

OPERATION SYSTEM ASSIGNMENT

NAME: ARAVIND KOTTAKOTA SECTION:K18JE ROLL NO:A13 GROUP:A

Write a program for multilevel queue scheduling algorithm. There must be three queues generated. There must be specific range of priority associated with every queue. Now prompt the user to enter number of processes along with their priority and burst time. Each process must occupy the respective queue with specific priority range according to its priority. Apply Round robin algorithm with quantum time 4 on queue with highest priority range. Apply priority scheduling algorithm on the queue with medium range of priority and First come first serve algorithm on the queue with lowest range of priority. Each and every queue should get a quantum time of 10 seconds. Cpu will keep on shifting between queues after every 10 seconds i.e. to apply round robin algorithm OF 10 seconds on over all structure.

Calculate Waiting time and turnaround time for every process. The input for number of processes should be given by the user.

CODE:

- `#include<stdio.h>`
- `int arrival_time1[30],arrival_time2[30],priority2[30],process2[30],arrival_time3[30];`
- `int burst_time1[30],burst_time2[30],burst_time3[30];`
-
- `int Total=0,t1=0,t2=0,t3=0;`
-
- `int n,i,at[30],bt[30],pr[30],j=0,k=0,l=0;`
-
- `int total,x,temp[30],counter=0;`
- `float avg_waiting_time1=0.0,avg_turnaround_time1=0.0;`
-
- `int p,waiting_time3[30],turnaround_time3[30];`
- `float avg_waiting_time3=0.0,avg_turnaround_time3=0.0;`
-
- `int position,q,temp1,sum=0,waiting_time2[30],turnaround_time2[30];`
- `float avg_waiting_time2,avg_turnaround_time2;`
-
- `void round_robin()`
- `{`
- `printf("Time Quantum for Queue1 is 4\n");`
- `for(i=0;i<j;i++)`

- {
- temp[i]=burst_time1[i];
- }
- printf("\nProcess ID\tBurst Time\t Turnaround Time\t Waiting Time\n");
- x=j;
- for(i=0,total=0;x!=0;)
 - {
 - if(temp[i]<=4&&temp[i]>0)
 - {
 - printf("\nProcess[%d] of Queue1 is running for %d units",i+1,temp[i]);
 - total=total+temp[i];
 - temp[i]=0;
 - counter=1;
 - }
 - else if(temp[i]>0)
 - {
 - printf("\nProcess[%d] of Queue1 is running for 4 units",i+1);
 - temp[i]=temp[i]-4;
 - total=total+4;
 - }
 - if(temp[i]==0&&counter==1)
 - {
 - x--;
 - printf("\nProcess[%d]\t%d\t%d\t%d",i+1,burst_time1[i],total-arrival_time1[i],total-arrival_time1[i]-burst_time1[i]);
 - avg_waiting_time1=avg_waiting_time1+total-arrival_time1[i]-burst_time1[i];
 - avg_turnaround_time1=avg_turnaround_time1+total-arrival_time1[i];
 - counter = 0;
 - }
 - if(i==j-1)
 - {
 - i=0;
 - }
 - else if(arrival_time1[i+1]<=total)
 - {
 - i++;
 - }
 - else
 - {
 - i=0;

- }
- }
- avg_waiting_time1=avg_waiting_time1/j;
- avg_turnaround_time1=avg_turnaround_time1/j;
- printf("\nAverage Waiting Time:%f",avg_waiting_time1);
- printf("\nAverage Turnaround Time:%f\n",avg_turnaround_time1);
- }
-
- void priority()
- {
- for(i=0;i<k;i++)
- {
- position=i;
- for(q=i+1;q<k;q++)
- {
- if(priority2[q]<priority2[position])
- {
- position=q;
- }
- }
- temp1=priority2[i];
- priority2[i]=priority2[position];
- priority2[position]=temp1;
-
- temp1=burst_time2[i];
- burst_time2[i]=burst_time2[position];
- burst_time2[position]=temp1;
-
- temp1=process2[i];
- process2[i]=process2[position];
- process2[position]=temp1;
- }
- waiting_time2[0]=0;
- for(i=1;i<k;i++)
- {
- waiting_time2[i]=0;
- for(q=0;q<i;q++)
- {
- waiting_time2[i]=waiting_time2[i]+burst_time2[j];
- }
- }

- sum=sum+waiting_time2[i];
- }
- avg_waiting_time2=sum/k;
- sum=0;
- printf("\nProcess ID\t\tBurst Time\t Waiting Time\t Turnaround Time\n");
- for(i=0;i<k;i++)
- {
- turnaround_time2[i]=burst_time2[i]+waiting_time2[i];
- sum=sum+turnaround_time2[i];
- printf("\nProcess[%d]\t\t%d\t\t %d\t\t
- %d\n",process2[i],burst_time2[i],waiting_time2[i],turnaround_time2[i]);
- }
- avg_turnaround_time2=sum/k;
- printf("\nAverage Waiting Time:\t%f",avg_waiting_time2);
- printf("\nAverage Turnaround Time:\t%f\n",avg_turnaround_time2);
-
- for(i=0;i<k;i++)
- {
- while(burst_time2[i]!=0)
- {
- if(burst_time2[i]>10)
- {
- printf("\nProcess[%d] of Queue2 is running for 10 units",i+1);
- burst_time2[i]=burst_time2[i]-10;
- }
- else if(burst_time2[i]<=10)
- {
- printf("\nProcess[%d] of Queue2 is running for %d
- units",i+1,burst_time2[i]);
- burst_time2[i]=0;
- }
- }
- }
-
- }
-
- void fcfs()
- {
- waiting_time3[0] = 0;
- for(i=1;i<l;i++)

```

• {
•     waiting_time3[i] = 0;
•     for(p=0;p<l;p++)
•     {
•         waiting_time3[i]=waiting_time3[i]+burst_time3[p];
•     }
• }
• printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time\n");
• for(i=0;i<l;i++)
• {
•     turnaround_time3[i]=burst_time3[i]+waiting_time3[i];
•     avg_waiting_time3=avg_waiting_time3+waiting_time3[i];
•     avg_turnaround_time3=avg_turnaround_time3+turnaround_time3[i];
•
•     printf("\nProcess[%d]\t\t%d\t\t%d\t\t%d\n",i+1,burst_time3[i],waiting_time3[i],turnaround_time3[i]);
• }
• avg_waiting_time3=avg_waiting_time3/l;
• avg_turnaround_time3=avg_turnaround_time3/l;
• printf("\nAverage Waiting Time=%f",avg_waiting_time3);
• printf("\nAverage Turnaround Time=%f",avg_turnaround_time3);
• for(i=0;i<l;i++)
• {
•     while(burst_time3[i]!=0)
•     {
•         if(burst_time3[i]>10)
•         {
•             printf("\nProcess[%d] of Queue3 is running for 10 units",i+1);
•             burst_time3[i]=burst_time3[i]-10;
•         }
•         else if(burst_time3[i]<=10)
•         {
•             printf("\nProcess[%d] of Queue2 is running for %d
units",i+1,burst_time3[i]);
•             burst_time3[i]=0;
•         }
•     }
• }
• }
•

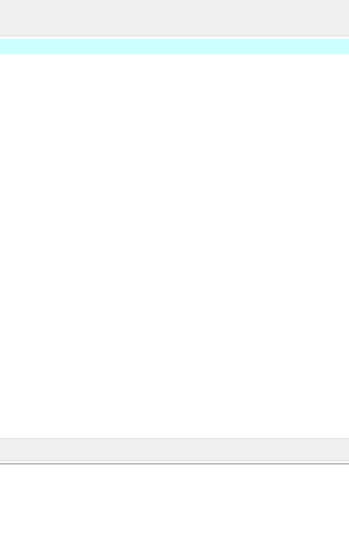
```

```

• void round_robin1()
• {
•     printf("Time Quantum between the 3 queues is 10\n");
•     for(i=1;i<Total;i=i+10)
•     {
•         if(t1>10)
•         {
•             printf("Queue1 is running for 10 units\n");
•             t1=t1-10;
•         }
•         else if(t1<=10&&t1!=0)
•         {
•             printf("Queue1 is running for %d units\n",t1);
•             t1=0;
•         }
•         if(t2>10)
•         {
•             printf("Queue2 is running for 10 units\n");
•             t2=t2-10;
•         }
•         else if(t2<=10&&t2!=0)
•         {
•             printf("Queue2 is running for %d units\n",t2);
•             t2=0;
•         }
•         if(t3>10)
•         {
•             printf("Queue3 is running for 10 units\n");
•             t3=t3-10;
•         }
•         else if(t3<=10&&t3!=0)
•         {
•             printf("Queue3 is running for %d units\n",t3);
•             t3=0;
•         }
•     }
• }
•
• int main()
• {

```

- printf("Enter the no. of process you want to enter\n");
- scanf("%d",&n);
- for(i=0;i<n;i++)
- {
- printf("Enter details of process[%d]\n",i+1);
- printf("Arrival Time:");
- scanf("%d",&at[i]);
- printf("Burst Time:");
- scanf("%d",&bt[i]);
- printf("Priority(1 to 15):");
- scanf("%d",&pr[i]);
- Total=Total+bt[i];
- }
- for(i=0;i<n;i++)
- {
- if(pr[i]>=1&&pr[i]<=5)
- {
- printf("\n\nProcess[%d] belongs to Queue 1\n",i+1);
- arrival_time1[j]=at[i];
- burst_time1[j]=bt[i];
- j++;
- t1=t1+bt[i];
- }
- else if(pr[i]>=6&&pr[i]<=10)
- {
- printf("Process[%d] belongs to Queue 2\n",i+1);
- arrival_time2[k]=at[i];
- burst_time2[k]=bt[i];
- priority2[k]=pr[i];
- process2[k]=k+1;
- k++;
- t2=t2+bt[i];
- }
- else if(pr[i]>=11&&pr[i]<=15)
- {
- printf("Process[%d] belongs to Queue 3\n\n\n",i+1);
- arrival_time3[l]=at[i];
- burst_time3[l]=bt[i];

A screenshot of a Windows command prompt window. The title bar at the top reads "Command Prompt". The window has a light blue header bar. The command prompt shows the text "C:\>" followed by the command "echo [30];" entered in red. The cursor is at the end of the command. The taskbar at the bottom shows the system clock as 19:55 on 09-04-2020, and the language is set to ENG.