FCFS Algorithm

Introduction

First Come First Serve (FCFS) is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival. It is the easiest and simplest CPU scheduling algorithm. In this type of algorithm, processes which requests the CPU first get the CPU allocation first. This is managed with a FIFO queue. The full form of FCFS is First Come First Serve.

Characteristics of FCFS method

- It supports non-pre-emptive scheduling algorithm.
- Jobs are always executed on a first come, first-serve basis.
- It is easy to implement and use.
- This method is poor in performance, and the general wait time is quite high.

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.

ALLOCATED QUESTION

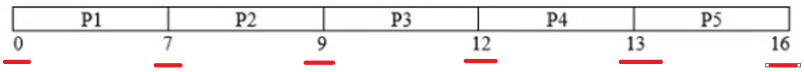
Write a C program to implement the First Come First Served Scheduling policy. The program should accept number of processes, burst time and arrival time from the user and generate the order of execution, waiting time and turnaround time. The program also should calculate the average waiting time and average turnaround time.

PseudoCode/Algorithm

1. Input the processes along with their arrival time (AT) & burst time (BT).

2. Sort the processes according to their arrival time using bubble sort

3. Find out the time point for every process in Gantt chart (gval)



4. Calculate waiting time wt[n] using gval

5. Calculate turnaround time tt[n] using gval

6. Find average waiting time and average turnaround time

7. Draw Gantt chart for illustration purpose

Code

```
#include<stdio.h>
int main()
{
	int n;
	printf("enter the number of process: ");
	scanf("%d",&n);
	int po[n],a[n],b[n],w[n],gval[n+1],tt[n],temp,i,j;
	float att=0,awt=0;
	gval[0]=0;
	for(i=0;i<n;i++
	     a[i]=0; b[i]=0; w[i]=0;po[i]=i+1;
	for(i=0;i<n;i++)
	{
			printf("-----------------\n");
			printf("Process %d\n",i+1);
			printf("Arrival Time: ");
			scanf("%d",&a[i]);
			printf("Burst Time: ");
			scanf("%d",&b[i]);

	}

	for(i=0;i<n;i++)
	{
			for(j=i+1;j<n;j++)
			{
				if(a[j]<a[i])
				{
					temp=a[i];
					a[i]=a[j];
					a[j]=temp;
					temp=po[i];
					po[i]=po[j];
					po[j]=temp;
					temp=b[i];
					b[i]=b[j];
					b[j]=temp;
				}
			}
	}

	for(i=1;i<n+1;i++)
		gval[i]=gval[i-1]+b[i-1];

	for(i=1;i<n;i++)
		w[i]=gval[i]-a[i];

	for(i=0;i<n;i++)
		tt[i]=gval[i+1]-a[i];


	printf("\n\nP\tAT\tBT\tWT\tTT\n");
	printf("-----------------------------------\n");
	for(i=0;i<n;i++)
	{
```

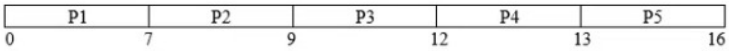
```
            printf("%d",po[i]);
            printf("\t%d",a[i]);
            printf("\t%d",b[i]);
            printf("\t%d",w[i]);
            printf("\t%d\n",tt[i]);
      }

      for(i=0;i<n;i++)
      {
                  att+=tt[i];
                  awt+=w[i];
      }
      att/=n;
      awt/=n;
      printf("\n\nAverage turnaround time:\t%f\n",att);
      printf("Average wait time:\t%f\n\n\n\n",awt);

      printf("Gantt chart\n");
      for(i=0;i<n;i++)
        printf("----------------");
      printf("\n");

      for(i=0;i<n;i++)
       printf("|\tP%d\t",po[i]);
      printf("|\n");

      for(i=0;i<n;i++)
        printf("----------------");
      printf("\n");

      for(i=0;i<=n;i++)
            printf("%d\t \t",gval[i]);
      printf("\n\n\n");
}
```

Output

Example Problem 1:

1. Consider the following set of processes with a length of the CPU burst time given in milliseconds.

Process	Arrival Time	Burst Time	Waiting Time	Turnaround Time
P1	0	7	0	7-0=7
P2	3	2	4	9-3=6
P3	4	3	5	12-4=8
P4	4	1	8	13-4=9
P5	5	3	8	16-5=11



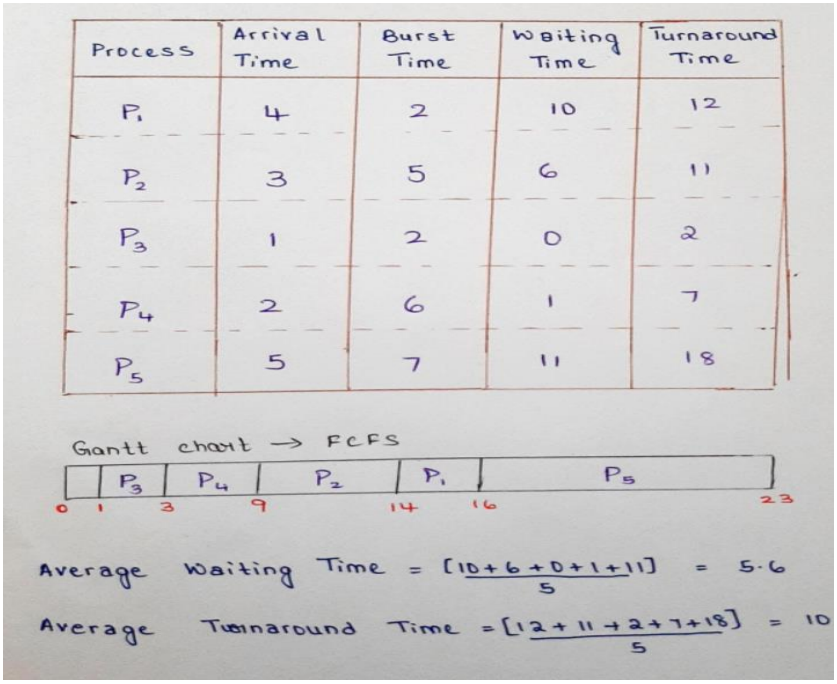
average waiting time: (0+4+5+8+8)/5 = 25/5 = 5

average turnaround time = (7+6+8+9+11)/5 = 41/5 = 8.2

```
(base) ramanan@ramanan-VirtualBox:~/Lab programs/End Sem$ ./fcfs
enter the number of process: 5
-----------------
Process 1
Arrival Time: 0
Burst Time: 7
-----------------
Process 2
Arrival Time: 3
Burst Time: 2
-----------------
Process 3
Arrival Time: 4
Burst Time: 3
-----------------
Process 4
Arrival Time: 4
Burst Time: 1
-----------------
Process 5
Arrival Time: 5
Burst Time: 3

P       AT      BT      WT      TT
-------------------------------------
1       0       7       0       7
2       3       2       4       6
3       4       3       5       8
4       4       1       8       9
5       5       3       8       11

Average turnaround time:        8.200000
Average wait time:      5.000000


Gantt chart
-------------------------------------------------------------------------------
|       P1      |       P2      |       P3      |       P4      |       P5      |
-------------------------------------------------------------------------------
0               7               9               12              13              16
```

Output

Example Problem 2:



Process	Arrival Time	Burst Time	Waiting Time	Turnaround Time
P_1	4	2	10	12
P_2	3	5	6	11
P_3	1	2	0	2
P_4	2	6	1	7
P_5	5	7	11	18

Gantt chart → FCFS

Average Waiting Time = $\frac{[10+6+0+1+11]}{5}$ = 5.6

Average Turnaround Time = $\frac{[12+11+2+7+18]}{5}$ = 10

```
(base) ramanan@ramanan-VirtualBox:~/Lab programs/End Sem$ ./fcfs
enter the number of process: 5
-----------------
Process 1
Arrival Time: 4
Burst Time: 2
-----------------
Process 2
Arrival Time: 3
Burst Time: 5
-----------------
Process 3
Arrival Time: 1
Burst Time: 2
-----------------
Process 4
Arrival Time: 2
Burst Time: 6
-----------------
Process 5
Arrival Time: 5
Burst Time: 7


P       AT      BT      WT      TT
-------------------------------------
3       1       2       0       2
4       2       6       1       7
2       3       5       6       11
1       4       2       10      12
5       5       7       11      18


Average turnaround time:        10.000000
Average wait time:      5.600000



Gantt chart
-------------------------------------------------------------------------------
|       P3      |       P4      |       P2      |       P1      |       P5      |
-------------------------------------------------------------------------------
1               3               9               14              16              23
```

Conclusion

While FCFS is the simplest form of a CPU scheduling algorithm & easy to implement, it also has high Average wait time and is not ideal for time sharing system as short processes that arrival late will have a long waiting time which makes FCFS not very efficient.