SIMULATION BASE ASSIGNEMENT ASSESMENT

On

<u>Scheduling of Processes, Turnaround Time and Average Waiting Time.</u>

Name: Rhituraj Sarkar, R.g:11802139 ,**Sec:**K18GA

Gmail:rhituraj1999@gmail.com,

GitHub: https://github.com/SnOoPdoG1999

Submit to: Pro. Navneet Kaur.

School of Computer Science and Engineering

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

DESCRIPTION:

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. Process scheduling is an essential part of a Multiprogramming operating systems.

The Operating System maintains the following important process scheduling queues

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.

Turnaround Time:

Turnaround time (TAT) means the amount of time taken to complete a process or fulfil a request. The concept thus overlaps with lead time and can be contrasted with cycle time.

Turnaround time = Burst time + Waiting time.

Or

Turnaround time = Exit time - Arrival time

Average Waiting Time:

Waiting time is the total time spent by the process in the ready state waiting for CPU.

Waiting time = Turnaround time - Burst time.

ALGORITHM:

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms.

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

From the above, the better one algorithm is ‘Round Robin Scheduling.

1- Create an array rem_bt[] to keep track of remaining

burst time of processes. This array is initially a

copy of bt[] (burst times array)

2- Create another array wt[] to store waiting times of processes. Initialize this array as 0.

3- Initialize time : t = 0

4- 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;

c- Else // Last cycle for this process

(i) t = t + bt_rem[i];

(ii) wt[i] = t - bt[i]

(ii) bt_rem[i] = 0; // This process is over

PURPOSE OF USE:

Round-robin (RR) is one of the algorithms employed by process and network scheduler in computing. As the term is generally used, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation -free. Round-robin scheduling can be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an operating system concept.

CODE SNIPPET:

```
#include <stdlib.h>

#include<unistd.h>

#include<stdio.h>

int main()

{

  int count,j,n;

  int time,remaining;

  int flag=0,time_quantum=10;

  int waiting_time=0,turn_around_time=0,arrival_time[10],burst_time[10],rt[10];

  printf("\n\nEnter the Total number of Process:\t ");

  scanf("%d",&n);
```

```
  remaining=n;
  for(count=0;count<n;count++)
  {
    printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
    scanf("%d",&arrival_time[count]);
    scanf("%d",&burst_time[count]);
    rt[count]=burst_time[count];
  }
  printf("Enter Time Quantum:%d\t",time_quantum);

  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(time=0,count=0;remaining!=0;)
  {
    if(rt[count]<=time_quantum && rt[count]>0)
    {
      time+=rt[count];
      rt[count]=0;
      flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-=time_quantum;
      time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remaining--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-arrival_time[count],time-
arrival_time[count]-burst_time[count]);
      waiting_time+=time-arrival_time[count]-burst_time[count];
      turn_around_time+=time-arrival_time[count];
```

```
    flag=0;
   }
   if(count==n-1)
    count=0;
   else if(arrival_time[count+1]<=time)
    count++;
   else
    count=0;
  }
  printf("\nAverage Waiting Time= %f\n",waiting_time*1.0/n);
  printf("Avg Turnaround Time = %f",turn_around_time*1.0/n);

  return 0;
}
```

Boundary Condition:

1. Time consuming scheduling for small quantums .
2. There is larger waiting time and response time.
3. There is low throughput.
4. There is contex switches.

Test Cases:

1) If we keep quantums time is too large, this become SJN/FCFS.
2) The number processes should be fixed because while calculating average waiting time, it get divide by number of process.
3) Number of processes, Arrival time and Brust time should be positive otherwise it tends to wrong calculation.
4) Exit time should be greater than arrival time, if it is not it give wrong turnaround time calculation.

Github: