Laboratory 4

Title of the Laboratory Exercise: Programs for process scheduling algorithms

1. **Introduction and Purpose of Experiment**

   A Process Scheduler schedules different processes to CPU based on particular scheduling algorithms. There are various scheduling algorithms present in each group of operating system. By solving these problems students will be able use different scheduling algorithms as part of their implementation

2. **Aim and Objectives-**

   Aim

   - To develop programs to implement scheduling algorithms

   Objectives

   At the end of this lab, the student will be able to

   - Distinguish different scheduling algorithms
   - Apply the logic of scheduling algorithms wherever required
   - Create C programs to simulate scheduling algorithms

3. **Experimental Procedure**

   i. Analyse the problem statement

   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code

   iii. Implement the algorithm in C language

   iv. Compile the C program

   v. Test the implemented program

   vi. Document the Results

   vii. Analyse and discuss the outcomes of your experiment

4. Questions

Write a multithreaded program to simulate the following process scheduling algorithms. Calculate average waiting time and average turnaround time for processes under each scheduling algorithm by separate threads

Instructions: Assume all the processes arrive at the same time. For round robin scheduling algorithm, read the number of processes in the system, their CPU burst times and the size of the time slice. For priority scheduling algorithm, read the number of processes in the system, their CPU burst times and the priorities.

a) Priority
b) Round Robin

5. Calculations/Computations/Algorithms

Algorithm For Priority Scheduling

```
int main()
{int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
Input number of processes
   Display("\nEnter Burst Time and Priority\n");
   for(i=0;i<n;i++)
   {
      Display("\nP[%d]\n",i+1);
      Display("Burst Time:");
      Input(bt[i]);
      Display("Priority:");
      scanf("%d",&pr[i]);
      p[i]=i+1;
   }
   for(i=0;i<n;i++)
   {
      pos=i;
      for(j=i+1;j<n;j++)
      {
         if(pr[j]<pr[pos])
           assign  pos=j;
      }
      Assign temp=pr[i];
      Assign pr[i]=pr[pos];
      Assign pr[pos]=temp;

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

      assign total+=wt[i];
   }

 Assign avg_wt=total/n;
 Assign total=0;
 Display("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
 for(i=0;i<n;i++)
 {
   Assign   tat[i]=bt[i]+wt[i];
   Assign   total+=tat[i];
    Display("\nP[%d]\t\t  %d\t\t    %d\t\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
 }


     Assign  avg_tat=total/n;     //average turnaround time
    Display ("\n\nAverage Waiting Time=%d",avg_wt);
    display("\nAverage Turnaround Time=%d\n",avg_tat);
   return 0;
   }
```

Algorithm For Round Robin Algorithm

```
int main()
{
 int count,j,n,time,remain,flag=0,time_quantum;
 int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
 printf("Enter Total Process:\t ");
 scanf("%d",&n);
 remain=n;
 for(count=0;count<n;count++)
 {  Display("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
  Input(at[count]);
  Input(bt[count]);
 assign rt[count]=bt[count]; }
 Display("Enter Time Quantum:\t");
 Input(time_quantum);
 Display("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
 for(time=0,count=0;remain!=0;)
 {
  if(rt[count]<=time_quantum && rt[count]>0)
  {
   Assign time+=rt[count];
  Assign  rt[count]=0;
  Assign  flag=1;
  }
  else if(rt[count]>0)
  {
   Assign rt[count]-=time_quantum;
  Assign  time+=time_quantum;
  }
  if(rt[count]==0 && flag==1)
  {
   remain--;
  Display("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
  Assign  wait_time+=time-at[count]-bt[count];
 Assign  turnaround_time+=time-at[count];
  Assign  flag=0;
  }
  if(count==n-1)
  assign  count=0;
  else if(at[count+1]<=time)

        assign  count++;
        else
        assign  count=0;
       }
       Display("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
       Display("Avg Turnaround Time = %f",turnaround_time*1.0/n);
       return 0;
      }
```

6. Presentation of Results

Programs

a) Program for Priority Scheduling Algorithm

```c
//Program By Deepak R 18ETCS002041
#include<stdio.h>
int main()
{
    int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
    printf("Enter Total Number of Process:");
    scanf("%d",&n);

    printf("\nEnter Burst Time and Priority\n");
    for(i=0;i<n;i++)
    {
        printf("\nP[%d]\n",i+1);
        printf("Burst Time:");
        scanf("%d",&bt[i]);
        printf("Priority:");
        scanf("%d",&pr[i]);
        p[i]=i+1;           //contains process number
    }

    //sorting burst time, priority and process number in ascending order using selection sort
    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;    //waiting time for first process is zero

    //calculate waiting time
    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;      //average waiting time
    total=0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];     //calculate turnaround time
        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=total/n;     //average turnaround time
    printf("\n\nAverage Waiting Time=%d",avg_wt);
    printf("\nAverage Turnaround Time=%d\n",avg_tat);

    return 0;
}
```

b) Program for Round Robin Algorithm

```c
//Program By Deepak R 18ETCS002041
#include<stdio.h>
int main()
{
  int count,j,n,time,remain,flag=0,time_quantum;
  int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
  printf("Enter Total Process:\t ");
  scanf("%d",&n);
  remain=n;
  for(count=0;count<n;count++)
  {
    printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
    scanf("%d",&at[count]);
    scanf("%d",&bt[count]);
    rt[count]=bt[count];
  }
  printf("Enter Time Quantum:\t");
  scanf("%d",&time_quantum);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(time=0,count=0;remain!=0;)
  {
    if(rt[count]<=time_quantum && rt[count]>0)
    {
      time+=rt[count];
      rt[count]=0;
      flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-=time_quantum;
      time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
      wait_time+=time-at[count]-bt[count];
      turnaround_time+=time-at[count];
      flag=0;
    }
    if(count==n-1)
      count=0;
    else if(at[count+1]<=time)
      count++;
    else
      count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
}
```

Outputs

Test Cases for Priority Scheduling Algorithm

Test Case1 for Priority Scheduling Algorithm

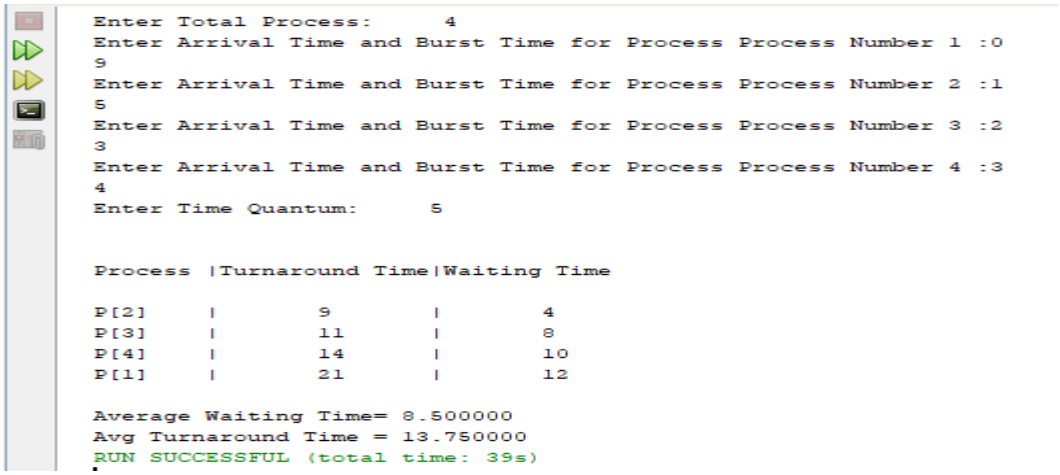
```
Enter Total Number of Process:4

Enter Burst Time and Priority

P[1]
Burst Time:6
Priority:3

P[2]
Burst Time:2
Priority:2

P[3]
Burst Time:14
Priority:1

P[4]
Burst Time:6
Priority:4

Process     Burst Time          Waiting Time    Turnaround Time
P[3]              14                0                 14
P[2]              2                 14                16
P[1]              6                 16                22
P[4]              6                 22                28

Average Waiting Time=13
Average Turnaround Time=20
```

Test Case 2 for Priority Scheduling Algorithm

```
Enter Total Number of Process:4

Enter Burst Time and Priority

P[1]
Burst Time:5
Priority:3

P[2]
Burst Time:3
Priority:2

P[3]
Burst Time:12
Priority:1

P[4]
Burst Time:5
Priority:4

Process     Burst Time          Waiting Time    Turnaround Time
P[3]              12                0                 12
P[2]              3                 12                15
P[1]              5                 15                20
P[4]              5                 20                25

Average Waiting Time=11
Average Turnaround Time=18

RUN SUCCESSFUL (total time: 43s)
```

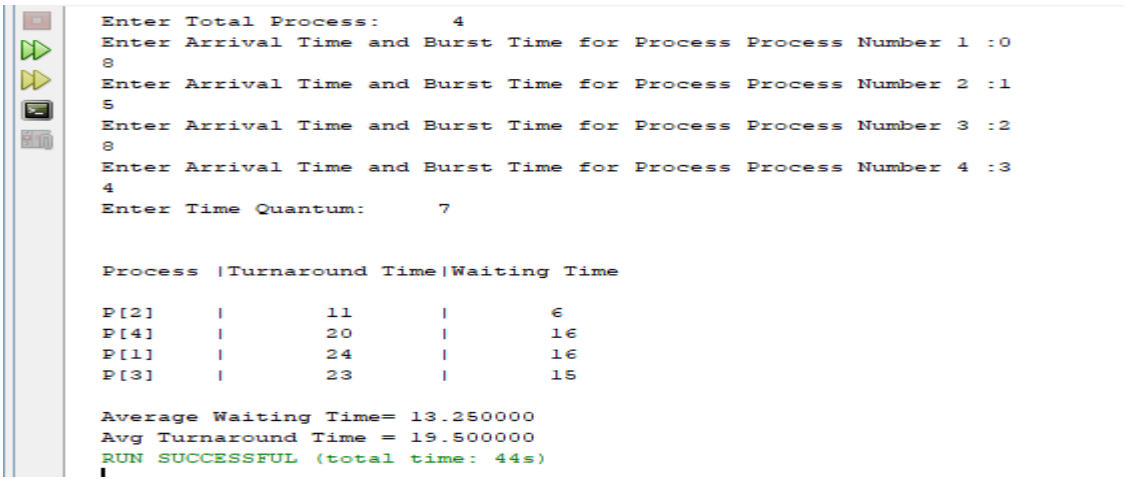
Test Cases for Round Robin Algorithm

Test Case 1 for Round Robin Algorithm



```
Enter Total Process:      4
Enter Arrival Time and Burst Time for Process Process Number 1 :0
9
Enter Arrival Time and Burst Time for Process Process Number 2 :1
5
Enter Arrival Time and Burst Time for Process Process Process Number 3 :2
3
Enter Arrival Time and Burst Time for Process Process Number 4 :3
4
Enter Time Quantum:      5


Process |Turnaround Time|Waiting Time

P[2]     |        9        |        4
P[3]     |        11       |        8
P[4]     |        14       |        10
P[1]     |        21       |        12

Average Waiting Time= 8.500000
Avg Turnaround Time = 13.750000
RUN SUCCESSFUL (total time: 39s)
```

Test Case 2 for Round Robin Algorithm



```
Enter Total Process:      4
Enter Arrival Time and Burst Time for Process Process Number 1 :0
8
Enter Arrival Time and Burst Time for Process Process Number 2 :1
5
Enter Arrival Time and Burst Time for Process Process Process Number 3 :2
8
Enter Arrival Time and Burst Time for Process Process Number 4 :3
4
Enter Time Quantum:      7


Process |Turnaround Time|Waiting Time

P[2]     |        11       |        6
P[4]     |        20       |        16
P[1]     |        24       |        16
P[3]     |        23       |        15

Average Waiting Time= 13.250000
Avg Turnaround Time = 19.500000
RUN SUCCESSFUL (total time: 44s)
```

7. Analysis and Discussions

The program calculates the Average Turn Around Times for the two algorithms namely Priority Scheduling and Round Robin Scheduling, where the CPU Burst times, Priorities and the Time Slice is taken as input from the user. The Respective Completion times are calculated and then the Turn Around Times is calculated for each of the process, the work is divided among two threads, which also calculate the average by dividing the sum of Turn Around Times by the total number of processes.

The main thread waits for the data returned by the two worker threads and then prints out the results obtained.

8. Conclusions

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.

- It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.
- One of the most commonly used technique in CPU scheduling as a core.
- It is preemptive as processes are assigned CPU only for a fixed slice of time at most.
- The disadvantage of it is more overhead of context switching.

Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.

- Each process is assigned a priority.
- Process with the highest priority is to be executed first and so on.
- Processes with the same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

9. Comments

1. Limitations of Experiments

The experiment assumes the arrival time for each of the process in the system to be same, this does not make the comparison between the two algorithms fair.

2. Limitations of Results

The result does not cover some of the important edge cases to consider for comparing the advantages and disadvantages of the two algorithms. Such as Priority Scheduling works well for Interactive Systems to provide the CPU to processes that need to be processed first.

3. Learning happened

We learnt the two commonly used scheduling algorithms i.e. Priority Scheduling and Round Robin Scheduling.

4. Recommendations

The program should be tested for more test cases such as when processes are generated randomly at different time intervals.