Lab Report No : 07

Lab Report Name : Implementation of FCFS scheduling algorithm

Objective :

i) What is FCFS scheduling algorithm.
ii) How can implementation in c.

Theory:

In the "First come first serve" scheduling algorithm, as the name suggests, the process which arrives first, gets executed first, or we can say that the process which requests the CPU first, gets the CPU allocated first.

- First Come First Serve, is just like **FIFO**(First in First out) Queue data structure, where the data element which is added to the queue first, is the one who leaves the queue first.
- This is used in *Batch Systems.*
- It's **easy to understand and implement** programmatically, using a Queue data structure, where a new process enters through the **tail** of the queue, and the scheduler selects process from the **head** of the queue.

A perfect real life example of FCFS scheduling is buying tickets at ticket counter.

Example of FCFS scheduling

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.

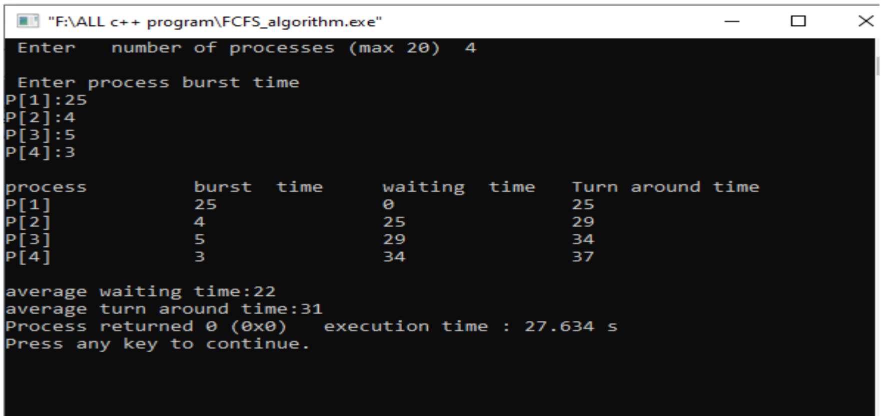
Implementaion First come First Serve Scheduling Algorithm in c.

```
#include<bits/stdc++.h>

#include<stdio.h>

int main()

{
```

```
int k,l ,x,but[20],wt[20],tat[20],awt=0,atat=0;

printf(" Enter   number of processes (max 20)  ");
scanf("%d",&x);

printf("\n Enter process burst time\n");
for( k=0;k<x;k++)
{
   printf("P[%d]:",k+1);
   scanf("%d",&but[k]);
}

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

//calculating waiting time
for(k=1;k<x;k++)
{
   wt[k]=0;
   for(l=0;l<k;l++)
       wt[k]+=but[l];
}

printf("\nprocess\t\tburst  time\twaiting  time\tTurn around time");

//calculating turnaround time
for(k=0;k<x;k++)
{
   tat[k]=but[k]+wt[k];
   awt+=wt[k];
   atat+=tat[k];
   printf("\nP[%d]\t\t%d\t\t%d\t\t%d",k+1,but[k],wt[k],tat[k]);
}
```

```
    awt/=k;

    atat/=k;

    printf("\n\naverage waiting time:%d",awt);

    printf("\naverage turn around time:%d",atat);


    return 0;

}
```

OUtPUt:



```
 Enter   number of processes (max 20)  4

 Enter process burst time
P[1]:25
P[2]:4
P[3]:5
P[4]:3

process         burst  time      waiting  time    Turn around time
P[1]            25               0                25
P[2]            4                25               29
P[3]            5                29               34
P[4]            3                34               37

average waiting time:22
average turn around time:31
Process returned 0 (0x0)   execution time : 27.634 s
Press any key to continue.
```