**Name: Sanjoy karmakar**

**Roll no: 33**

**Registration No.:** 11902783

**Section:** K18AW

**Email Address:** sanjoykarmakar0512@gmail.com

**GitHub link:**

**Question No. 7:**

Researchers designed one system that classified interactive and noninteractive processes automatically by looking at the amount of terminal I/O. If a process did not input or output to the terminal in a 1-second interval, the process was classified as noninteractive and was moved to a lower-priority queue. In response to this policy, one programmer modified his programs to write an arbitrary character to the terminal at regular intervals of less than 1 second. The system gave his programs a high priority, even though the terminal output was completely meaningless.

**Explanation:**

1. In the given solution I have taken classified interactive and noninteractive processes automatically by looking at the amount of terminal I/O.
2. User will get a menu to select any two out of three resources that are to be placed on shared table.
3. If a process did not input or output to the terminal in a 1-second interval, the process was classified as noninteractive and was moved to a lower-priority queue.
4. In response to this policy, one programmer modified his programs to write an arbitrary character to the terminal at regular intervals of less than 1 second
5. When a process starts to execute it acquires the lock and when it completes the execution releases the lock.
6. After completion of all the three processes the program will end

**Solution:**

#include<stdio.h>

int main()

{

int i, type[20],n;

int resptime[20];

printf("Number of process: ");

scanf("%d",&n);

printf("Enter the data\n");

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

{

printf("Response time of P%d (in milliseconds): ",i);

scanf("%d",&resptime[i]);

if(resptime[i]<1000)

{

type[i]=1;

}

else

{

type[i]=0;

}

}

printf("Process Number\tResponse Time\tType\t\tPriority");

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

{

printf("\nP%d\t\t%dms\t\t",i,resptime[i]);

if(type[i]==1)

{

printf("Interactive\tHigh");

}

else

{

printf("Non-Interactive\tLow");

}

}

}

**Question No. 20:**

There are 3 student processes and 1 teacher process. Students are supposed to do their assignments and they need 3 things for that-pen, paper and question paper. The teacher has an infinite supply of all the three things. One students has pen, another has paper and another has question paper. The teacher places two things on a shared table and the student having the third complementary thing makes the assignment and tells the teacher on completion. The teacher then places another two things out of the three and again the student having the third thing makes the assignment and tells the teacher on completion. This cycle continues. WAP to synchronise the teacher and the students

**Explanation:**

1. In the given solution I have taken all the student processes and resources in 2D array and initialized then to 0.
2. I have made 3 student processes in three different functions which will be executed by single s\_thread and one t\_thread for execution of teacher process.
3. User will get a menu to select any two out of three resources that are to be placed on shared table.
4. If one process is completed there will be a message printed on the screen saying process is completed.
5. When one process is executing no other student or teacher process will execute and for achieving this I have used Mutex lock.
6. When a process starts to execute it acquires the lock and when it completes the execution releases the lock.
7. After completion of all the three processes the program will end

**Solution:**

#include<iostream>

#include<conio.h>

using namespace std;

struct Student

{

bool resource\_pen;

bool resource\_paper;

bool resource\_question\_paper;

bool resource\_all\_resources;

}s1,s2,s3;

void student\_process\_one()

{

s1.resource\_all\_resources=1;

s1.resource\_paper=1;

s1.resource\_question\_paper=1;

cout<<"Student Process One Completed"<<endl;

}

void student\_process\_two()

{

s2.resource\_all\_resources=1;

s2.resource\_pen=1;

s2.resource\_question\_paper=1;

cout<<"Student Process Two Completed"<<endl;

}

void student\_process\_three()

{

s3.resource\_all\_resources=1;

s3.resource\_pen=1;

s3.resource\_paper=1;

cout<<"Student Process Three Completed"<<endl;

}

int main()

{

s1.resource\_all\_resources=0;s1.resource\_paper=0;s1.resource\_pen=0;s1.resource\_question\_paper=0;

s2.resource\_all\_resources=0;s2.resource\_paper=0;s2.resource\_pen=0;s2.resource\_question\_paper=0;

s3.resource\_all\_resources=0;s3.resource\_paper=0;s3.resource\_pen=0;s3.resource\_question\_paper=0;

do

{

int a,b;

cout<<"1.Pen"<<endl;

cout<<"2.Paper"<<endl;

cout<<"3.Question Paper"<<endl;

cout<<"Select any two items to be placed on the shared table:"<<endl;

cin>>a>>b;

if(a==1 && b==2 && s3.resource\_all\_resources==0)

{

student\_process\_three();

}

if(a==2 && b==1 && s3.resource\_all\_resources==0)

{

student\_process\_three();

}

if(a==2 && b==3 && s1.resource\_all\_resources==0)

{

student\_process\_one();

}

if(a==3 && b==2 && s1.resource\_all\_resources==0)

{

student\_process\_one();

}

if(a==1 && b==3 && s2.resource\_all\_resources==0)

{

student\_process\_two();

}

if(a==3 && b==1 && s2.resource\_all\_resources==0)

{

student\_process\_two();

}

}

while(s1.resource\_all\_resources==0||s2.resource\_all\_resources==0||s3.resource\_all\_resources==0);

cout<<"All Student Processes Completed";

getch();

}