EX:NO:9 Priority and Round Robin Scheduling

Aim :

To study the concepts of Priority and Round Robin Scheduling

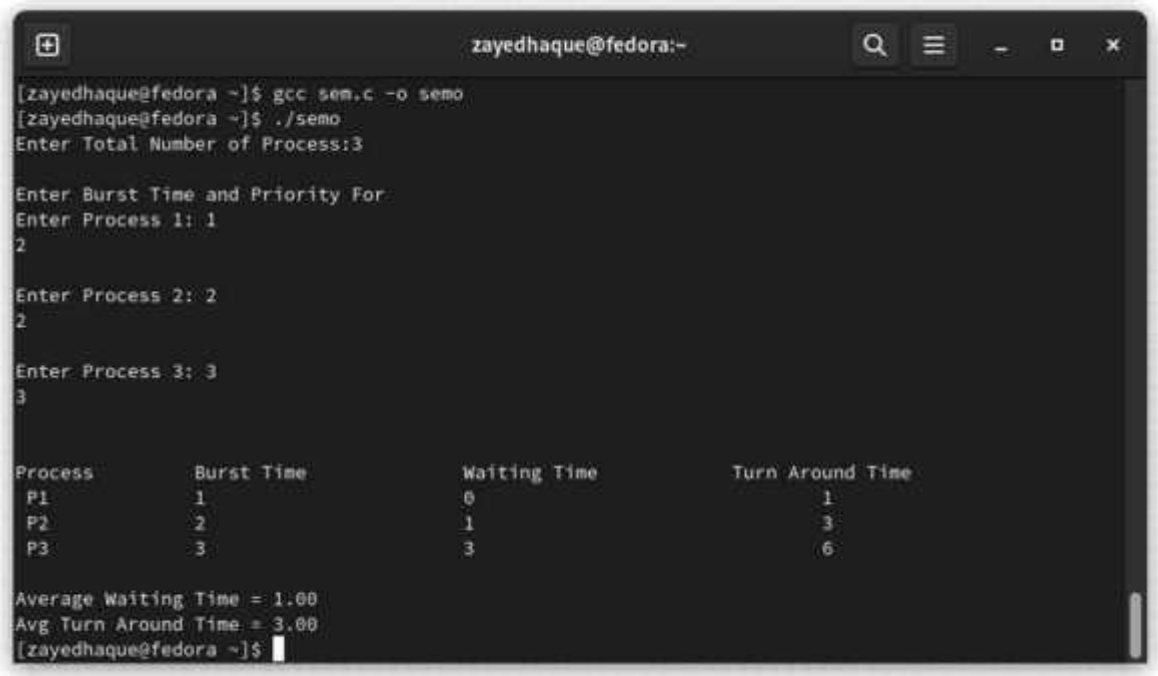
Program :

Priority Scheduling

```
#include<stdio.h>
#define max 10
int main()
{
int i,j,n,bt[max],p[max],wt[max],tat[max],pr[max],total=0,pos,temp;
float avg_wt,avg_tat;
printf("Enter Total Number of Process:");
scanf("%d",&n);
printf("\nEnter Burst Time and Priority For ");
for(i=0;i<n;i++)
{
printf("\nEnter Process %d: ",i+1);
scanf("%d",&bt[i]);
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])
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;
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;
total=0;
```

```
printf("\n\nProcess\t\tBurst Time\t\tWaiting Time\t\tTurn Around Time");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
total+=tat[i];
printf("\n P%d\t\t%d\t\t\t%d\t\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
   }
avg_tat=total/n;
printf("\n\nAverage Waiting Time = %.2f",avg_wt);
printf("\nAvg Turn Around Time = %.2f\n",avg_tat);
return 0;
}
```

Output :



Round Robin Scheduling

```
#include<stdio.h>

int main()
{
    //Input no of processed
    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;

    //Input details of processes
    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];
    }

    //Input time slot
    int time_slot;
    printf("Enter Time Slot:");
    scanf("%d", &time_slot);

    //Total indicates total time
    //counter indicates which process is executed
    int total = 0,  counter = 0,i;
    printf("Process ID      Burst Time      Turnaround Time      Waiting Time\n");
    for(total=0, i = 0; x!=0; )
    {
        // define the conditions
        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--; //decrement the process no.
            printf("\nProcess No %d  \t\t %d\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 :

```
zayedhaque@fedora:~
Avg Turnaround Time:21.666666[zayedhaque@fedora ~]$ gcc round.c -o roundo
[zayedhaque@fedora ~]$ ./roundo
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 Quantum: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
[zayedhaque@fedora ~]$
```

Result :

Implemented the concepts of Priority Scheduling and Round Robin Scheduling in C successfully