science and Technology University

Lab-Report

Report No	: 07
Experiment name	: Implementation of FCFS Scheduling Algorithm
Course code	: ICT-3110
Course title	: Operating System Lab.
Date of Performance	:
Date of Submission	:

Submitted by

Name: Iqbal Hossen
ID: IT-18041
3rd year 1st semester
Session: 2017-18
Dept. of ICT
MBSTU.

Submitted To

Nazrul Islam
Assistant Professor
Dept. of ICT
MBSTU.

i) What is FCFS Scheduling Algorithm?

First Come First Serve (FCFS)

First Come First Serve is the simplest and easiest scheduling algorithm. In this algorithm, the CPU is allocated to the processes in the order they request it. The **implementation of FCFS** is easily done with a queue (a FIFO structure). When the first process enters the system it starts its execution immediately and runs till it completes its execution. As other processes enter the system, they are put at the end of the queue and wait to get the CPU. When a process finishes executing, it releases the CPU, is removed from the queue and the CPU is allocated to next process at the head of the queue.

Example:

Process	Burst Time
P1	24
P2	3
P3	3

The processes arrive in the order **P1, P2, P3** and are served as per the **FCFS algorithm**. The Gantt chart is as shown:

P1	P2	P3

24 27 30

ii) Implementation FCFS algorithm in C?

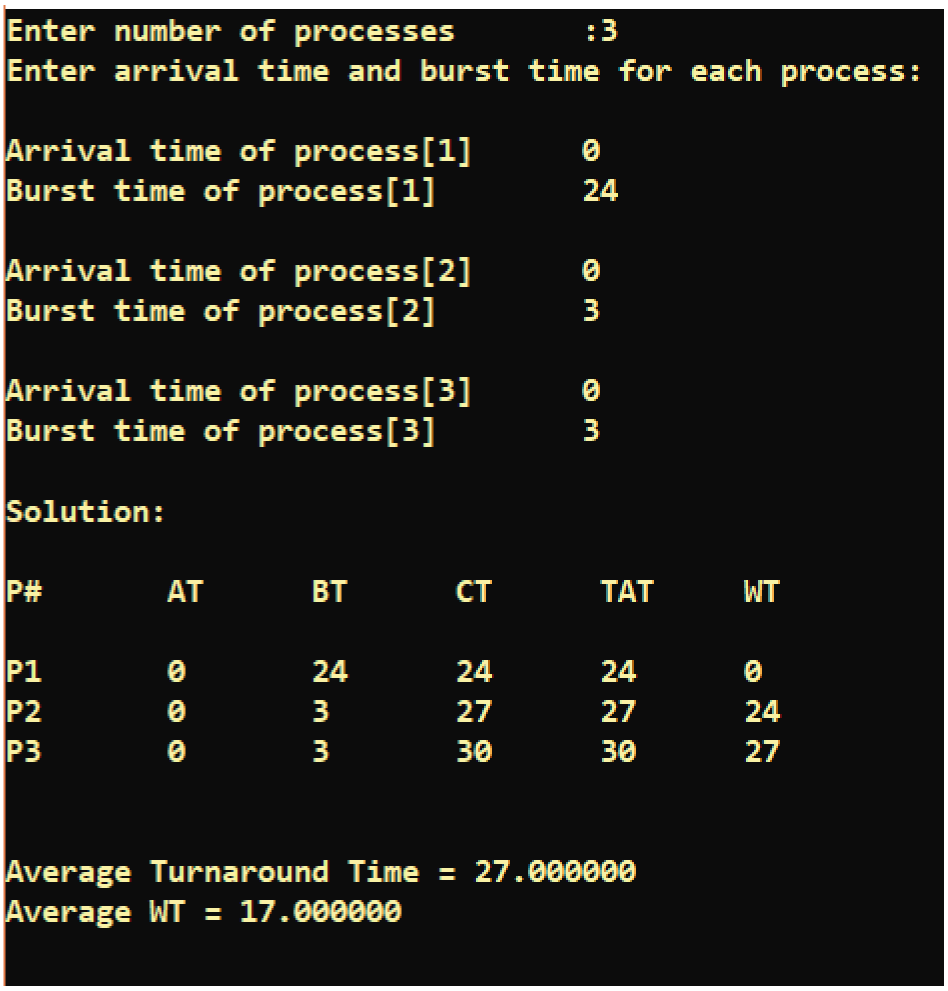
The implementation of FCFS algorithm in C is given below:

Code:

```
#include<stdio.h>

int main(){


        int bt[10]={0},at[10]={0},tat[10]={0},wt[10]={0},ct[10]={0};
        int n,sum=0;
        float totalTAT=0,totalWT=0;
        printf("Enter number of processes        :");
        scanf("%d",&n);
        printf("Enter arrival time and burst time for each process:\n\n");
        for(int i=0;i<n;i++)
        {
                printf("Arrival time of process[%d]     ",i+1);
                scanf("%d",&at[i]);
                printf("Burst time of process[%d]        ",i+1);
                scanf("%d",&bt[i]);
                printf("\n");
        }
        for(int j=0;j<n;j++)
        {
                sum+=bt[j];
                ct[j]+=sum;
        }
        for(int k=0;k<n;k++)
        {
                tat[k]=ct[k]-at[k];
                totalTAT+=tat[k];
        }
```

```
	for(int k=0;k<n;k++)

	{

		wt[k]=tat[k]-bt[k];

		totalWT+=wt[k];

	}

	printf("Solution: \n\n");

	printf("P#\t AT\t BT\t CT\t TAT\t WT\t\n\n");

	for(int i=0;i<n;i++)

	{

		printf("P%d\t %d\t %d\t %d\t %d\t 
%d\n",i+1,at[i],bt[i],ct[i],tat[i],wt[i]);

	}

	printf("\n\nAverage Turnaround Time = %f\n",totalTAT/n);

	printf("Average WT = %f\n\n",totalWT/n);

	return 0;

}
```

<u>Output:</u>



```
Enter number of processes       :3
Enter arrival time and burst time for each process:

Arrival time of process[1]      0
Burst time of process[1]        24

Arrival time of process[2]      0
Burst time of process[2]        3

Arrival time of process[3]      0
Burst time of process[3]        3

Solution:

P#        AT        BT        CT        TAT       WT

P1        0         24        24        24        0
P2        0         3         27        27        24
P3        0         3         30        30        27


Average Turnaround Time = 27.000000
Average WT = 17.000000
```