**Lovely Professional university**



Subject: CSE 316

Name: Kamisetty Vishal

Section: K17GZ

Roll No: A08

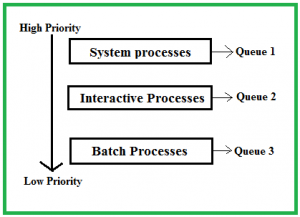
GitHub: https://github.com/vishal2213

**Description:**

The project is based on the concept of CPU scheduling using multilevel queue scheduling algorithm. Each queue is assigned some priority according to the processes in them.

**Multilevel Queue scheduling algorithm:** It is an algorithm that partitions the single ready queue into a number of ready queues according to the priority assigned by the CPU or the user itself. These queues are also known as priority queues, there can be a 2 or even more than 2 priority queues.

**Ready Queue** is divided into separate queues for each class of processes. For example, let us take three different types of process System processes, Interactive processes and Batch Processes. All three process have their own queue. Now look at the below figure.



All three different type of processes have there own queue. Each queue have its own Scheduling algorithm. For example, queue 1 and queue 2 uses **Round Robin** while queue 3 can use **FCFS** to schedule there processes. In this method each queue gets certain portion of CPU time and can use it to schedule its own processes.

The process which are present in the first and second ready queue are executed by the round robin algorithm and the remaining third queue processes will be executed by the process of FCFS.

**FCFS:** First in, first out (FIFO), also known as first come, first served (FCFS), is the simplest scheduling algorithm. FIFO simply queues processes in the order that they arrive in the ready queue.

**Steps or Algorithm of the Multilevel Queue Scheduling algorithm**

1. Take the inputs required for the execution of the program (such as the No. of processes, arrival time of each process, burst time of each process and the priority assigned to each of the algorithm).
2. Push the very first process inside the queue(whether it is Q1 or Q2) according to their priority. Ex- Insert faculty query in Q1 and the students query inside the Q2.
3. Execute all the processes present in Q1 by using the Round-Robin algorithm utill it becomes empty, after completing the Q1 go ahead for the Q2 and execute its process by First Come First Serve(FCFS) method.
4. If any process arrives with the priority of 1 or for the Q1, immediately preampt the currectly executing process of Q2, push it at the end of Q2 and start executing the Q1 again utill it becomes empty.
5. When both all the processes are executed and the queues are empty then terminate the loop and print the Completion time, Turnaround time and the waiting time.

**Complexity:** The complecity for the each Loop will be O(n), here n is the maximum number of queries i.e, 10. The complexity of the whole code is O(n\*2).

**Source Code:**

*#include<iostream>*

*#include<stdio.h>*

*#include<stdlib.h>*

*using namespace std;*

*int i,j,n,at[10],ct[10],bt[10],rt[10],pr[10],q1[50],q2[10],p[10];*

*int qt,t,temp,count=0,f2=-1,r2=-1,f1=-1,r1=-1;*

*int wt[10],tat[10],flag=1,time=0;*

*//Function to check if queue is empty or not*

*bool isempty(int front,int rear)*

*{*

*if(front == -1 && rear == -1)*

*return true;*

*else*

*return false;*

*}*

*//function to enter elements in queue1*

*void enqueue1(int value )*

*{*

*if ((r1 + 1) == f1)*

*cout<<"Queue is full \n";*

*else*

*{*

*if( f1 == -1)*

*f1 = 0;*

*r1 = (r1+1);*

*q1[r1] = value;*

*}*

*}*

*//function to enter elements in queue2*

*void enqueue2(int value)*

*{*

*if ((r2 + 1) == f2)*

*cout<<"Queue is full \n";*

*else*

*{*

*if( f2 == -1)*

*f2 = 0;*

*r2 = (r2+1);*

*q2[r2] = value;*

*}*

*}*

*//function to delete/remove element from queue1*

*int dequeue1()*

*{*

*if( isempty(f1,r1) )*

*cout<<"Queue is empty\n";*

*else*

*//only one element*

*if( f1 == r1 )*

*f1 = r1 = -1;*

*else*

*f1 = (f1 + 1);*

*return p[f1-1];*

*}*

*//function to delete/remove element from queue2*

*int dequeue2()*

*{*

*if( isempty(f2,r2) )*

*cout<<"Queue is empty\n";*

*else*

*//only one element*

*if( f2 == r2 )*

*f2 = r2 = -1;*

*else*

*f2 = (f2 + 1);*

*return p[f2-1];*

*}*

*void takevalues()*

*{*

*for(i=0;i<n;i++)*

*{*

*cout<<"Query "<<i+1<<"\n";*

*cout<<"Arrival Time";*

*cin>>at[i];*

*cout<<"Burst Time";*

*cin>>bt[i];*

*cout<<"Priority(1 for Faculty and 2 for Student)";*

*cin>>pr[i];*

*rt[i]=bt[i];*

*p[i]=i;*

*}*

*}*

*/\*void check(int x);*

*{*

*}\*/*

*void show()*

*{*

*cout<<"Summary\n\n";*

*cout<<"Query Number\tArrival Time\tBurst Time\tPriority\tTurn-around Time\tWaiting Time\n";*

*for(i=0;i<n;i++)*

*{*

*cout<<"Query "<<p[i]+1<<"\t\t"<<at[i]<<"\t\t"<<bt[i]<<"\t\t"<<pr[i]<<"\t\t"<<tat[i]<<"\t\t\t"<<wt[i]<<"\n";*

*}*

*cout<<"\n\nThe given Quantum Time was ",qt;*

*}*

*void calculate()*

*{*

*if(pr[0]==1)*

*enqueue1(p[0]);*

*else*

*enqueue2(p[0]);*

*while(flag==1)*

*{*

*if(isempty(f1,r1)!=true)*

*{*

*while(r1<=f1)*

*{*

*j=dequeue1();*

*cout<<"q1"<<j<<"\n";*

*if(rt[j]>t&&rt[j]>0)*

*{*

*rt[j]-=t;*

*time+=t;*

*t=qt;*

*}*

*else if(rt[j]<t&&rt[j]>0)*

*{*

*t-=rt[j];*

*time+=(qt-t);*

*ct[j]=time;*

*}*

*else*

*{*

*time+=t;*

*ct[j]+=time;*

*rt[j]=0;*

*t=qt;*

*}*

*//check(j);*

*for(i=at[j];i<=time;i++)*

*{*

*if(at[i]<time&&pr[i]==1)*

*{*

*enqueue1(p[i]);*

*}*

*if(at[i]<time&&pr[i]==2)*

*{*

*enqueue2(p[i]);*

*}*

*}*

*if(rt[j]>0)*

*enqueue1(p[j]);*

*}*

*}*

*else*

*{*

*while(isempty(f2,r2)!=true)*

*{*

*j=dequeue2();*

*cout<<"q2"<<j<<"\n";*

*while(rt[j]>0)*

*{*

*if(rt[j]>t&&rt[j]>0)*

*{*

*rt[j]-=t;*

*time+=t;*

*t=qt;*

*}*

*else if(rt[j]<t&&rt[j]>0)*

*{*

*t-=rt[j];*

*time+=(qt-t);*

*ct[j]=time;*

*}*

*else*

*{*

*time+=t;*

*ct[j]+=time;*

*rt[j]=0;*

*t=qt;*

*}*

*//check(j);*

*for(i=at[i];i<=time;i++)*

*{*

*if(at[i]<time&&pr[i]==1)*

*{*

*enqueue1(p[i]);*

*}*

*if(at[i]<time&&pr[i]==2)*

*{*

*enqueue2(p[i]);*

*}*

*}*

*}*

*}*

*flag=0;*

*}*

*}*

*for(i=0;i<n;i++)*

*{*

*tat[i]=ct[i]-at[i];*

*wt[i]=tat[i]-bt[i];*

*}*

*}*

*main()*

*{*

*cout<<"Welcome to the Query session of the Linux Expert Mr. Sudesh Sharma\n\n\n";*

*cout<<"Enter the number of queries you want to ask(Maximum 10)\n";*

*cin>>n;*

*count=n;*

*cout<<"Now enter the arrival time, burst time, priority and the name of the process\n";*

*cout<<"Enter the processes according to their arrival timr in ascending order\n\n";*

*takevalues();*

*cout<<"Enter the time to be given to each query\n";*

*cin>>qt;*

*t=qt;*

*calculate();*

*//show();*

*for(i=0;i<n;i++)*

*{*

*sum1+=tat[i];*

*sum2+=wt[i];*

*}*

*cout<<"\n The average turn-ariound time is"<<sum1/n;*

*cout<<"\n The average turn-ariound time is"<<sum2/n;*

*}*

**Constraints Given in the Problem**

The Question was:

Sudesh Sharma is a Linux expert who wants to have an online system where he can handle student queries. Since there can be multiple requests at any time he wishes to dedicate a fixed amount of time to every request so that everyone gets a fair share of his time. He will log into the system from 10am to 12am only.  He wants to have separate requests queues for students and faculty. Implement a strategy for the same. The summary at the end of the session should include the total time he spent on handling queries and average query time.

>> The Problem was to design a scheduling program that is having two seperate queues for both student and the faculty so that he can solve the query of the teacher first and after that he qill give his time to solve student’s problem.

>> If at any point while solving the student’s query, any faculty has requested for the time, then he will preampt the student’s query and solve the student have to wait until the teacher’s problem is not solved.

**Test Case:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst time | Priority | Turn around | Wating time |
| 1 | 0 | 4 | 1 | 6 | 2 |
| 2 | 1 | 3 | 1 | 8 | 5 |
| 3 | 3 | 5 | 2 | 11 | 6 |
| 4 | 4 | 2 | 1 | 4 | 2 |
| Avg: |  |  |  | 7.25 | 3.75 |