SCHEDULING SIMULATOR

A
MINI PROJECT
OPERATING SYSTEMS
(CS1401)
IV SEMESTER

Submitted
By

Amar Sharma (179303017)

CCE-A

MANIPAL UNIVERSITY
JAIPUR

Department of Computer Science and Engineering
SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY
MANIPAL UNIVERSITY JAIPUR

2018-19

Problem Statement :

Create a simulator for scheduling a given set of processes in user space only. The simulator should read from a configuration file a set of parameters for each process: Length of time for which
process will execute, priority of the process and the preferred scheduling policy - FIFO or Round
Robin, the time at which the process executes and if it is a CPU intensive process _or an I/O
intensive process. Apart from this read the quantum of time given to each process and the
number of priority levels for scheduling the process. Now simulate a scheduling algorithm which
uses FIFO/Round Robin with priority based scheduling. At the end of the run print the following
quantities for each process:

a. Number of times the process was scheduled.
b. A timeline for the process containing the state transitions - Ready, waiting,
c. Running and Terminated and the timestamp for each transition.
d. Time taken to complete the process.
e. Number of times the process waited for I/O.
f. The priority of the process and preferred scheduling algorithm.

After printing the above values print the average time of completion for processes.

Program Code :

```
#include<stdio.h>
struct process
{
   char p_id[2];
   int BT;
   int Priority;
   int CPU_IO;
   int FCFS_RR;
   int TQ;
   int TAT;
   int WT;
};

int N=0,CPU=0,IO=0;
struct process p_CPU[10];
struct process p_IO[10];
void RR();
void FCFS();

int main()
{
   char buffer[200],details[10][12];
   FILE * filePointer;
   filePointer = fopen("Input.txt","r");
   if(filePointer == NULL)
   {
      printf("Failed to open a file.");
   }
   else
   {
      while(fgets(buffer,200,filePointer)!=NULL)
      {
         for(int i=0;i<12;i++)
            details[N][i]=buffer[i];
         N++;
      }
      fclose(filePointer);
```

```
    }
    printf("|-----------------------------------------------------------------|\n");
    printf("| P_id  |  BT  |  Priority  | CPU/IO  |  SCHEDULING  |  TQ |\n");
    printf("|-----------------------------------------------------------------|\n");
    for(int j=0 ;j<N;j++)
    {
       if(details[j][7]-'0' == 1)
       {
          p_CPU[CPU].p_id[0] = details[j][0];
          p_CPU[CPU].p_id[1] = (CPU+1)+'0';
          p_CPU[CPU].BT = details[j][3]-'0';
          p_CPU[CPU].Priority = details[j][5]-'0';
          p_CPU[CPU].CPU_IO = details[j][7]-'0';
          p_CPU[CPU].FCFS_RR = details[j][9]-'0';
          p_CPU[CPU].TQ = details[j][11]-'0';

                      printf("|  %c%c  |  %d  |  %d      |  %d    |  %d
|  %d  |\n",

        p_CPU[CPU].p_id[0],p_CPU[CPU].p_id[1],p_CPU[CPU].BT,p_CPU[C
PU].Priority,p_CPU[CPU].CPU_IO,p_CPU[CPU].FCFS_RR,p_CPU[CPU].T
Q);
          CPU++;
       }

       else
       {
          p_IO[IO].p_id[0] = details[j][0];
          p_IO[IO].p_id[1] = (IO+1)+'0';
          p_IO[IO].BT = details[j][3]-'0';
          p_IO[IO].Priority = details[j][5]-'0';
          p_IO[IO].CPU_IO = details[j][7]-'0';
          p_IO[IO].FCFS_RR = details[j][9]-'0';
          p_IO[IO].TQ = details[j][11]-'0';

          printf("|  %c%c  |  %d  |  %d      |  %d    |  %d        |  %d
|\n",

        p_IO[IO].p_id[0],p_IO[IO].p_id[1],p_IO[IO].BT,p_IO[IO].Priority,p_I
O[IO].CPU_IO,p_IO[IO].FCFS_RR,p_IO[IO].TQ);
```

```
            IO++;
        }


    }
    printf("|----------------------------------------------------------------|\n");

   if(p_CPU[0].FCFS_RR  == 0)
         FCFS();
         else
         RR();
  return 0;
}



void FCFS()
{
          printf("\n---------------------------------- FCFS -----------------------------------
---");
          int nwt[CPU];
    for(int i=0;i<CPU;i++)
    {
       nwt[i]=0;
    }

    int temp=0;
    for(int i=0;i<CPU;i++)
      {
       for(int j=i+1;j<CPU;j++)
        {
           if(p_CPU[i].Priority>p_CPU[j].Priority)
             {
                 temp=p_CPU[i].Priority;                              //sorting
according to Priority
                 p_CPU[i].Priority=p_CPU[j].Priority;
                 p_CPU[j].Priority=temp;

                 temp=p_CPU[i].p_id[1];
                 p_CPU[i].p_id[1]=p_CPU[j].p_id[1];
```

```
            p_CPU[j].p_id[1]=temp;

            temp=p_IO[i].p_id[1];
            p_IO[i].p_id[1]=p_IO[j].p_id[1];
            p_IO[j].p_id[1]=temp;

            temp=p_CPU[i].BT;
            p_CPU[i].BT=p_CPU[j].BT;
            p_CPU[j].BT=temp;

            temp=p_IO[i].BT;
            p_IO[i].BT=p_IO[j].BT;
            p_IO[j].BT=temp;
        }
    }
  }

  printf("\n GANTT  CHART \n");
  int j=0,time=0,IOBT=0,IOtime=0,flag=0;


  for(int i=0;i<CPU;i++)
  {

    for(int k=0 ; k<IO ; k++)
    {
        if(p_CPU[i].p_id[1] == p_IO[k].p_id[1] )
        {
                IOBT = p_IO[k].BT;
                break;
        }
    }
    if(IOBT)
    {
                            if(flag==0)
                            {
                printf(" %d-%d\t:  P%c\t ||  P%c : %d-%d (I/O)\n",

      time,time+p_CPU[i].BT,p_CPU[i].p_id[1],p_CPU[i].p_id[1],time+p_CP
U[i].BT,time+p_CPU[i].BT+IOBT);
```

```
                IOtime = time+p_CPU[i].BT+IOBT;
                time = time+p_CPU[i].BT;
                IOBT=0;
                flag=1;
        }
        else
        {
                if((time+p_CPU[i].BT)< IOtime)
                {
                                                        nwt[i]=1;
                                                        p_IO[i].WT =
IOtime - (time+p_CPU[i].BT);
                                                }
                                                else
                                                        IOtime =
time+p_CPU[i].BT;
                printf(" %d-%d\t:  P%c\t ||  P%c : %d-%d (I/O)\n",

        time,time+p_CPU[i].BT,p_CPU[i].p_id[1],p_CPU[i].p_id[1],IOtime,IOti
me+IOBT);
                                                time = time+p_CPU[i].BT;
                IOtime = IOtime+IOBT;
                IOBT=0;
        }
      }

      else
      {
                printf(" %d-%d\t:
P%c\n",time,time+p_CPU[i].BT,p_CPU[i].p_id[1]);
                                        time = time+p_CPU[i].BT;
      }
  }

       printf("\n\nNo. of times the process waited for (I/O) :\n");
       for(int i=0;i<IO;i++)
       {
         printf("P%c  :  %d\n",p_IO[i].p_id[1],nwt[i]);
       }
```

```
   for (int i=1; i<CPU ; i++)
   {
      p_CPU[i].WT =  p_CPU[i-1].BT + p_CPU[i-1].WT;
   }

   for (int i = 0; i <CPU; i++)
   {
      p_CPU[i].TAT = p_CPU[i].BT + p_CPU[i].WT;
   }
   for (int i = 0; i <IO; i++)
   {
      p_IO[i].TAT = p_IO[i].BT + p_IO[i].WT;
      p_CPU[i].TAT = p_CPU[i].TAT + p_IO[i].TAT;
   }
   int total_wt=0,total_tat=0;
   for (int i = 0; i < CPU; i++)
   {
      total_wt = total_wt + p_CPU[i].WT + p_IO[i].WT;
      total_tat = total_tat + p_CPU[i].TAT;
   }
   printf("\n\nWaiting Time and Turn-around time for each process :\n");
         printf("|-----------------------------------------------|\n");
   printf("| P_id\t|  Waiting Time\t|  Turnaround Time\t|\n");
   printf("|-----------------------------------------------|\n");
   for(int j=0 ;j<CPU;j++)
   {
      printf("|
P%c\t:\t%d\t:\t\t%d\t|\n",p_CPU[j].p_id[1],(p_CPU[j].WT+p_IO[j].WT),p_
CPU[j].TAT);
   }
   printf("|-----------------------------------------------|\n");
   printf("Total waiting time %d\n",total_wt);
   printf("Total turnaround time %d\n",total_tat);
   printf("Average waiting time = %f\n",(float)total_wt/(float)CPU);
   printf("Average turnaround time = %f\n\n",(float)total_tat/(float)CPU);
}


void RR()
{
```

```
        printf("\n---------------------------------- ROUND ROBIN ----------------------
-----------------");
   int ts = p_CPU[0].TQ,temp=0;
   for(int i=0;i<CPU;i++)
    {
      for(int j=i+1;j<CPU;j++)
       {
          if(p_CPU[i].Priority>p_CPU[j].Priority)
           {
               temp=p_CPU[i].Priority;                              //sorting
according to Priority
               p_CPU[i].Priority=p_CPU[j].Priority;
               p_CPU[j].Priority=temp;

               temp=p_CPU[i].p_id[1];
               p_CPU[i].p_id[1]=p_CPU[j].p_id[1];
               p_CPU[j].p_id[1]=temp;

               temp=p_IO[i].p_id[1];
               p_IO[i].p_id[1]=p_IO[j].p_id[1];
               p_IO[j].p_id[1]=temp;

               temp=p_CPU[i].BT;
               p_CPU[i].BT=p_CPU[j].BT;
               p_CPU[j].BT=temp;

               temp=p_IO[i].BT;
               p_IO[i].BT=p_IO[j].BT;
               p_IO[j].BT=temp;
           }
       }
    }
   int rt[CPU],cs[CPU],nwt[CPU];
   for(int i=0;i<CPU;i++)
   {
      rt[i]=p_CPU[i].BT;
      p_CPU[i].WT=0;
      cs[i]=0;
      nwt[i]=0;
```

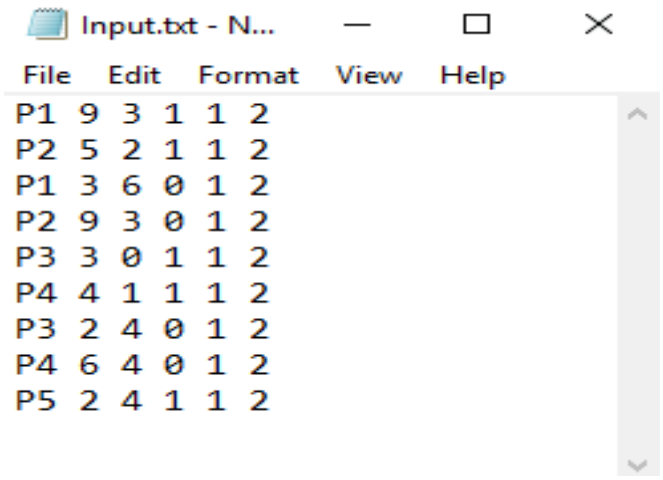
```
    }
    printf("\n GANTT  CHART \n");
    int j=0,time=0,IOBT=0,IOtime=0,flag=0;

    while(j<=CPU)
      {
        j++;
        for(int i=0;i<CPU;i++)
         {
            if(rt[i]==0)
               continue;
            if(rt[i]>ts)
            {
               printf(" %d-%d\t:  P%c\n",time,time+ts,p_CPU[i].p_id[1]);
               time=time+ts;
               rt[i]=rt[i]-ts;
               cs[i]=cs[i]+1;
            }
            else
            {
                 for(int k=0 ; k<IO ; k++)
                 {
                              if(p_CPU[i].p_id[1] == p_IO[k].p_id[1] )
                              {
                                      IOBT = p_IO[k].BT;
                                      break;
                              }
                 }
                 if(IOBT)
                 {
                                                      if(flag==0)
                                                      {
                              printf(" %d-%d\t:  P%c\t ||  P%c : %d-%d (I/O)\n",

        time,time+rt[i],p_CPU[i].p_id[1],p_CPU[i].p_id[1],time+rt[i],time+rt[i]
+IOBT);
                              IOtime = time+rt[i]+IOBT;
                              IOBT=0;
                              flag=1;
                      }
```

```
                else
                {
                        if((time+rt[i])< IOtime)
                        {
                                                                nwt[i]=1;
                                                                p_IO[i].WT =
IOtime - (time+p_CPU[i].BT);
                                                        }
                                                        else
                                                                IOtime =
time+rt[i];
                                printf(" %d-%d\t:  P%c\t ||  P%c : %d-%d (I/O)\n",

        time,time+rt[i],p_CPU[i].p_id[1],p_CPU[i].p_id[1],IOtime,IOtime+IOB
T);
                                IOtime = IOtime+IOBT;
                                IOBT=0;
                        }
                }
                else
                {
                        printf(" %d-%d\t:
P%c\n",time,time+rt[i],p_CPU[i].p_id[1]);
                }
                p_CPU[i].WT=time-cs[i]*ts;  //
                time=time+rt[i];
                rt[i]=0;
              }
           }
        }
   int wt=0,tat=0;

        printf("\nNo. of times the processes was scheduled :\n");
        for(int i=0;i<CPU;i++)
        {
          printf("P%c  :  %d\n",p_CPU[i].p_id[1],cs[i]+1);
        }

        printf("\nNo. of times the process waited for (I/O) :\n");
        for(int i=0;i<IO;i++)
```

```
        {
          printf("P%c  :  %d\n",p_IO[i].p_id[1],nwt[i]);
        }
        for(int i=0;i<IO;i++)
        {
          p_IO[i].TAT = p_IO[i].WT  + p_IO[i].BT;
        }

        printf("\nWaiting Time and Turn-around time for each process :\n");
        printf("|-----------------------------------------------|\n");
   printf("| P_id\t|  Waiting Time\t|  Turnaround Time\t|\n");
   printf("|-----------------------------------------------|\n");
   for(int i=0;i<CPU;i++)
     {
                p_CPU[i].TAT = p_CPU[i].WT+p_CPU[i].BT + p_IO[i].TAT;
                printf("|
P%c\t:\t%d\t:\t\t%d\t|\n",p_CPU[i].p_id[1],(p_CPU[i].WT+p_IO[i].WT),p_C
PU[i].TAT);
                wt  = wt + p_CPU[i].WT + p_IO[i].WT;
                tat = tat + p_CPU[i].TAT;
      }
   printf("|-----------------------------------------------|\n");

   printf("\n\nTotal waiting time %d\n",wt);
   printf("Avg wainting time %f\n",(float)wt/(float)CPU);
   printf("Total turnaround time %d\n",tat);
   printf("Avg turnaround time %f\n\n",(float)tat/(float)CPU);
}
```

Test Cases :

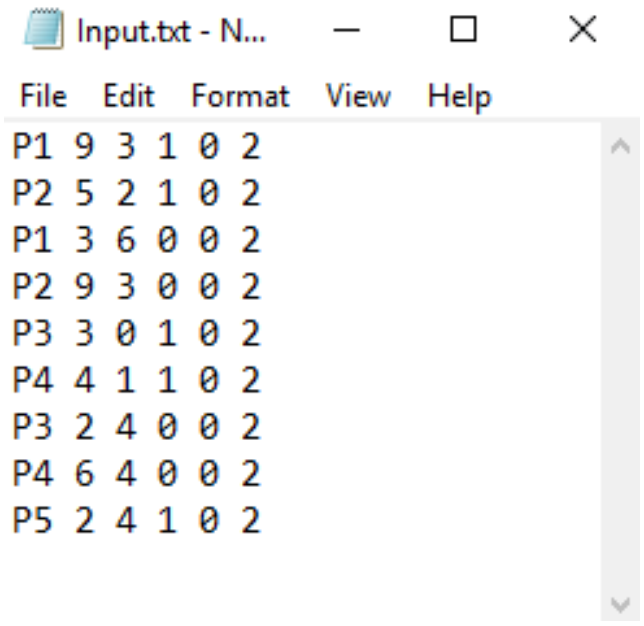
1. Input:



Output:

```
amar@amar:~$ gcc ss.c
amar@amar:~$ ./a.out
|-------------------------------------------------------------------|
| P_id   |   BT  |   Priority  |  CPU/IO  |   SCHEDULING   |   TQ  |
|-------------------------------------------------------------------|
|   P1   |   9   |   3         |   1      |   1            |   2   |
|   P2   |   5   |   2         |   1      |   1            |   2   |
|   P1   |   3   |   6         |   0      |   1            |   2   |
|   P2   |   9   |   3         |   0      |   1            |   2   |
|   P3   |   3   |   0         |   1      |   1            |   2   |
|   P4   |   4   |   1         |   1      |   1            |   2   |
|   P3   |   2   |   4         |   0      |   1            |   2   |
|   P4   |   6   |   4         |   0      |   1            |   2   |
|   P5   |   2   |   4         |   1      |   1            |   2   |
|-------------------------------------------------------------------|

---------------------------------- ROUND ROBIN ----------------------------------------
 GANTT  CHART
 0-2    :  P3
 2-4    :  P4
 4-6    :  P2
 6-8    :  P1
 8-10   :  P5
 10-11  :  P3    ||  P3 : 11-13 (I/O)
 11-13  :  P4    ||  P4 : 13-19 (I/O)
 13-15  :  P2
 15-17  :  P1
 17-18  :  P2    ||  P2 : 19-28 (I/O)
 18-20  :  P1
 20-22  :  P1
 22-23  :  P1    ||  P1 : 28-31 (I/O)

No. of times the processes was scheduled :
P3  :  2
P4  :  2
P2  :  3
P1  :  5
P5  :  1

No. of times the process waited for (I/O) :
P3  :  0
P4  :  0
P2  :  1
P1  :  1

Waiting Time and Turn-around time for each process :
|-------------------------------------------------|
| P_id  |  Waiting Time |  Turnaround Time        |
|-------------------------------------------------|
| P3    :       8       :                13       |
| P4    :       9       :                19       |
| P2    :       10      :                24       |
| P1    :       11      :                23       |
| P5    :       8       :                10       |
|-------------------------------------------------|


Total waiting time 46
Avg wainting time 9.200000
Total turnaround time 89
Avg turnaround time 17.799999

amar@amar:~$
```

2. Input:


Input.txt - N...
File Edit Format View Help
P1 9 3 1 0 2
P2 5 2 1 0 2
P1 3 6 0 0 2
P2 9 3 0 0 2
P3 3 0 1 0 2
P4 4 1 1 0 2
P3 2 4 0 0 2
P4 6 4 0 0 2
P5 2 4 1 0 2

Output:

```
amar@amar:~$ gcc ss.c
amar@amar:~$ ./a.out
|-------------------------------------------------------------------|
| P_id   |   BT  |   Priority  |  CPU/IO  |   SCHEDULING   |   TQ  |
|-------------------------------------------------------------------|
|   P1   |   9   |   3         |   1      |   0            |   2   |
|   P2   |   5   |   2         |   1      |   0            |   2   |
|   P1   |   3   |   6         |   0      |   0            |   2   |
|   P2   |   9   |   3         |   0      |   0            |   2   |
|   P3   |   3   |   0         |   1      |   0            |   2   |
|   P4   |   4   |   1         |   1      |   0            |   2   |
|   P3   |   2   |   4         |   0      |   0            |   2   |
|   P4   |   6   |   4         |   0      |   0            |   2   |
|   P5   |   2   |   4         |   1      |   0            |   2   |
|-------------------------------------------------------------------|

---------------------------------- FCFS ----------------------------------------
 GANTT  CHART
 0-3    :  P3    ||  P3 : 3-5 (I/O)
 3-7    :  P4    ||  P4 : 7-13 (I/O)
 7-12   :  P2    ||  P2 : 13-22 (I/O)
 12-21  :  P1    ||  P1 : 22-25 (I/O)
 21-23  :  P5


No. of times the process waited for (I/O) :
P3  :  0
P4  :  0
P2  :  1
P1  :  1


Waiting Time and Turn-around time for each process :
|-----------------------------------------------|
| P_id  |  Waiting Time |  Turnaround Time      |
|-----------------------------------------------|
| P3    :       0       :               5       |
| P4    :       3       :               13      |
| P2    :       8       :               22      |
| P1    :       13      :               25      |
| P5    :       21      :               23      |
|-----------------------------------------------|
Total waiting time 45
Total turnaround time 88
Average waiting time = 9.000000
Average turnaround time = 17.600000

amar@amar:~$
```