Experiment no: 5

Name: Saloni Vishwakarma
Batch-Roll no: C1-13
Semester-Section:4th-C
Subject: OS(Operating System) Lab
Date of execution: 10 May 2023

Aim: To write and execute C programs to demonstrate different CPU scheduling algorithms.

5A) To write and execute C programs to demonstrate First Come First Served CPU scheduling algorithm.

Theory:

FCFS (First-Come-First-Serve) is a non-preemptive scheduling algorithm used in operating systems to determine the order in which processes are executed on a CPU. It follows a simple rule: the process that arrives first is scheduled first and executed until completion before the next process is executed.

In FCFS, the processes are executed in the order they arrive, forming a queue. The CPU is assigned to the first process in the queue until it completes or performs an I/O operation. Once a process completes or blocks for I/O, the CPU is passed to the next process in the queue.

Key features of the FCFS algorithm:

Non-preemptive: Once a process is allocated the CPU, it continues to execute until completion or I/O request without interruption.

FIFO (First-In-First-Out): The processes are scheduled in the order they arrive, forming a queue.

Simple implementation: FCFS is easy to understand and implement, requiring minimal scheduling overhead.

However, FCFS has some drawbacks:

Convoy effect: If a long process arrives before short processes, it can cause delays for subsequent processes, leading to poor average waiting time.

Poor utilization: The CPU may remain idle if a long process arrives early, blocking shorter processes in the queue.

Overall, FCFS is a simple and intuitive scheduling algorithm.

Code and Output:

```
#include<stdio.h>
int main()
{
   int n;
   printf("\n Enter number of Processes: ");
   scanf("%d",&n);
   int BT[n];
   int AT[n],Fin[n],TAT[n],WT[n],temp,Process[n],i,j;
   double avg_tat,avg_wt;
   avg_tat=avg_wt=0;
   for(i=0;i<n;i++)
      Process[i]=i+1;

   printf("\n Enter Burst Time: ");
   for(int i=0;i<n;i++)
      scanf("%d",&BT[i]);
   printf("\n Enter Arrival Time: ");
   for(int i=0;i<n;i++)
      scanf("%d",&AT[i]);

   for (i = 0; i < n - 1; i++)
     for (j = 0; j < n - i - 1; j++)
        if (AT[j] > AT[j + 1]){
        //sorting AT
        temp=AT[j];
        AT[j]=AT[j+1];
        AT[j+1]=temp;

        //sorting BT
        temp=BT[j];
        BT[j]=BT[j+1];
        BT[j+1]=temp;
```

```
        //sorting process
        temp=Process[j];
        Process[j]=Process[j+1];
        Process[j+1]=temp;

        }

     Fin[0]=AT[0]+BT[0];
    for(i=1;i<n;i++)
    {
       Fin[i]=Fin[i-1]+BT[i];
    }
    printf("\n");
    for(i=0;i<n;i++)
       TAT[i]=Fin[i]-AT[i];

    for(i=0;i<n;i++)
       WT[i]=TAT[i]-BT[i];
    printf("+-----------+-----------+-----------+------------+------------+------------+\n");
    printf("| Processes |  Arrival  | BurstTime |TurnAroundTime| WaitingTime | FinishTime
|\n");
    printf("+-----------+-----------+-----------+------------+------------+------------+\n");
    for(i = 0; i < n; i++) {
       printf("| %-9d | %-9d | %-9d | %-11d | %-11d | %-11d |\n", Process[i], AT[i], BT[i],
TAT[i], WT[i], Fin[i]);
       printf("+-----------+-----------+-----------+------------+------------+------------+\n");
    }
    printf("\n");
    printf("\n\t     Gantt Chart \n----------------------------------------\n");

    for(i=0;i<n;i++)
       printf("|%d       ",Process[i]);
    printf("|");
    printf("\n----------------------------------------");
    printf("\n|%d",AT[0]);
    for(i=0;i<n;i++)
       printf("     |%d",Fin[i]);
```

```
    printf("   |");
    printf("\n");
    for(i=0;i<n;i++)
        avg_tat = avg_tat+(double)TAT[i];
    avg_tat=avg_tat/n;
    for(i=0;i<n;i++)
        avg_wt = avg_wt+(double)WT[i];
    avg_wt=avg_wt/n;
    printf("\n Avg TAT:%lf",avg_tat);
    printf("\n Avg Waiting Time:%lf",avg_wt);
    avg_wt=avg_wt/4;
    printf("\n");
}
```

```
 Enter number of Processes: 5

 Enter Burst Time: 6 2 8 3 4

 Enter Arrival Time: 2 5 1 0 4

+-----------+-----------+-----------+-------------+-------------+-------------+
| Processes |  Arrival  | BurstTime |TurnAroundTime| WaitingTime | FinishTime |
+-----------+-----------+-----------+-------------+-------------+-------------+
| 4         | 0         | 3         | 3           | 0           | 3           |
+-----------+-----------+-----------+-------------+-------------+-------------+
| 3         | 1         | 8         | 10          | 2           | 11          |
+-----------+-----------+-----------+-------------+-------------+-------------+
| 1         | 2         | 6         | 15          | 9           | 17          |
+-----------+-----------+-----------+-------------+-------------+-------------+
| 5         | 4         | 4         | 17          | 13          | 21          |
+-----------+-----------+-----------+-------------+-------------+-------------+
| 2         | 5         | 2         | 18          | 16          | 23          |
+-----------+-----------+-----------+-------------+-------------+-------------+


             Gantt Chart
-----------------------------------------
|4       |3       |1       |5       |2       |
-----------------------------------------
|0     |3     |11     |17     |21     |23    |

 Avg TAT:12.600000
 Avg Waiting Time:8.000000


...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion: Through this practical, we successfully implemented the First Come First Serve (FCFS) algorithm in the C programming language. FCFS provided a simple and straightforward approach to process scheduling, executing the processes in the order of their arrival.