MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh,Tangail-1902

LAB REPORT

Lab Report No	: 07
Lab Report name	: Implementation of FCFS Scheduling algorithm.
Course Title	: Operating System Lab
Course Code	: ICT-3110
Date of Performance	: 20-08-2020
Date of Submission	: 28-08-2020

Submitted by,

Student Name	: Tanvir Ahmed
Student ID	: IT-18043
Session	: 2017-18

3rd Year 1st semester

Dept. of ICT

Submitted to,

Nazrul Islam

Assistant Professor

Dept. of ICT,

MBSTU.

Lab Report No. **07**

Lab Report Name: **Implementation of FCFS Scheduling algorithm .**

Objectives:

i What is FCFS Scheduling algorithm.
iiHow to implementation in C

Theory: First Come First Served (FCFS) is a **Non-Preemptive** scheduling algorithm. FIFO (First In First Out) strategy assigns priority to process in the order in which they request the processor. The process that requests the CPU first is allocated the CPU first. This is easily implemented with a FIFO queue for managing the tasks. As the process come in, they are put at the end of the queue. As the CPU finishes each task, it removes it from the start of the queue and heads on to the next task.

Turn Around Time = Completion Time - Arrival Time

Waiting Time = Turnaround time – Burst Time

Advantages of FCFS

Simple

Easy

First come, First serv

Disadvantages of FCFS

1.The scheduling method is non preemptive, the process will run to the completion.

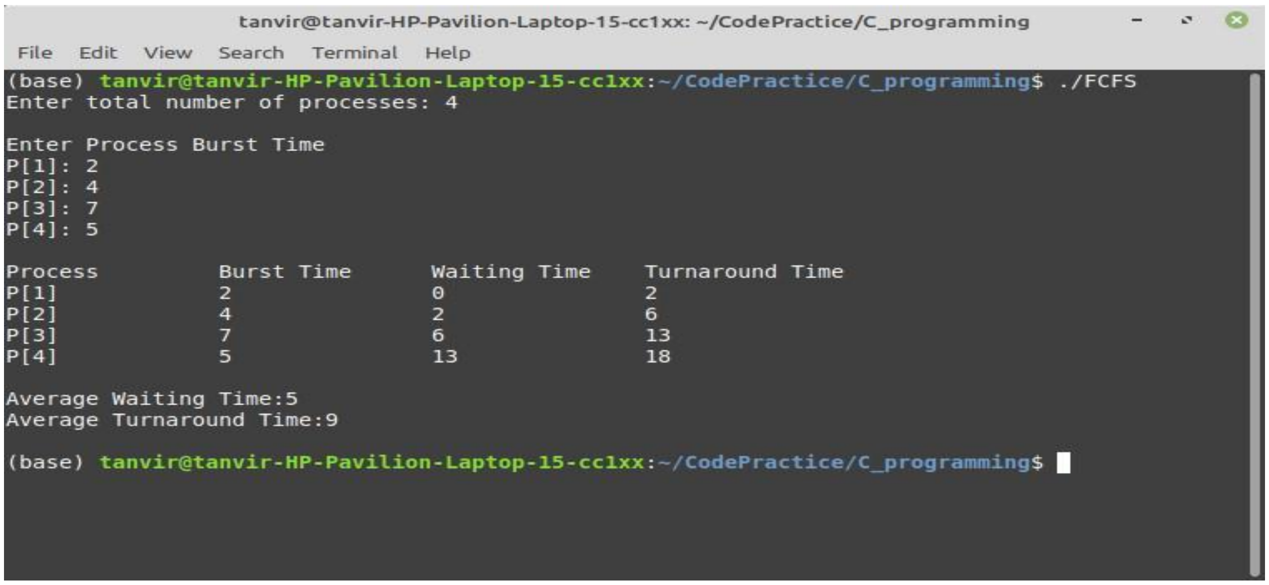
2.Due to the non-preemptive nature of the algorithm, the problem of starvation may occur.

3.Although it is easy to implement, but it is poor in performance since the average waiting time is higher as compare to other scheduling algorithms.

Corresponding Code:

```
#include<stdio.h>

int main()

{

   int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;

   printf("Enter total number of processes: ");

   scanf("%d",&n);


   printf("\nEnter Process Burst Time\n");

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

   {

      printf("P[%d]: ",i+1);

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

   }

   wt[0]=0;    //waiting time for first process is 0


   //calculating waiting time

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

   {

      wt[i]=0;

      for(j=0;j<i;j++)

         wt[i]+=bt[j];

   }

   printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
//calculating turnaround time
   for(i=0;i<n;i++)
   {
      tat[i]=bt[i]+wt[i];
      avwt+=wt[i];
      avtat+=tat[i];
      printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
   }
   avwt/=i;
   avtat/=i;
   printf("\n\nAverage Waiting Time:%d",avwt);
   printf("\nAverage Turnaround Time:%d",avtat);
   printf("\n\n");
   return 0;
}
```

Output:


tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx: ~/CodePractice/C_programming
File Edit View Search Terminal Help

```
(base) tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx:~/CodePractice/C_programming$ ./FCFS
Enter total number of processes: 4

Enter Process Burst Time
P[1]: 2
P[2]: 4
P[3]: 7
P[4]: 5

Process         Burst Time      Waiting Time    Turnaround Time
P[1]            2               0               2
P[2]            4               2               6
P[3]            7               6               13
P[4]            5               13              18

Average Waiting Time:5
Average Turnaround Time:9

(base) tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx:~/CodePractice/C_programming$
```

Conclusion : FCFS stands for First Come First Serve. It is the simplest form of a CPU scheduling algorithm.

A real-life example of the FCFS method is buying a movie ticket on the ticket counter In this scheduling algorithm, a person is served according to the queue manner. The person who arrives first in the queue first buys the ticket and then the next one. This will continue until the last person in the queue purchases the ticket. Using this algorithm, the CPU process works in a similar manner.

Although FCFS is a very easy algorithm it is not very efficient. The Average Waiting Time is high. Not an ideal technique for time-sharing systems. It is a Non-Preemptive CPU scheduling algorithm, so after the process has been allocated to the CPU, it will never release the CPU until it finishes executing.