4B OS LAB-07 ASSIGNMENT

Muhammad Danish | cs182019 | 4B

Round Robin Scheduling

priorityScheduler.cpp | RR.cpp

```cpp
#include <iostream>
using namespace std;

// cs182019
// LAB 07
void findCompletionTime(int process[] , int n , int bt[],int at[], int ct[], int timeQ)
{
    // sorting the processes according to thier arrival time.
    // to find out one with the lowest arrival time.
    // this process will execute first.
    int temp=0;
    for(int i =0;i<n;i++)
    {
        for( int j=0;j<(n-i-1);j++)
        {
            if(at[j]>at[j+1])
            {
                temp=process[j];
                process[j]=process[j+1];
                process[j+1]=temp;

                temp=at[j];
                at[j]=at[j+1];
                at[j+1]=temp;

                temp=bt[j];
                bt[j]=bt[j+1];
                bt[j+1]=temp;
            }
        }
    }
    // array for copying the burst times
    int rem_bt_time[n];
    for(int i=0;i<n;i++)
    {
        // copying all the burst time for all the process.
        // before execution intial remaining time of each ana evert process will equals to its burst lenght.

            rem_bt_time[i]=bt[i];

```

priorityScheduler.cpp RR.cpp

```
31      }
32      // array for copying the burst times
33      int rem_bt_time[n];
34      for(int i=0;i<n;i++)
35      {
36          // copying all the burst time for all the process.
37          // before execution intial remaining time of each ana evert process will equals to its burst lenght.
38
39              rem_bt_time[i]=bt[i];
40
41              // makign every program completion time 0 at the start.
42              ct[i]=0;
43
44      }
45      int timeTaken=0;
46      int check=1;
47      int leaveTime[n];
48      temp = timeQ;
49
50      cout <<endl<< "Processes Execution Sequence : "<<endl;
51      while(1)
52      {
53          // true.
54          check=1;
55          for( int i=0;i<n;i++)
56          {
57              if(rem_bt_time[i]>0)
58              {
59                  timeQ = temp;
60                  // there is a pending process.
61                  check =0;
62                  // if burst time is greater than time quantum.
63                  // then
64                  // execute the process for time quantum.
65                  if(rem_bt_time[i]>=timeQ)
66                  {
67                      timeTaken= timeTaken+timeQ;
68                      leaveTime[i]=timeTaken;
69                      cout<<endl<<"Process    " << process[i]<<" Left at : "<<leaveTime[i];
70                      rem_bt_time[i]=rem_bt_time[i]-timeQ;
```

priorityScheduler.cpp RR.cpp

```
58                  {
59                      timeQ = temp;
60                      // there is a pending process.
61                      check =0;
62                      // if burst time is greater than time quantum.
63                      // then
64                      // execute the process for time quantum.
65                      if(rem_bt_time[i]>=timeQ)
66                      {
67                          timeTaken= timeTaken+timeQ;
68                          leaveTime[i]=timeTaken;
69                          cout<<endl<<"Process    " << process[i]<<" Left at : "<<leaveTime[i];
70                          rem_bt_time[i]=rem_bt_time[i]-timeQ;
71                          ct[i]=timeTaken;
72                      }
73                      else
74                      {
75                          timeQ=1;
76                          timeTaken=timeTaken+timeQ;
77                          leaveTime[i]=timeTaken;
78                          cout<<endl<<"Process    " << process[i]<<" Left at : "<<leaveTime[i];
79                          rem_bt_time[i]=0;
80                          ct[i]=timeTaken;
81                      }
82                  }
83              }
84                  if(check==1)
85                  {
86                      break;
87                  }
88          }
89      }
90
```

***FIND_COMPLETION_TIME FUCNTION END HERE.**

priorityScheduler.cpp [*] RR.cpp

```cpp
void findturnAroundTime (int processes[], int n , int at[], int tat[],int ct[])
{
    for(int i=0; i<n ;i++)
    {
        tat[i]=ct[i]-at[i];
    }
}

void findWatingTime(int processes[], int n, int wt[],int tat[], int bt[])
{
    for(int i=0;i<n;i++)
    {
        wt[i] =tat[i]-bt[i];
    }
}

void findAvgTime(int processes[],int n , int bt[], int at[], int timeQ)
{

    int wt[n], tat[n], ct[n],total_wt =0 , total_tat=0;
    findCompletionTime(processes,n,bt,at,ct,timeQ);
    findturnAroundTime(processes,n,at,tat,ct);
    findWatingTime(processes ,n , wt ,tat,bt);
    cout << "\n\nProcesses "<< " Arrival time "<<" Burst time " << " Completion time "<<" turn around time " <<" wating time\n ";
    for(int i=0; i<n;i++)
    {
        total_wt = total_wt + wt[i];
        total_tat =total_tat+tat[i];
        cout <<"  "<<i+1<<"\t\t"<<at[i] <<"\t   "<<bt[i]<<"\t\t " <<ct[i]<<"\t\t "<<tat[i]<<"\t\t   "<<wt[i]<< endl;
    }

    cout<< "average waiting time : " << (float)total_wt / (float)n <<endl ;
    cout<< "average tat time : " << (float)total_tat / (float)n;
}

    int main ()
    {
        int processes[]= {1,2,3,4};
        int timeQuantum=2;
        int n=4;

        int arrivalTime[]={0,2,5,6};
        int burstTime[]={6,1,4,3};

        findAvgTime(processes,n,burstTime,arrivalTime,timeQuantum);
        return 0;
    }
```

ROUND ROBIN SCHEDULING CODE OUTPUT

```
C:\Users\user\Desktop\Semester 4\OS Lab\Lab 07\Assg\cs182019\RR.exe

Processes Execution Sequence :

Process   1 Left at : 2
Process   2 Left at : 3
Process   3 Left at : 5
Process   4 Left at : 7
Process   1 Left at : 9
Process   3 Left at : 11
Process   4 Left at : 12
Process   1 Left at : 14

Processes  Arrival time  Burst time  Completion time  turn around time  wating time
   1            0          6             14               14               8
  2             2          1             3                1                0
  3             5          4             11               6                2
  4             6          3             12               6                3
average waiting time : 3.25
average tat time : 6.75
--------------------------------
Process exited after 0.1139 seconds with return value 0
Press any key to continue . . .
```

Priority Scheduling

priorityScheduler.cpp | RR.cpp

```cpp
#include <iostream>
using namespace std;

// cs182019
// LAB 07
void findCompletionTime(int process[] , int n ,int at[], int bt[],int ct[] , int p[])
{
    int temp=0;

    // arranging every process acording to least arrival times
    for(int i =0;i<n;i++)
    {
        for( int j=0;j<(n-i-1);j++)
        {
            if(at[j]>at[j+1])
            {
                temp=process[j];
                process[j]=process[j+1];
                process[j+1]=temp;

                temp=at[j];
                at[j]=at[j+1];
                at[j+1]=temp;

                temp=bt[j];
                bt[j]=bt[j+1];
                bt[j+1]=temp;

                temp=p[j];
                p[j]=p[j+1];
                p[j+1]=temp;
            }
        }
    }
    // now we know which process has lowest arrival time and it will execute first.

    // now we have to check which process has the highest priority.
```

Continued….

priorityScheduler.cpp RR.cpp

```
34      }
35      // now we know which process has lowest arrival time and it will execute first.
36
37      // now we have to check which process has the highest priority.
38      // for this, leaving the first process
39      // and arranging the remaining processes accoring to highest priority.
40      for(int i =1;i<n;i++)
41      {
42          for( int j=1;j<(n-i);j++)
43          {
44              if(p[j]>p[j+1])
45              {
46                  temp=process[j];
47                  process[j]=process[j+1];
48                  process[j+1]=temp;
49
50                  temp=at[j];
51                  at[j]=at[j+1];
52                  at[j+1]=temp;
53
54                  temp=bt[j];
55                  bt[j]=bt[j+1];
56                  bt[j+1]=temp;
57
58                  temp=p[j];
59                  p[j]=p[j+1];
60                  p[j+1]=temp;
61              }
62          }
63      }
64
65      // executing every process
66      ct[0]=bt[0];
67      for(int i =1;i<n ;i++)
68      {
69          ct[i]=ct[i-1]+bt[i];
70      }
71  }
72
```

priorityScheduler.cpp RR.cpp

```
73  void findturnAroundTime (int processes[], int n , int at[], int tat[], int ct[])
74  {
75      for(int i=0; i<n ;i++)
76      {
77          tat[i]=ct[i]-at[i];
78      }
79  }
80
81  void findWatingTime(int processes[], int n, int wt[], int tat[], int bt[])
82  {
83      wt[0]=0;
84
85      for(int i=1;i<n;i++)
86      {
87          wt[i] =tat[i]-bt[i];
88      }
89  }
90
```

```
void findAvgTime(int processes[],int n , int bt[], int at[], int p[])
{
    int wt[n], tat[n], ct[n],total_wt =0 , total_tat=0;
    findCompletionTime(processes,n,at,bt,ct,p);
    findturnAroundTime(processes,n,at,tat,ct);
    findWatingTime(processes ,n , wt ,tat,bt);
    cout << "\nOrder at which all the processes executed : ";
    for(int i=0;i<n;i++)
    {
        cout << processes[i] <<" " ;
    }
    cout<<"\n\n";
    cout << "Processes |"<< " Arrival time |"<<" Burst time |" << " Priority |" << "Completion time |"<<" turn around time |" <<" wating time\n ";
    for(int i=0; i<n;i++)
    {
        total_wt = total_wt + wt[i];
        total_tat =total_tat+tat[i];

        cout <<" "<<processes[i]<<"\t\t"<<at[i]<<"\t    "<<bt[i]<<"\t\t "<<p[i]<<"\t\t "<<ct[i]<<"\t\t "<<tat[i]<<"\t\t   "<<wt[i]<< endl;
    }
    cout<< "average waiting time : " << (float)total_wt / (float)n <<endl ;
    cout<< "average tat time : " << (float)total_tat / (float)n;
}

    int main ()
    {
        int processes[]= {1,2,3,4};
        int n=4;
        int burstTime[]={6,1,4,3};
        int arrivalTime[]={0,2,5,6};
        int priority[]={2,4,1,3};

        findAvgTime(processes,n,burstTime,arrivalTime, priority);
        return 0;
    }
```

PRIORITY SCHEDULING CODE OUTPUT

```
C:\Users\user\Desktop\Semester 4\OS Lab\Lab 07\Assg\cs182019\priorityScheduler.exe

Order at which all the processes executed : 1 3 4 2

Processes | Arrival time | Burst time | Priority |Completion time | turn around time | wating time
  1              0           6            2              6               6                0
 3               5           4            1              10              5                1
 4               6           3            3              13              7                4
 2               2           1            4              14              12               11
average waiting time : 4
average tat time : 7.5
--------------------------------
Process exited after 0.1064 seconds with return value 0
Press any key to continue . . .
```