

Write a C program to solve the following problem:

Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO Order, is: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the SCAN disk-scheduling algorithms?

Solution : First come, first served (FCFS) is an operating system process scheduling algorithm

Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO Order, is: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the SCAN disk-scheduling algorithms?

Solution : First come, first served (FCFS) is an operating system process scheduling algorithm

and a network routing management mechanism that automatically executes queued requests and processes by the order of their arrival. With first come, first served, what comes first is handled first; the next request in line will be executed once the one before it is complete.

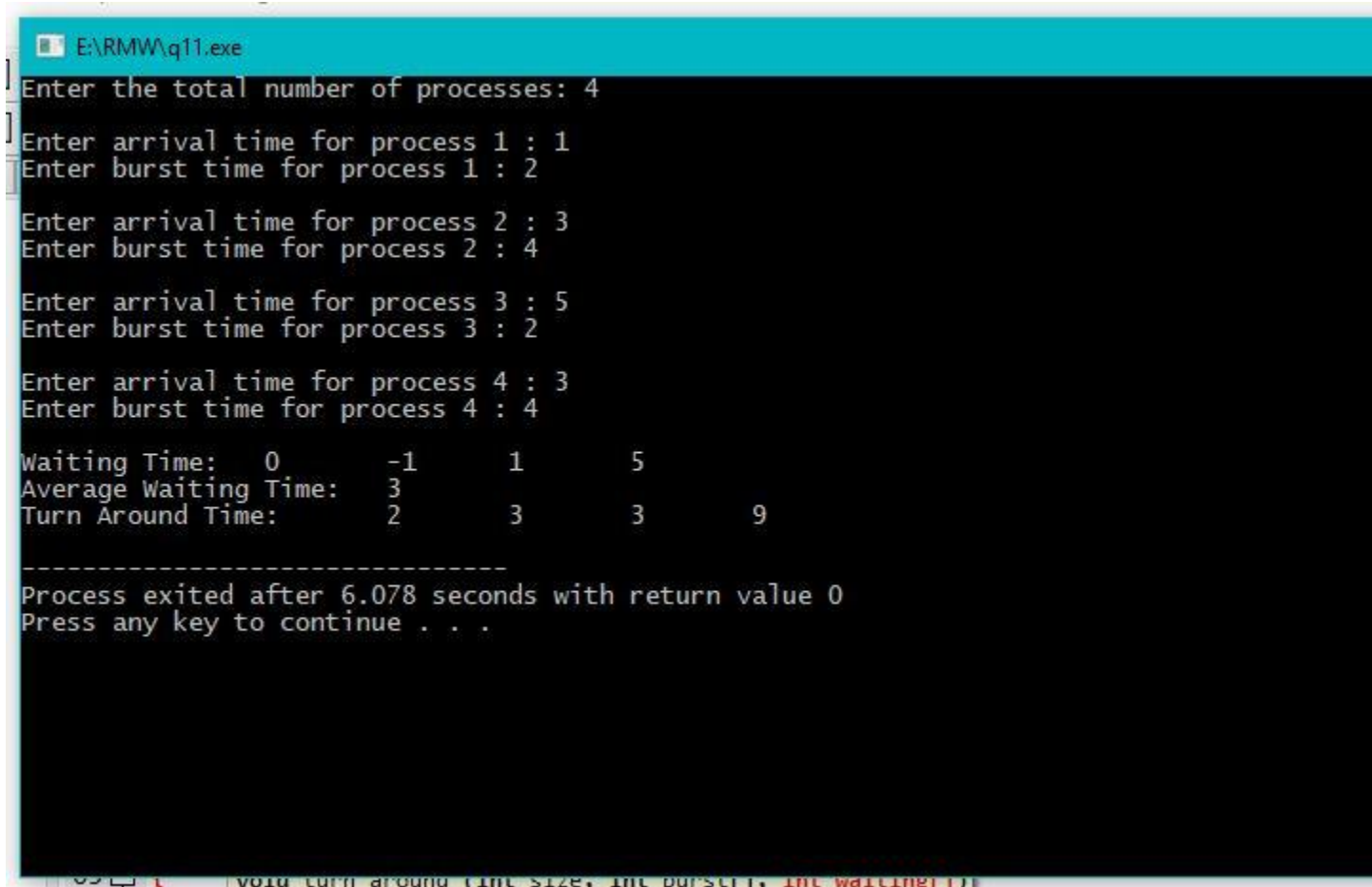
FCFS provides an efficient, simple and error-free process scheduling algorithm that saves valuable CPU resources. It uses non-preemptive scheduling in which a process is automatically queued and processing occurs according to an incoming request or process order. FCFS derives its concept from real-life customer service.

Let's take a look at how FCFS process scheduling works. Suppose there are three processes in a

queue: P1, P2 and P3. P1 is placed in the processing register with a waiting time of zero seconds and 10 seconds for complete processing. The next process, P2, must wait 10 seconds and is placed in the processing cycle until P1 is processed. Assuming that P2 will take 15 seconds to complete, the final process, P3, must wait 25 seconds to be processed. FCFS may not be the fastest process scheduling algorithm, as it does not check for priorities associated with processes. These priorities may depend on the processes' individual execution times.

Console Output :

=



```
E:\RMW\q11.exe
Enter the total number of processes: 4
Enter arrival time for process 1 : 1
Enter burst time for process 1 : 2
Enter arrival time for process 2 : 3
Enter burst time for process 2 : 4
Enter arrival time for process 3 : 5
Enter burst time for process 3 : 2
Enter arrival time for process 4 : 3
Enter burst time for process 4 : 4

Waiting Time:    0      -1      1      5
Average Waiting Time:  3
Turn Around Time:  2      3      3      9

-----
Process exited after 6.078 seconds with return value 0
Press any key to continue . . .
```

Code Snap :: (Main)

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

Console Output : =

Code Snap :: (Main)

```
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");

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("\n\nAverage Turnaround Time:%d",avtat); }

```

Question 14 :

If a teacher is being served at the food mess and during the period when he is being served, another teacher comes, then that teacher would get the service (food) next. This process might continue leading to increase in waiting time of students to get food. Ensure in your program that the waiting time of students is minimized.

Solution is: Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler chooses the tasks to work as per the priority, which is different from other types of scheduling, for example, a simple round robin. Priority scheduling involves priority assignment to every process, and processes with higher priorities are carried out first, whereas tasks with equal priorities are carried out on a first-come-first-served (FCFS) or round robin basis. An example of a general-priority-scheduling algorithm is the shortest-job-first (SJF) algorithm

Priorities can be either dynamic or static. Static priorities are allocated during creation, whereas dynamic priorities are assigned depending on the behavior of the processes while in the system. To illustrate, the scheduler could favor input/output (I/O) intensive tasks, which lets expensive requests to be issued as soon as possible.

Priorities may be defined internally or externally. Internally defined priorities make use of some measurable quantity to calculate the priority of a given process. In contrast, external priorities are

defined using criteria beyond the operating system (OS), which can include the significance of the process, the type as well as the sum of resources being utilized for computer use, user preference, commerce and other factors like politics, etc. **Console output :-**

```

C:\Users\admin-sheet\Desktop\d.exe
Enter Total Number of Process:4

Enter Burst Time and Priority

P[1]
Burst Time:1
Priority:2

P[2]
Burst Time:3
Priority:4

P[3]
Burst Time:
5
Priority:3

P[4]
Burst Time:1
Priority:2

Process      Burst Time      Waiting Time      Turnaround Time
P[1]          1                0                 1
P[4]          1                1                 2
P[3]          5                2                 7
P[2]          3                7                10

Average Waiting Time=2
Average Turnaround Time=5

-----
Process exited after 8.18 seconds with return value 0
Press any key to continue . . .

```

Code snap:

```

for(i= 0 ;i<n;i++)
{
pos=i;
for(j=i+ 1 ;j<n;j++)
{
if(pr[j]<pr[pos])
pos=j;
}
temp=pr[i];
pr[i]=pr[pos];
pr[pos]=temp;
temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;
temp=p[i];
p[i]=p[pos];
p[pos]=temp;
}

```

```

}
wt[ 0 ]= 0 ;
for(i= 1 ;i<n;i++)
{
wt[i]= 0 ;
for(j= 0 ;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
}
avg_wt=total/n;
total= 0 ;
printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i= 0 ;i<n;i++)
{
tat[i]=bt[i]+wt[i];
total+=tat[i];
printf("\nP[%d]\t\t %d\t\t %d\t\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
}
avg_tat=total/n;
printf("\n\nAverage Waiting Time=%d",avg_wt);
printf("\n\nAverage Turnaround Time=%d\n",avg_tat);

```