EX.NO: 6 **CPU SCHEDULING ALGORITHMS**
Date : 09.09.2024

Round Robin ALGORITHM

AIM:

To write a C program to implement round robin scheduling ALGORITHM.

ALGORITHM:

1. Read no. of processes and time quantum (TQ).
2. Read process name and burst time (BT) for each process.
3. Ready queue is treated as circular queue. CPU schedules all processes (according to their of
   Order of arrival) only up to given time quantum.
4. A timer is set to interrupt the scheduling if time quantum expires for a process.
5. If BT of process is greater than TQ then after executing upto TQ, it gets added to tail of
   ready queue.
6. If BT of process is less than TQ then CPU gets released from it and schedules next process
   in ready queue.
7. Set waiting time (WT) of first process as zero and turnaround time (TAT) as burst time.
8. Calculate waiting time and turnaround time of other processes as follows:
   Pi (WT) = Pi-1(WT) +P i-1(BT)
   P i (TAT) = P i (BT) + P i (WT)
9. Calculate and display average WT and TAT.
10. Display order of execution of processes ie. Process name, burst time, WT and TAT.

PROGRAM:

```
#include<stdio.h>
int main()
{
   int  n;
   printf("Enter Total Number of Processes:");
   scanf("%d", &n);
   int wait_time = 0, ta_time = 0, arr_time[n], burst_time[n], temp_burst_time[n];
   int x = n;
   for(int i = 0; i < n; i++)
   {
      printf("Enter Details of Process %d \n", i + 1);
      printf("Arrival Time:  ");
      scanf("%d", &arr_time[i]);
      printf("Burst Time:   ");
      scanf("%d", &burst_time[i]);
      temp_burst_time[i] = burst_time[i];
   }
```

```
    int time_slot;
    printf("Enter Time Slot:");
    scanf("%d", &time_slot);
    int total = 0,  counter = 0,i;
    printf("Process ID       Burst Time       Turnaround Time      Waiting Time\n");
    for(total=0, i = 0; x!=0; )
    {
       if(temp_burst_time[i] <= time_slot && temp_burst_time[i] > 0)
       {
total = total + temp_burst_time[i];
          temp_burst_time[i] = 0;
          counter=1;
       }
       else if(temp_burst_time[i] > 0)
       {
          temp_burst_time[i] = temp_burst_time[i] - time_slot;
          total  += time_slot;
       }
       if(temp_burst_time[i]==0 && counter==1)
       {
          x--;
          printf("\nProcess No %d  \t\t %d\t\t\t\t %d\t\t\t %d", i+1, burst_time[i],
                total-arr_time[i], total-arr_time[i]-burst_time[i]);
          wait_time = wait_time+total-arr_time[i]-burst_time[i];
          ta_time += total -arr_time[i];
          counter =0;
       }
       if(i==n-1)
       {
          i=0;
       }
       else if(arr_time[i+1]<=total)
       {
          i++;
       }
       else
       {
          i=0;
       }
    }
    float average_wait_time = wait_time * 1.0 / n;
    float average_turnaround_time = ta_time * 1.0 / n;
    printf("\nAverage Waiting Time:%f", average_wait_time);
    printf("\nAvg Turnaround Time:%f", average_turnaround_time);
    return 0;
}
```

OUTPUT:

```
Enter Total Number of Processes:3
Enter Details of Process 1
Arrival Time:  0
Burst Time:   10
Enter Details of Process 2
Arrival Time:  1
Burst Time:   8
Enter Details of Process 3
Arrival Time:  2
Burst Time:   7
Enter Time Slot:5

Process ID      Burst Time      Turnaround Time      Waiting Time

Process No 1      10              20              10
Process No 2      8               22                      14
Process No 3      7               23                      16

Average Waiting Time: 13.333333
Avg Turnaround Time: 21.666666
```

EX.NO: 6(b) **Shortest Job First Algorithm**

AIM:

To write a C program to implement SJF (Shortest Job First) scheduling ALGORITHM.

ALGORITHM:

Step1: Read no. of processes.
Step2: Read process name and burst time for each process.
Step3: Sort the processes in ready queue according to burst time. CPU schedules
process with shortest burst time first followed by other processes.
Step4: Set waiting time(WT) of first process as zero and turnaround time(TAT) as
burst time.
Step5: Calculate waiting time and turnaround time of other processes as follows:
Pi (WT) = P i-1(WT) +P i-1(BT)
P i (TAT) = P i (BT) + P i (WT)
Step6: Calculate and display average WT and TAT.
Step7: Display order of execution of processes ie. Process name, burst time, WT and
TAT.

PROGRAM:

```
#include<stdio.h>
int main()
{
   int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,totalT=0,pos,temp;
   float avg_wt,avg_tat;
   printf("Enter number of process:");
   scanf("%d",&n);

   printf("\nEnter Burst Time:\n");
   for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;
   }

   //sorting of burst times
   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;

    //finding the waiting time of all the processes
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            //individual WT by adding BT of all previous completed processes
            wt[i]+=bt[j];

        //total waiting time
        total+=wt[i];
    }

    //average waiting time
    avg_wt=(float)total/n;

    printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        //turnaround time of individual processes
        tat[i]=bt[i]+wt[i];

        //total turnaround time
        totalT+=tat[i];
        printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }
     //average turnaround time
    avg_tat=(float)totalT/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f",avg_tat);
}
```

OUTPUT:

Enter number of process:4

Enter Burst Time:
p1:5
p2:4
p3:12
p4:7

Process	Burst Time	Waiting Time	Turnaround Time
p2	4	0	4
p1	5	4	9
p4	7	9	16
p3	12	16	28

Average Waiting Time=7.250000
Average Turnaround Time=14.250000

EX.NO: 6(C) **First Come First Serve ALGORITHM (FCFS)**

AIM:

To write a C program to implement FCFS scheduling ALGORITHM.

ALGORITHM:

Step1: Read no. of processes.
Step2: Read process name and burst time for each process.
Step3: CPU schedules processes according to their order of arrival in read queue (ie It first
executes process which is at head of ready queue)
Step4: Set waiting time(WT) of first process as zero and turnaround time(TAT) as burst time.
Step5: Calculate waiting time and turnaround time of other processes as follows:
Pi (WT) = P i-1(WT) +P i-1(BT)
P i (TAT) = P i (BT) + P i (WT)
Step6: Calculate and display average WT and TAT.
Step7: Display order of execution of processes ie. Process name, burst time, WT and TAT.

PROGRAM:

```
#include <stdio.h>
int main()
{
   int pid[15];
   int bt[15];
   int n;
   printf("Enter the number of processes: ");
   scanf("%d",&n);

   printf("Enter process id of all the processes: ");
   for(int i=0;i<n;i++)
   {
      scanf("%d",&pid[i]);
   }

   printf("Enter burst time of all the processes: ");
   for(int i=0;i<n;i++)
   {
      scanf("%d",&bt[i]);
   }
    int i, wt[n];
   wt[0]=0;
    //for calculating waiting time of each process
   for(i=1; i<n; i++)
   {
      wt[i]= bt[i-1]+ wt[i-1];
```

```
    }

    printf("Process ID    Burst Time    Waiting Time    TurnAround Time\n");
    float twt=0.0;
    float tat= 0.0;
for(i=0; i<n; i++)
    {
        printf("%d\t\t", pid[i]);
        printf("%d\t\t", bt[i]);
        printf("%d\t\t", wt[i]);

        //calculating and printing turnaround time of each process
        printf("%d\t\t", bt[i]+wt[i]);
        printf("\n");

        //for calculating total waiting time
        twt += wt[i];

        //for calculating total turnaround time
        tat += (wt[i]+bt[i]);
    }
    float att,awt;

    //for calculating average waiting time
    awt = twt/n;

    //for calculating average turnaround time
    att = tat/n;
    printf("Avg. waiting time= %f\n",awt);
    printf("Avg. turnaround time= %f",att);
}
```

OUTPUT:

```
Enter the number of processes: 3
Enter process id of all the processes: 1 2 3
Enter burst time of all the processes: 5 11 11
Process ID    Burst Time    Waiting Time    TurnAround Time
1             5             0             5
2             11            5             16
3             11            16            27
Avg. waiting time= 7.000000
Avg. turnaround time= 16.000000
```

EX.NO: 6 (D) **Priority Scheduling ALGORITHM**

AIM:

To write a C program to implement priority scheduling ALGORITHM.

ALGORITHM:

Step1: Read no. of processes.
Step2: Read process name, burst time and priority for each process.
Step3: Sort the processes in ready queue according to priority. (i.e. Process with high priority get placed at head of ready queue) CPU schedules process with high priority first followed by other processes.
Step4: Set waiting time (WT) of first process as zero and turnaround time (TAT) as burst time.
Step5: Calculate waiting time and turnaround time of other processes as follows:
Pi (WT) = P i-1(WT) +P i-1(BT)
P i (TAT) = P i (BT) + P i (WT)
Step6: Calculate and display average WT and TAT.
Step7: Display order of execution of processes ie. Process name, burst time, priority, WT and
TAT.
Step8: Stop the program.

PROGRAM:

```
#include <stdio.h>
//Function to swap two variables
void swap(int *a,int *b)
{
   int temp=*a;
   *a=*b;
   *b=temp;
}
int main()
{
   int n;
   printf("Enter Number of Processes: ");
   scanf("%d",&n);
   int b[n],p[n],index[n];
   for(int i=0;i<n;i++)
   {
      printf("Enter Burst Time and Priority Value for Process %d: ",i+1);
      scanf("%d %d",&b[i],&p[i]);
     index[i]=i+1;
   }
   for(int i=0;i<n;i++)
   {
      int a=p[i],m=i;
      for(int j=i;j<n;j++)
```

```
        {
            if(p[j] > a)
            {
                a=p[j];
                m=j;
            }
        }
        swap(&p[i], &p[m]);
        swap(&b[i], &b[m]);
        swap(&index[i],&index[m]);
    }
    int t=0;
    printf("Order of process Execution is\n");
    for(int i=0;i<n;i++)
    {
        printf("P%d is executed from %d to %d\n",index[i],t,t+b[i]);
        t+=b[i];
    }
    printf("\n");
    printf("Process Id     Burst Time   Wait Time    TurnAround Time\n");
    int wait_time=0;
    for(int i=0;i<n;i++)
    {
        printf("P%d         %d          %d
%d\n",index[i],b[i],wait_time,wait_time + b[i]);
        wait_time += b[i];
    }
    return 0;
}
```

OUTPUT:

```
Enter Number of Processes: 3
Enter Burst Time and Priority Value for Process 1: 10 2
Enter Burst Time and Priority Value for Process 2: 5 0
Enter Burst Time and Priority Value for Process 3: 8 1
Order of process Execution is
P1 is executed from 0 to 10
P3 is executed from 10 to 18
P2 is executed from 18 to 23

Process Id     Burst Time   Wait Time    TurnAround Time
   P1          10           0            10
   P3          8            10           18
   P2          5            18           23
```

RESULT:

Thus the Process Scheduling algorithms programs were executed successfully.