Ex. No.: 7d
Date: 13-04-2024

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:

```
#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\t\tArrival Time\t\tBurst Time\t\tTurnaround Time\t\tWaiting 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%d\t\t\t %d\t\t\t\t %d", i + 1, arr_time[i], 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:  2
Burst Time:   6
Enter Details of Process 3
Arrival Time:  3
Burst Time:   7
Enter Time Slot:5
Process ID              Arrival Time           Burst Time             Turnaround Time        Waiting Time

Process No 1             0                     10                     20                            10
Process No 2             2                     6                      19                            13
Process No 3             3                     7                      20                            13
Average Waiting Time:12.000000
Avg Turnaround Time:19.666666
```

Result:

Hence the C program to implement the Round Robin Scheduling technique has been successfully completed and executed and also verified