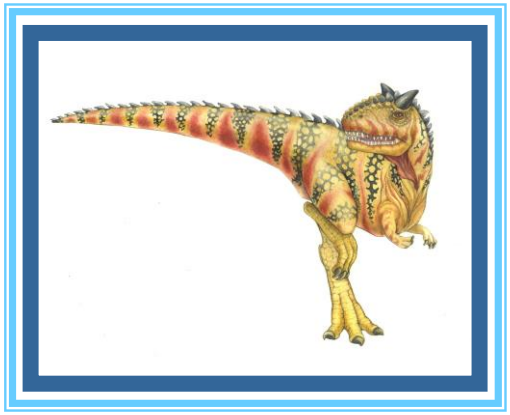
Operating System LAB-03

Prepared By

Md. Mustafizur Rahman

Lecturer, CSE,DCC.

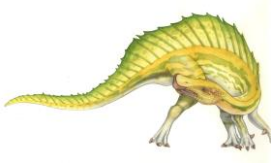


EXPERIMENT NO-3

CPU SCHEDULINGALGORITHMS

ROUND ROBIN

- AIM: To simulate the CPU scheduling algorithm round-robin



EXPERIMENT NO-03

DESCRIPTION

To aim is to calculate the average waiting time. There will be a time slice, each process should be executed within that time-slice and if not it will go to the waiting state so first check whether the burst time is less than the time-slice. If it is less than it assign the waiting time to the sum of the total times. If it is greater than the burst-time then subtract the time slot from the actual burst time and increment it by time-slot and the loop continues until all the processes are completed.

EXPERIMENT NO-03: ALGORITHM

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Calculate the no. of time slices for each process where No. of time slice for process (n) = burst time process (n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

EXPERIMENT NO-03: ALGORITHM

a) Waiting time for process (n) = waiting time of process(n-1)+ burst time of process(n-1) + the time difference in getting the CPU from process(n-1)

b) Turnaround time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).

Step 7: Calculate

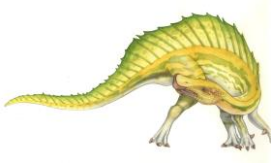
c) Average waiting time = Total waiting Time / Number of process

d) Average Turnaround time = Total Turnaround Time / Number of process Step

8: Stop the process

Round Robin (RR)

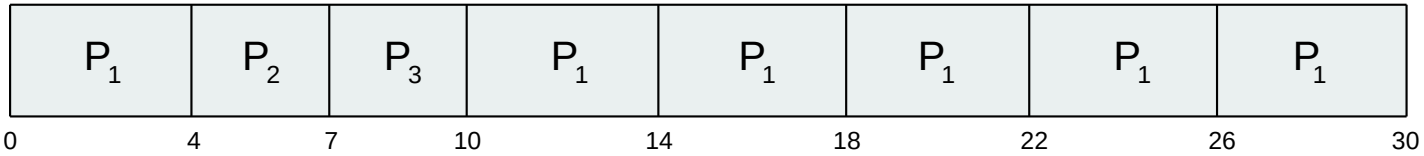
- Each process gets a small unit of CPU time (**time quantum** *q*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
- Timer interrupts every quantum to schedule next process
- Performance
  - *q* large $\Rightarrow$ FIFO
  - *q* small $\Rightarrow$ *q* must be large with respect to context switch, otherwise overhead is too high



Example of RR with Time Quantum = 4

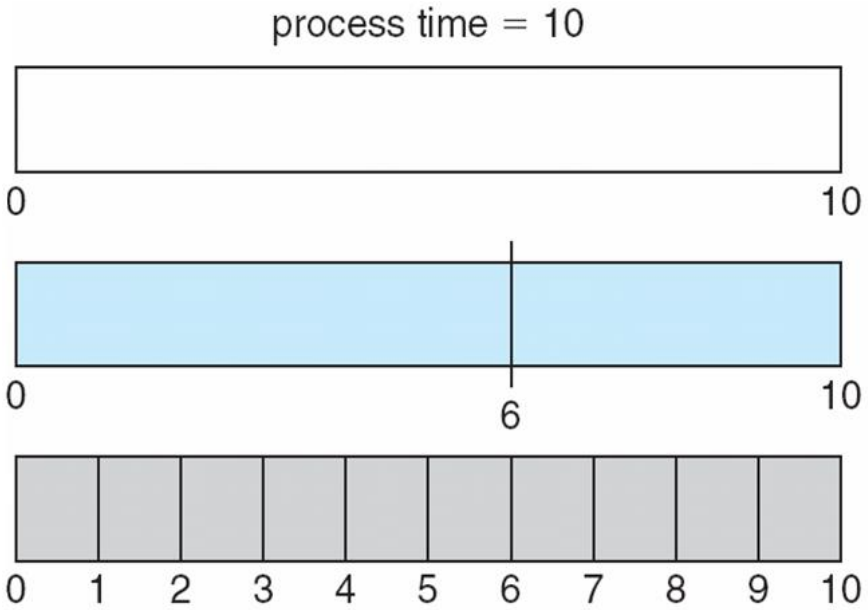
Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better ***response***
- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 usec

Time Quantum and Context Switch Time



quantum	context switches
12	0
6	1
1	9

EXPERIMENT NO-03: C PROGRAM

```
#include<stdio.h>
int main()
{
   int n,i,qt,count=0,temp,sq=0,bt[10],wt[10],tat[10],rem_bt[10];
   float awt=0,atat=0;
   printf("Enter the Number of process: ");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("Enter burst time for Process P%d~~",i+1);
      scanf("%d",&bt[i]);
      rem_bt[i]=bt[i];
   }
   printf("Enter quantum time: ");
   scanf("%d",&qt);
```

EXPERIMENT NO-03: C PROGRAM

```
while(1)
   {
      for(i=0,count=0;i<n;i++)
      {
         temp = qt;
         if(rem_bt[i]==0)
         {
            count++;
            continue;
         }
         if(rem_bt[i]>qt)
            rem_bt[i]=rem_bt[i]-qt;
         else
            if(rem_bt[i]>=0)
            {
               temp=rem_bt[i];
               rem_bt[i]=0;
            }
```

EXPERIMENT NO-03: C PROGRAM

```
sq=sq+temp;
            tat[i]=sq;
      }
      if(n==count)
         break;
   }
   printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");
   for(i=0;i<n;i++)
   {
      wt[i]=tat[i]-bt[i];
      awt=awt+wt[i];
      atat=atat+tat[i];
      printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,bt[i],wt[i],tat[i]);
   }
   awt=awt/n;
   atat=atat/n;
   printf("\nThe Average Turnaround time is -- %f",atat);
   printf("\nThe Average Waiting time is -- %f ",awt);
   return 0;
}
```

End of Lab 02