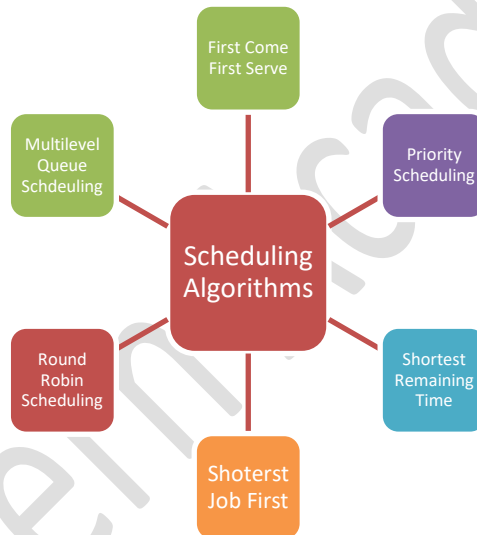


CPU Scheduling Algorithms (FCFS) & (SJF)

What is the Process? Program under execution.

There are mainly six types of process scheduling algorithms:

1. First Come First Serve (FCFS)
2. Shortest-Job-First (SJF) Scheduling
3. Shortest Remaining Time
4. Priority Scheduling
5. Round Robin Scheduling
6. Multilevel Queue Scheduling



First Come First Serve (FCFS)

First Come First Serve is the full form of FCFS. It is the easiest and most simple CPU scheduling algorithm. In this type of algorithm, the process which requests the CPU gets the CPU allocation first. This scheduling method can be managed with a FIFO queue.

Characteristics of FCFS method:

1. Jobs are always executed on a first-come, first-serve basis
2. It is easy to implement and use.
3. However, this method is poor in performance, and the general wait time is quite high.

✚ Solved Example#1 (Simple FCFS):

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1, P_2, P_3
The Gantt Chart for the schedule is:



- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

✚ Suppose that the processes arrive in the order: P_2, P_3, P_1

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order:

$$P_2, P_3, P_1$$

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$
- Much better than previous case

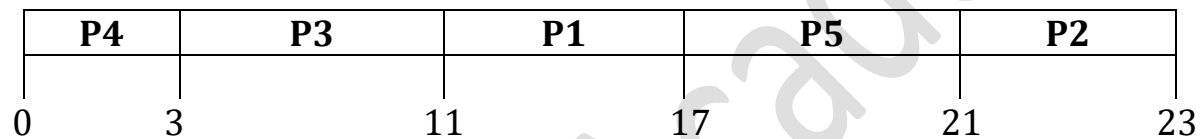
Solved Example#2 (FCFS Example):

Draw the Gantt chart. Then, Calculate the Waiting Time for each Process, and the Average waiting time. Using FCFS Algorithm.

Process	Arrival Time	Burst Time
P1	2	6
P2	5	2
P3	1	8
P4	0	3
P5	4	4

Answer:

Gantt chart:



Waiting Time for each Process:

Waiting time = Start time - Arrival time

Waiting time for P4 = 0-0 =0

Waiting time for P3 = 3-1 =2

Waiting time for P1 = 11-2 =9

Waiting time for P5 = 17-4 =13

Waiting time for P2 = 21-5 =16

Average Waiting time= (0+2+9+13+16) / 5 = 40/5 = 8 msec

OR

Waiting time [i] =End time [i] - Arrival time[i] - Burst time[i]

Waiting time for P4 = 3 - 0 - 3 =0

Waiting time for P3 = 11 - 1 - 8 =2

Waiting time for P1 = 17 - 2 - 6 =9

Waiting time for P5 = 21 - 4 - 4 =13

Waiting time for P2 = 23 - 5 - 2 =16

Average Waiting time= (0+2+9+13+16) / 5 = 40/5 = 8 msec

Exercise

Process	Arrival Time	Burst Time
P1	1	4
P2	3	8
P3	1	6
P4	0	2
P5	6	12

Average Waiting time= 6.2 msec

FCFS Algorithm:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process name, arrival time, and the burst time

Step 4: Sort the processes according to its arrival time

Step 5: Set the waiting of the first process as _0'

Step 6: For each process in the Ready Q calculate

$$\text{Waiting time (n)} = \sum_0^n \text{burst time} - \text{Arrival time (n)} - \text{Burst time (n)}$$

Step 7: Calculate Average waiting time = Total waiting Time / Number of process

Step 8: Stop the process

FCFS Code in C++:

```
#include <iostream>
#include <iomanip>
using namespace std;
struct process
{
    int id, at, bt, wt;
};

void main()
{
    process *arr;
    int num;
    cout<<" Enter number of Processes \n";
    cin>>num;
    arr=new process[num]; //Dynamic Array of Struct

    cout<<" \tArrival time \t Burst time"<<endl;

    for (int i=0;i<num;i++) //Enter Arrival time & Burst time
    {
        arr[i].id=i+1;
        cout<<"P"<<i+1<<"\t";
```

THEORY OF OPERATING SYSTEM | 3RD LEVEL | LAB 4

```
        cin>>arr[i].at>>arr[i].bt;
    }

    //Sorting Process in ascending order according to its arrival time
using Bubble Sort
process temp;

for (int i=0 ;i<num-1 ;i++)
{
    for (int j=i+1 ; j<num ; j++)
    {
        if (arr[i].at >arr[j].at)
        {
            temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
        }
    }
}

// Calculating Waiting Time for each Process
//Waiting time [i] =End time [i] - Arrival time[i] - Burst time[i]
int total=0;
for (int i=0;i<num;i++)
{
    total+=arr[i].bt;
    arr[i].wt = total - arr[i].at - arr[i].bt;

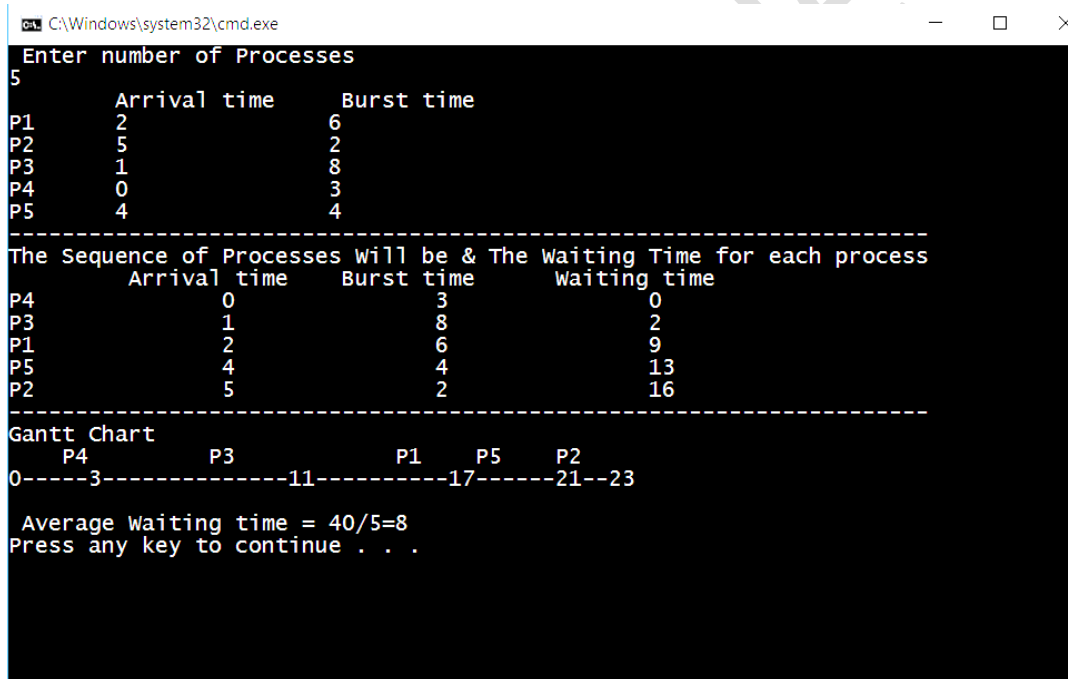
}
cout<<"-----
----- \n";
cout<<"The Sequence of Processes Will be & The Waiting Time for each
process \n";
cout<<"\t Arrival time \t Burst time \t Waiting time \n";
for (int i=0;i<num;i++)
{
    cout<<"P"<<arr[i].id<<"\t \t"<<arr[i].at<<" \t
\t"<<arr[i].bt<<"\t \t"<<arr[i].wt<<endl;
}
cout<<"-----
----- \n";
// Drawing Gantt chart
cout<<"Gantt Chart \n ";
for (int i=0;i<num ; i++)
{
    cout<<setw(arr[i].bt)<<"P"<<arr[i].id<<"\t";
}
cout<<endl<<"0";
total=0;
for (int i=0;i<num ; i++)
{
```

THEORY OF OPERATING SYSTEM | 3RD LEVEL | LAB 4

```
total+=arr[i].bt;
cout<<setfill('-')<<setw(arr[i].bt*2)<<total;
}
//Calculating the Average Waiting Time
int sum=0;
for(int i=0;i<num;i++)
{
    sum+=arr[i].wt;
}
cout<<"\n \n Average Waiting time =
"<<sum<<"/"<<num<<"="<<(float)sum/num<<endl;

delete[] arr;
}
```

Output:



```
C:\Windows\system32\cmd.exe
Enter number of Processes
5
Arrival time    Burst time
P1              2              6
P2              5              2
P3              1              8
P4              0              3
P5              4              4

-----
The Sequence of Processes Will be & The Waiting Time for each process
Arrival time    Burst time    Waiting time
P4              0              3              0
P3              1              8              2
P1              2              6              9
P5              4              4              13
P2              5              2              16
-----

Gantt Chart
P4      P3      P1      P5      P2
0-----3-----11-----17-----21-----23

Average waiting time = 40/5=8
Press any key to continue . . .
```

✚ Shortest Job First (SJF) Algorithm

- Each process is associated the length of its next CPU burst.
- According to the algorithm the scheduler selects the process with the shortest time.
- There are two types of SJF Algorithm: Preemptive and non-Preemptive SJF.

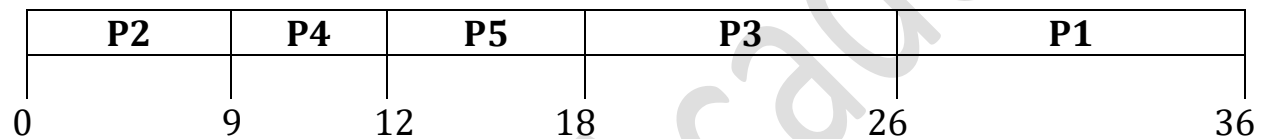
Solved Example#2 (Non-Preemptive SJF Example):

Draw the Gantt chart. Then, Calculate the Waiting Time for each Process, and the Average waiting time. Using Non-Preemptive SJF Algorithm.

Process	Arrival Time	Burst Time
P1	5	10
P2	0	9
P3	3	8
P4	5	3
P5	6	6

Answer:

Gantt chart:



Calculating Waiting Time for each Process:

Waiting time [i] = End time [i] - Arrival time[i] - Burst time[i]

Waiting time for P2 = 9 - 0 - 9 = 0

Waiting time for P4 = 12 - 5 - 3 = 4

Waiting time for P5 = 18 - 6 - 6 = 6

Waiting time for P3 = 26 - 3 - 8 = 15

Waiting time for P1 = 36 - 5 - 10 = 21

Average Waiting time = (0+4+6+15+21) / 5 = 46/5 = 9.2 msec

SJF (Non-Preemptive) Algorithm:

Step 1: Start.

Step 2: Accept the number of processes in the ready Queue.

Step 3: For each process in the ready Q, assign the process name, arrival time, and the burst time

Step 4: Sort the processes according to its arrival time.

Step 4: Sort the processes again according to its burst time.

Step 5: Set the waiting of the first process as '0'

Step 6: For each process in the Ready Q calculate

$$\text{Waiting time (n)} = \sum_{i=0}^{n-1} \text{burst time} - \text{Arrival time (n)}$$

Step 7: Calculate Average waiting time = Total waiting Time / Number of process

Step 8: Stop the processes.

THEORY OF OPERATING SYSTEM | 3RD LEVEL | LAB 4

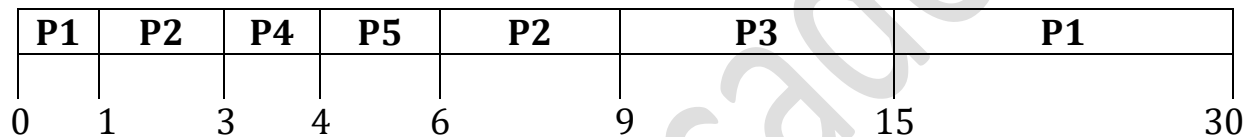
Solved Example#3: (Preemptive SJF Example):

Draw the Gantt chart. Then, Calculate the Waiting Time for each Process, and the Average waiting time. Using Preemptive SJF Algorithm.

Process	Arrival Time	Burst Time
P1	0	16
P2	1	5
P3	2	6
P4	3	1
P5	4	2

Answer:

Gantt chart:



Calculating waiting time for each process:-

Waiting time [i] = End time [i] - Arrival time[i] - Burst time[i]

Waiting time for P1 = $30 - 0 - 16 = 14$

Waiting time for P2 = $9 - 1 - 5 = 3$

Waiting time for P3 = $15 - 2 - 6 = 7$

Waiting time for P4 = $4 - 3 - 1 = 0$

Waiting time for P5 = $6 - 4 - 2 = 0$

Average Waiting time = $(14 + 3 + 7 + 0 + 0) / 5 = 4.8$ msec

Assignment #2:

Implement the Non-preemptive SJF Algorithm in C++.