Ex No. 3A. Program for Multi-level Feedback scheduling

Dr S.Rajarajan/SoC

Let us assume that all the processes arrive at the same time at time 0. So we don't need to consider arrival time for scheduling

Process	P0	P1	P2	P3	P4	P5
BT	4	3	6	2	5	8
FT	14	15	19	8	22	28

Number of queues: 2
Time quantum for each queue: 2,4

TQ=2

1-2 P0(2)	2-4 P1(1)	4-6 P2(4)	6-8 P3(0)	8-10 P4(3)	10-12 P5(6)				

TQ=4

12-14 P0(0)	14-15 P1(0)	15-19 P2(0)	19-22 P4(0)	22-26 P5(2)					

26-28 P5(0)									

Program:

```
#include <stdio.h>
struct process
{
                        int pid;
                        int bt;
                        int rbt; // remaining burst time
                        int ft;
}rdq[10];
int n;
void main()
{
                        int tqq[10], completed=0,qlevel=0,qcount,i, tq, ct=0;
                        printf("Enter number of queues");
                        scanf("%d",&qcount);
                        for(i=0;i<qcount;i++)
                        {
                              printf("Enter time quantum of Q%d",i);
```

```
        scanf("%d",&tqq[i]);
    }
    printf("Enter number of processes:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter BT of process P%d: ",i);
        scanf("%d",&rdq[i].bt);
        rdq[i].rbt=rdq[i].bt;
    }
    while(completed <n)
    {
        if(qlevel<qcount)  // Reached last level queue or not
        {
                tq=tqq[qlevel];
                qlevel++;
        }
        for(i=0;i<n;i++)
        {
                if(rdq[i].rbt!=0) // process not yet completed
                {
                        if(rdq[i].rbt<=tq ) //remaining burst time is <= time
                        //quantum
                        {
                                ct+=rdq[i].rbt;
                                rdq[i].rbt=0;
                                rdq[i].ft=ct;
                                completed++;
                        }
                        else //remaining burst time is > time quantum
                        {
                                ct+=tq;
                                rdq[i].rbt-=tq;
                        }
                }
        }
    }
    printf("PID\t BT\t FT\n");
    for(i=0;i<n;i++)
    {
        printf("P%d\t%d\t%d\n",i,rdq[i].bt,rdq[i].ft);
    }
}
```