Ex. No. : 9 **Date : 21/10/2022**

SIMULATION OF CPU SCHEDULING ALGORITHMS

AIM :

To implement the process scheduling using First Come, First Served (FCFS), Round Robin (RR), Shortest Job First (SJB) and Priority Scheduling Algorithms.

FIRST COME FIRST SERVE (FCFS) :

PROGRAM :

```
#include<stdio.h>

int main()

{
	int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
	printf("Enter total number of processes(maximum 20):");
	scanf("%d",&n);

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

	wt[0]=0;

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

	printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
	for(i=0;i<n;i++)
	{
```

```
            tat[i]=bt[i]+wt[i];//Turnaround Time = Burst time-waiting Time
            avwt+=wt[i];
            avtat+=tat[i];
            printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
        }

        avwt/=i;
        avtat/=i;
        printf("\n\nAverage Waiting Time:%d",avwt);
        printf("\nAverage Turnaround Time:%d",avtat);

        return 0;
}
```

OUTPUT :

```
ubuntu@ubuntu:~/2021242013$ gcc fcfs.c -o fcfs
ubuntu@ubuntu:~/2021242013$ ./fcfs
Enter total number of processes(maximum 20):4

Enter Process Burst Time
P[1]:10
P[2]:8
P[3]:4
P[4]:5

Process         Burst Time      Waiting Time    Turnaround Time
P[1]            10              0               10
P[2]            8               10              18
P[3]            4               18              22
P[4]            5               22              27

Average Waiting Time:12
Average Turnaround Time:19ubuntu@ubuntu:~/2021242013$
```

ROUND ROBIN :

PROGRAM :

```
#include<stdio.h>
int main()
{
    int i, limit, total = 0, x, counter = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10];
```

```
float average_wait_time, average_turnaround_time;
printf("\nEnter Total Number of Processes: ");
scanf("%d", &limit);
x = limit;
for(i = 0; i < limit; i++)
{
     printf("\nEnter Details of Process[%d]", i + 1);
     printf("\nArrival Time:");
     scanf("%d", &arrival_time[i]);
    printf("\nBurst Time:");
    scanf("%d", &burst_time[i]);
    temp[i] = burst_time[i];
}

printf("\nEnter Time Quantum: ");
scanf("%d", &time_quantum);
printf("\nProcess ID\t\tBurst Time\t Turnaround Time\tWaiting Time: ");
for(total = 0, i = 0; x != 0;)
{
     if(temp[i] <= time_quantum && temp[i] > 0)
     {
          total = total + temp[i];
          temp[i] = 0;
          counter = 1;
     }
     else if(temp[i] > 0)
     {
          temp[i] = temp[i] - time_quantum;
          total = total + time_quantum;
     }
     if(temp[i] == 0 && counter == 1)
     {
          x--;
          printf("\n\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst_time[i], total -
                                        arrival_time[i], total - arrival_time[i] - burst_time[i]);
          wait_time = wait_time + total - arrival_time[i] - burst_time[i];
          turnaround_time = turnaround_time + total - arrival_time[i];
          counter = 0;
     }
     if(i == limit - 1)
     {
          i = 0;
     }
     else if(arrival_time[i + 1] <= total)
     {
          i++;
```

```
        }
        else
        {
            i = 0;
        }
    }

    average_wait_time = wait_time * 1.0 / limit;
    average_turnaround_time = turnaround_time * 1.0 / limit;
    printf("\nAverage Waiting Time:\t%f", average_wait_time);
    printf("\n\nAverage Turnaround Time:\t%f\n", average_turnaround_time);
    return 0;
}
```

OUTPUT :

```
ubuntu@ubuntu:~/2021242013$ gcc rorob.c -o rorob
ubuntu@ubuntu:~/2021242013$ ./rorob

Enter Total Number of Processes: 4

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:1
Burst Time:4

Enter Details of Process[4]
Arrival Time:2
Burst Time:5

Enter Time Quantum: 3

Process ID              Burst Time       Turnaround Time        Waiting Time:
Process[3]              4                18                      14
Process[4]              5                19                      14
Process[2]              8                25                      17
Process[1]              10               27                      17

Average Waiting Time    :       15.500000
Average Turnaround Time :       22.250000
ubuntu@ubuntu:~/2021242013$
```

SHORTEST JOB FIRST (SJB) :

There are two types in SJF:

- **Non preemptive**
- **Preemptive**

NON-PREEMPTIVE :

PROGRAM :

```
#include <stdio.h>
int main()
{
    int A[100][4]; // Matrix for storing Process Id, Burst
                   // Time, Average Waiting Time & Average
                   // Turn Around Time.
    int i, j, n, total = 0, index, temp;
    float avg_wt, avg_tat;
    printf("Enter number of process: ");
    scanf("%d", &n);
    printf("Enter Burst Time:\n");
    // User Input Burst Time and alloting Process Id.
    for (i = 0; i < n; i++) {
        printf("P%d: ", i + 1);
        scanf("%d", &A[i][1]);
        A[i][0] = i + 1;
    }
    // Sorting process according to their Burst Time.
    for (i = 0; i < n; i++) {
        index = i;
        for (j = i + 1; j < n; j++)
```

```
        if (A[j][1] < A[index][1])
            index = j;
    temp = A[i][1];
    A[i][1] = A[index][1];
    A[index][1] = temp;

    temp = A[i][0];
    A[i][0] = A[index][0];
    A[index][0] = temp;
}
A[0][2] = 0;
// Calculation of Waiting Times
for (i = 1; i < n; i++) {
    A[i][2] = 0;
    for (j = 0; j < i; j++)
        A[i][2] += A[j][1];
    total += A[i][2];
}
avg_wt = (float)total / n;
total = 0;
printf("P     BT     WT     TAT\n");
// Calculation of Turn Around Time and printing the
// data.
for (i = 0; i < n; i++) {
    A[i][3] = A[i][1] + A[i][2];
    total += A[i][3];
    printf("P%d     %d     %d      %d\n", A[i][0],
            A[i][1], A[i][2], A[i][3]);
}
avg_tat = (float)total / n;
```

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

OUTPUT :

```
ubuntu@ubuntu:~/2021242013$ gcc sjb.c -o sjb
ubuntu@ubuntu:~/2021242013$ ./sjb
Enter number of process: 4

Enter Burst Time:
P1: 10
P2: 8
P3: 4
P4: 5
Process BurstingTime    WaitingTime     TurnaroundTime
P3              4               0               4
P4              5               4               9
P2              8               9               17
P1              10              17              27


Average Waiting Time    : 7.500000
Average Turnaround Time : 14.250000ubuntu@ubuntu:~/2021242013$
```

PREEMPTIVE

PROGRAM:

```
#include <stdio.h>


int main()

{

    int arrival_time[10], burst_time[10], temp[10];
    int i, smallest, count = 0, time, limit;
    double wait_time = 0, turnaround_time = 0, end;
```

```
float average_waiting_time, average_turnaround_time;
printf("\nEnter the Total Number of Processes:"); scanf("%d",
&limit);
printf("\nEnter Details of %d Processes\n", limit);

for(i = 0; i < limit; i++)
{
        printf("\nEnter Arrival Time:"); scanf("%d", &arrival_time[i]);
        printf("\nEnter Burst Time:"); scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
}
burst_time[9] = 9999;

for(time = 0; count != limit; time++)
{
        smallest = 9;
                for(i = 0; i < limit; i++)
        {
                if(arrival_time[i] <= time && burst_time[i] <
                burst_time[smallest] && burst_time[i] > 0)
                {
                        smallest = i;
                }
        }
        burst_time[smallest]--;

        if(burst_time[smallest] == 0)
        {
                count++;
                end = time + 1;
                wait_time = wait_time + end - arrival_time[smallest] -
                temp[smallest]; turnaround_time = turnaround_time +
                end - arrival_time[smallest];
        }
 }
```

```
        average_waiting_time = wait_time / limit; average_turnaround_time
        = turnaround_time / limit; printf("\n\nAverage Waiting Time:%lf\n",
        average_waiting_time);
        printf("Average Turnaround Time:t%lf\n",
        average_turnaround_time); return 0;
}
```

OUTPUT:

```
metta@metta-Latitude-E5440:~/2021242013$ nano sjfpreem
metta@metta-Latitude-E5440:~/2021242013$ gcc sjfpreem.c -o sjfpreem
metta@metta-Latitude-E5440:~/2021242013$ ./sjfpreem

Enter the Total Number of Processes:4

Enter Details of 4 Processes

Enter Arrival Time:0

Enter Burst Time:8

Enter Arrival Time:1

Enter Burst Time:3

Enter Arrival Time:1

Enter Burst Time:5

Enter Arrival Time:2

Enter Burst Time:5


Average Waiting Time:5.750000
Average Turnaround Time:t11.000000
```

PRIORITY SCHEDULING :

There are two types of priority scheduling.

- **Non preemptive**
- **Preemptive**

NON PREEMPTIVE:

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);

    // b is array for burst time, p for priority and index for process id
    int b[n],p[n],index[n];
    float avg_wt = 0, avg_tat = 0;
    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;

    //Finding out highest priority element and placing it at its desired position
    for(int j=i;j<n;j++)
    {
        if(p[j] > a)
        {
            a=p[j];
            m=j;
        }
    }

    //Swapping processes
    swap(&p[i], &p[m]);
    swap(&b[i], &b[m]);
    swap(&index[i],&index[m]);
}

// T stores the starting time of process
int t=0;

//Printing scheduled process
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]);
        avg_wt += wait_time;
        wait_time += b[i];
        avg_tat += wait_time + b[i];
    }
     avg_wt /= n;
     avg_tat /= n;
     printf("\n\nAverage Waiting Time   : %f ", avg_wt);
     printf("\nAvergae Turnaround Time : %f ", avg_tat);
     return 0;
}
```

OUTPUT :

```
ubuntu@ubuntu:~/2021242013$ gcc priority.c -o priority
ubuntu@ubuntu:~/2021242013$ ./priority
Enter Number of Processes: 4

Enter details of Process 1 :
Enter Burst Time  : 10
Enter Priority    : 3

Enter details of Process 2 :
Enter Burst Time  : 8
Enter Priority    : 2

Enter details of Process 3 :
Enter Burst Time  : 4
Enter Priority    : 4

Enter details of Process 4 :
Enter Burst Time  : 5
Enter Priority    : 1

Order of process Execution is
P3 is executed from 0 to 4
P1 is executed from 4 to 14
P2 is executed from 14 to 22
P4 is executed from 22 to 27


Process Id      Burst Time      Wait Time       TurnAround Time
P3              4               0               4
P1              10              4               14
P2              8               14              22
P4              5               22              27


Average Waiting Time   : 10.000000
Average Turnaround Time : 23.500000 ubuntu@ubuntu:~/2021242013$
```

PREEMPTIVE

PROGRAM:

```
#include<stdio.h>
int main()
{
        int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,burst=0,pri[10];
        int bt[10],temp,temp1,j,at[10],wt[10],rt[10],tt[10],ta=0,sum=0;
        float wavg,tavg,tsum,wsum;
        printf("\nEnter the No. processes ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
```

```
{
        printf("\nEnter the burst time of %d process : ",i+1);
        scanf("%d",&bt[i]);
        printf("Enter the arrival time of %d process : ",i+1);
        scanf("%d",&at[i]);
        printf("Enter the priority time of %d process : ",i+1);
        scanf("%d",&pri[i]);
        printf("\n");
}

for(i=0;i<n;i++)
{
        for(j=0;j<n;j++)
        {
                if(at[i]<at[j])
                {
                temp=p[j];
                p[j]=p[i];
                p[i]=temp;
                temp=at[j];
                at[j]=at[i];
                at[i]=temp;
                temp1=bt[j];
                bt[j]=bt[i];
                bt[i]=temp1;
                }
        }
}

for(j=0;j<n;j++)
{
        burst=burst+bt[j]; min=bt[k];
        for(i=k;i<n;i++)/*main logic*/
        {
                min=pri[k];
                if (burst>=at[i])
                {
```

```
                    if(pri[i]<min)
                    {
                    temp=p[k];
                    p[k]=p[i];
                    p[i]=temp;
                    temp=at[k];
                    at[k]=at[i];
                    at[i]=temp;
                    temp1=bt[k];
                    bt[k]=bt[i];
                    bt[i]=temp1;
                    temp=pri[k];
                    pri[k]=pri[i];
                    pri[i]=temp;

                    }
                }
        } k++;
}
wt[0]=0;
for(i=1;i<n;i++)
{
        sum=sum+bt[i-1];
        wt[i]=sum-at[i];
}

for(i=0;i<n;i++)
{
        wsum=wsum+wt[i];
}
wavg=wsum/n;

for(i=0;i<n;i++)
{
        ta=ta+bt[i];
        tt[i]=ta-at[i];
}
```

```
        for(i=0;i<n;i++)
        {
                tsum=tsum+tt[i];
        }
        tavg=tsum/n;
        for(i=0;i<n;i++)
        {
                rt[i]=wt[i];
        }
        printf("\nprocess\t burst\t arrival\tpriority " );
        for(i=0;i<n;i++)
        {
                printf("\n p%d",p[i]);
                printf("\t %d",bt[i]);
                printf("\t %d",at[i]);
                printf("\t\t %d",pri[i]);
        }
        printf("\nwaiting time\tturnaround time\tresponce time");

        for(i=0;i<n;i++)
        {
                printf("\n %d",wt[i]);
                printf("\t\t %d",tt[i]);
                printf("\t\t%d",rt[i]);
        }
        printf("\nAVERAGE WAITING TIME:- %f ms",wavg);
        printf("\nAVERAGE TURN AROUND TIME:- %f ms" ,tavg);
        printf("\nAVERAGE RESPONSE TIME:- %f ms\n",wavg);
}
```

OUTPUT:

```
metta@metta-Latitude-E5440:~/2021242013$ nano prioritypreem
metta@metta-Latitude-E5440:~/2021242013$ gcc prioritypreem.c -o prioritypreem
metta@metta-Latitude-E5440:~/2021242013$ ./prioritypreem

Enter the No. processes 4

Enter the burst time of 1 process : 2
Enter the arrival time of 1 process : 3
Enter the priority time of 1 process : 4


Enter the burst time of 2 process : 1
Enter the arrival time of 2 process : 2
Enter the priority time of 2 process : 3


Enter the burst time of 3 process : 4
Enter the arrival time of 3 process : 5
Enter the priority time of 3 process : 6


Enter the burst time of 4 process : 8
Enter the arrival time of 4 process : 9
Enter the priority time of 4 process : 7


process  burst   arrival        priority
 p2      1       2               4
 p1      2       3               3
 p3      4       5               6
 p4      8       9               7
waiting time    turnaround time responce time
 0               -1             0
 -2              0              -2
 -2              2              -2
 -2              6              -2
AVERAGE WAITING TIME:- -1.500000 ms
AVERAGE TURN AROUND TIME:- 1.750000 ms
AVERAGE RESPONSE TIME:- -1.500000 ms
```

RESULT :

Thus, the process scheduling were implemented using FCFS, Round Robin, SJB and Priority Scheduling Algorithms.