Statement of the problem:

UNIX operating system uses Round robin Time slicing with multilevel feedback for process scheduling. Assume that the processes (PCB’s) arrive at random. Assume the time slice is 3 units. Simulate Round robin time slicing with fixed and variable (Randomized) burst times. Run the processes for fixed amount of time and find the average waiting time, average turn- around time of the processes completed.

Explantion of the problem:

Round robin is a cpu scheduling algorithrm where each process is assigned a fixed time slot in a cyclic way.

It is simple,easy to implement,and starvation-free as all processes get fair share of cpu.

One of the most commonly used technique in cpu scheduling as a core.

It is preemptive as processes are assigned cpu only for a fixed slice of time at most.

The disadvantages of it is more overhead of context switching.

In round robin there are three types of time computing they are:

1. Completion Time: Time at which process completes its execution.
2. Trun Around Time: Time difference between completion time and arrival time.Trun around time and brust time. Trun around time=Completion Time - Arrival Time
3. Waiting Time: Time difference between trun around time and brust time. Waiting Time=Trun Around Time – Brust Time.

The tricky part is to compute waiting times. Once waiting times are computed,trun around times can be quickly computed.

Illustration:

Round Robin example:

|  |  |  |  |
| --- | --- | --- | --- |
| process | Brust Time | Order | Arrival Time |
| P1 | 6 | 1 | 0 |
| P2 | 5 | 2 | 0 |
| P3 | 4 | 3 | 0 |

The quantum of time quantum is 3 units.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P1 | P3 |

0 3 6 9 12 13

P1 waiting time: (9-3)=6

P2 waiting time: 3

P3 waiting time: (12-3)=9

The average waiting time: (6+3+9)/3=6

Turn around time:

P1 turn around time: 12

P2 turn around time: 6

P3 turn around time: 13

The average of turn around time: (12+6+13)/3=10.33

ALGORITHM

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