Ex. No.: 7d
Date 16.4.24

ROUND ROBIN SCHEDULING

Aim:

To implement the Round Robin (RR) scheduling technique

Algorithm:

1. Declare the structure and its elements.
2. Get number of processes and Time quantum as input from the user.
3. Read the process name, arrival time and burst time
4. Create an array **rem_bt[]** to keep track of remaining burst time of processes which is initially copy of bt[] (burst times array)
5. Create another array **wt[]** to store waiting times of processes. Initialize this array as 0.
6. Initialize time : t = 0
7. Keep traversing the all processes while all processes are not done. Do following for i'th process if it is not done yet.
   - a- If rem_bt[i] > quantum
     - (i) t = t + quantum
     - (ii) bt_rem[i] -= quantum;
   - b- Else // Last cycle for this process
     - (i) t = t + bt_rem[i];
     - (ii) wt[i] = t - bt[i]
     - (iii) bt_rem[i] = 0; // This process is over
8. Calculate the waiting time and turnaround time for each process.
9. Calculate the average waiting time and average turnaround time.
10. Display the results.

Program Code:

```
// vi rr.c
#include<stdio.h>
int main(){
    int i, n, total=0, x, counter=0, time_quant;
    int wait_time=0, tat=0;
    printf("Enter no. of processes: ");
    scanf("%d", &n);
    int at[n], bt[n], temp[n];
    x=n;
    for(i=0; i<n; i++){
        printf("Enter details of P%d:\n", i+1);
        printf("AT: ");
        scanf("%d", &at[i]);
        printf("BT:");
        scanf("%d", &bt[i]);
        temp[i]= bt[i];
    }
    printf("Enter time quantum: ");
    scanf("%d", &time_quant);
```

```
printf("\nProcess\t\t BT \t\t TAT \t\t WT \n");
for (total=0, i=0; x!=0;){
    if( temp[i]<=time_quant && temp[i]>0){
        total+=temp[i];
        temp[i]=0;
        count=1;
    }
    else if(temp[i]>0){
        temp[i]-=time_quant;
        total+=time_quant;
    }
    if(temp[i]==0 && count==1){
        x--;
        printf("\nP%d \t\t %d \t\t %d \t\t %d", i+1, bt[i],
            total-bt[i], total-at[i]-bt[i]);
        wait_time=wait_time+total-at[i]-bt[i];
        tat = tat+total-at[i];
        count=0;
    }
    if( i==n-1)
        i=0;
    else if(at[i+1]<=total)
        i++;
    else
        i=0;
}
avg_wt=wait_time*1.0/n;
avgtat: tat*1.0/n;
printf("Avg WT: %d", avg_wt);
printf("Avg TAT: %d", avg_tat);
return 0;
}
```

Output:

Enter no. of processes: 4

Enter details of Process 1

AT: 0

BT: 4

Enter details of P2

AT: 1

BT: 7

Enter details of P3

AT: 2

BT: 5

Enter details of P4

AT: 3

BT: 6

Enter time quantum: 3

Process	BT	TAT	WT
P1	4	13	9
P3	5	16	11
P4	6	18	12
P2	7	21	14

Avg WT : 11.500000

Avg TAT : 17.000000

RESULT:

The program has been compiled & executed successfully.

23/4/23