# REPORT ON

# SCHEDULING APPROACH

# (CSE316)

**@**


## LOVELY PROFESSIONAL UNIVERSITY,PHAGWARA

## JALANDHAR, PUNJAB (INDIA)

BY

S.Sai Surya Teja

#REE033A26 suryanaidu827@gmail.com

TO

Ms. Nahida Nazir

## PROBLEM

5. Consider a scheduling approach which is non pre-emptive similar to shortest job next in nature. The priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents

indefinite postponement. The jobs that have spent a long time waiting compete against those estimated to have short run times.

The priority can be computed as: = 1 +

Using the data given below compute the waiting time and turnaround time for each process and average ,waiting time and average turnaround time.

| process | Arrival time |
|---|---|
| P1 | 0 |
| P2 | 5 |
| P3 | 13 |
| P4 | 17 |

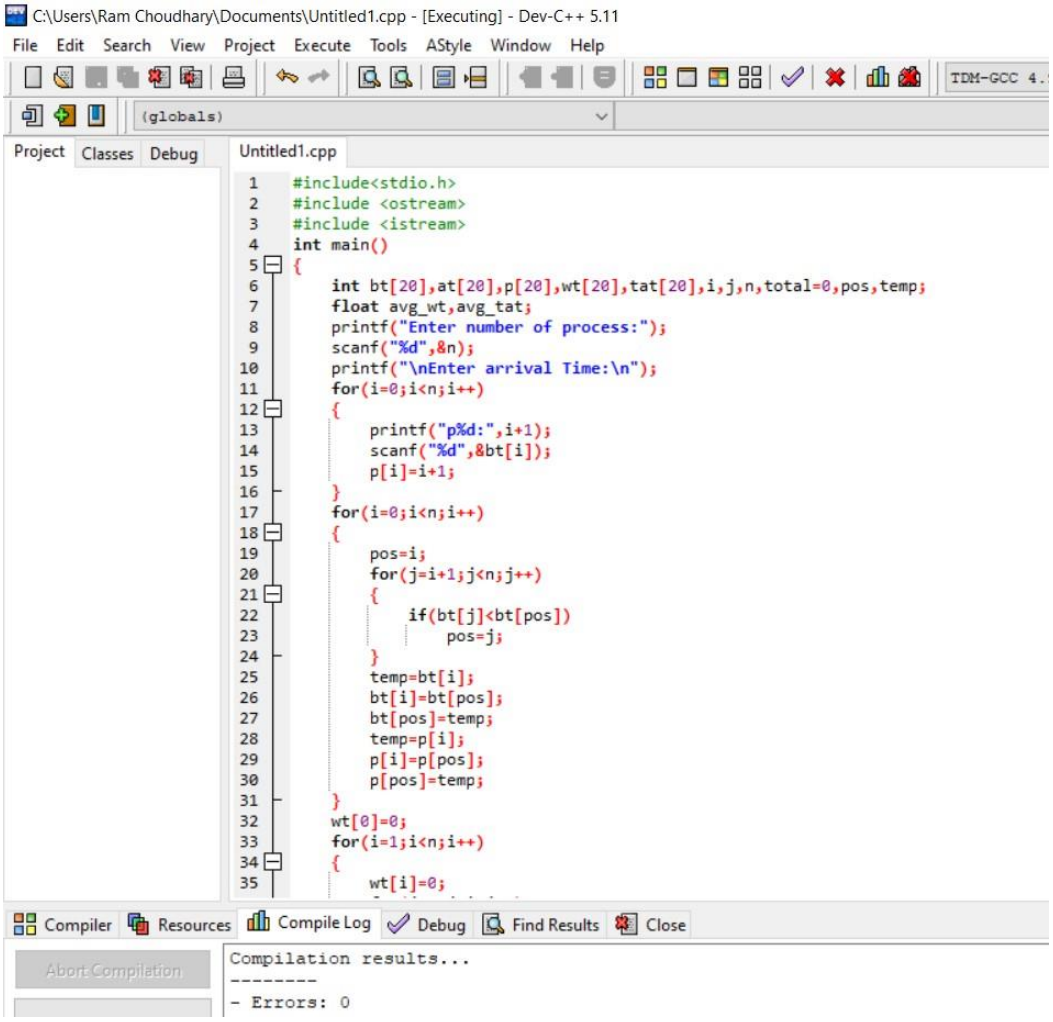
**SOLUTION CODE:**

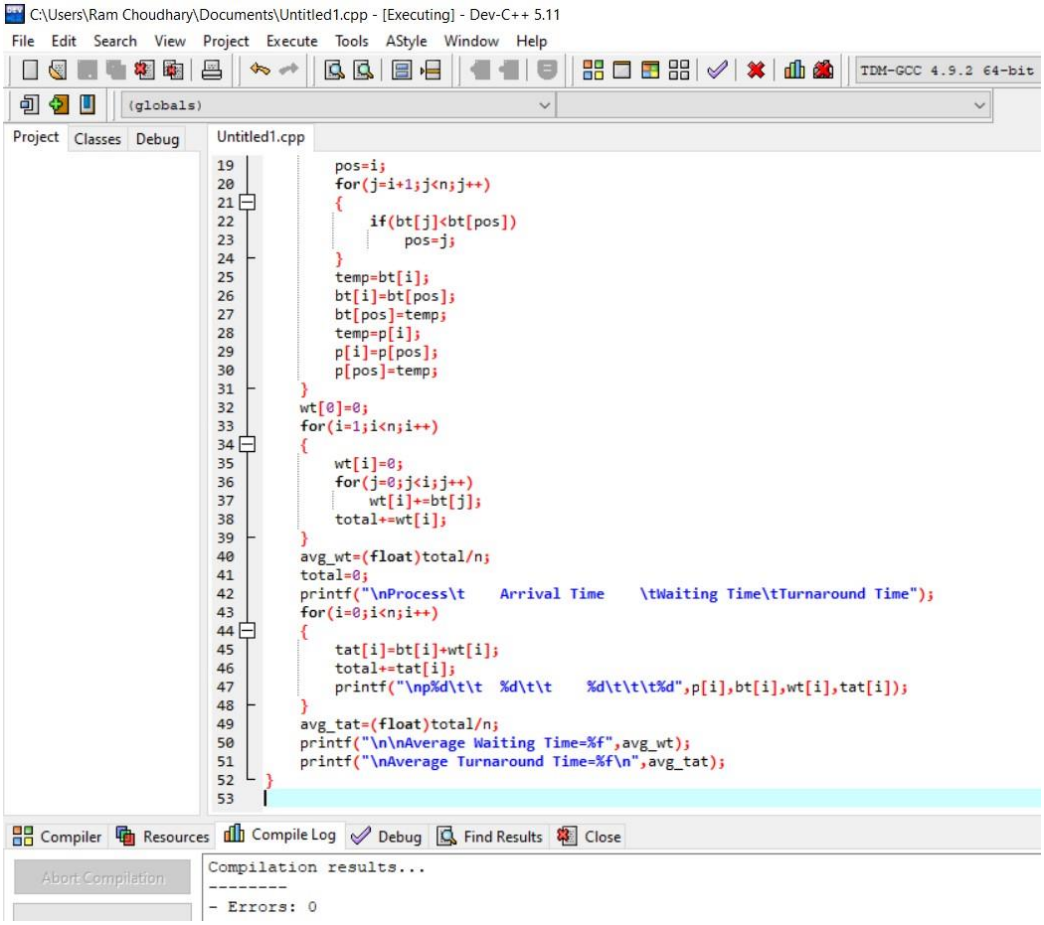
```
#include<stdio.h>
#include <ostream>
#include <istream>
int main() {
   int bt[20],at[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
float avg_wt,avg_tat;
   printf("Enter number of process:");
scanf("%d",&n);
   printf("\nEnter arrival Time:\n");
for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
scanf("%d",&bt[i]);        p[i]=i+1;
   }
   for(i=0;i<n;i++)
```

```
   {        pos=i;
for(j=i+1;j<n;j++)
       {
          if(bt[j]<bt[pos])
pos=j;
       }
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=(float)total/n;
total=0;
   printf("\nProcess\t    Arrival 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%d",p[i],bt[i],wt[i],tat[i]);
   }
   avg_tat=(float)total/n;
```

```
    printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

## SNAPSHOTS:






```
C:\Users\Ram Choudhary\Documents\Untitled1.exe
Enter number of process:4

Enter arrival Time:
p1:0
p2:5
p3:13
p4:17

Process     Arrival Time        Waiting Time    Turnaround Time
p1                0                   0                   0
p2                5                   0                   5
p3                13                  5                   18
p4                17                  18                  35

Average Waiting Time=5.750000
Average Turnaround Time=14.500000

--------------------------------
Process exited after 74.27 seconds with return value 0
Press any key to continue . . .
```

**Output:**

| process | arrival | waiting | **tat** |
|---|---|---|---|
| **1** | **0** | **0** | **0** |
| **2** | **5** | **0** | **5** |
| **3** | **13** | **5** | **18** |
| **4** | **17** | **18** | **35** |

## CODE ANALYSIS :

1)The turnaround time is calculated by adding the burst time and the waiting time.

```
for(i=0;i<n;i++)
{
  tat[i]=bt[i]+wt[i];
total+=tat[i];
  printf("np%d    %d    %d    %d",p[i],bt[i],wt[i],tat[i]);
}
```

2)The waiting time is calculated by adding the burst time to the waiting time.

```
for(i=1;i<n;i++)
 {      wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
 }
```

## DESCRIPTION:

- Shortest job next (SJN), is also known as shortest job first (SJF).
- **Shortest Job First (SJF)** is an algorithm in which the process having the smallest execution time is chosen for the next execution.
- This scheduling method can be preemptive or non-preemptive.

- It reduces the average amount of time each process has to wait until its execution is complete.
- **SJN** is frequently used for long term scheduling.
- It can improve process throughput by making sure that shorter jobs are executed first.
- This algorithm method is helpful for batch-type processing.
- Disadvantage of using shortest job next is that the total execution time of a job must be known before execution.

## Non Pre-emptive:

- Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state.
- In non-preemptive scheduling, once the CPU cycle is allocated to the process, the process holds it till it reaches a waiting state or terminated.
- Process can not be interrupted untill it terminates itself or its time is up.
- If a process with long burst time is running CPU, then later coming process with less CPU burst time may starve.
- It does not have overheads, it is rigid.

## Pre-emptive:

- Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state.
- The resources are allocated to the process for the limited amount of time and then taken away.
- The process is again placed back in the ready queue, if that process still has CPU burst time remaining.
- That process stays in ready queue till it gets next chance to execute.

### Advantages of shortest job next :

- SJN is frequently used for long term scheduling.
- It reduces the average waiting time
- SJN method gives the lowest average waiting time for a specific set of processes.
- It is appropriate for the jobs running in batch, where run times are known in advance.

- For the batch system of long-term scheduling, a burst time estimate can be obtained from the job description.
- For Short-Term Scheduling, we need to predict the value of the next burst time.Probably optimal with regard to average turnaround time.

## Dis- advantages:

- Job completion time must be known earlier, but it is hard to predict.
- It is often used in a batch system for long term scheduling.
- SJN can't be implemented for CPU scheduling for the short term. It is because there is no specific method to predict the length of the upcoming CPU burst.
- This algorithm may cause very long turnaround times or starvation.
- Requires knowledge of how long a process or job will run.
- It leads to the starvation that does not reduce average turnaround time.
- It is hard to know the length of the upcoming CPU request.
- Elapsed time should be recorded, that results in more overhead on the processor.

## GitHub Link :