学号 P21714001 专业 计算机英才班 姓名 刘峰

实验日期 **2019.12.2**  教师签字 成绩

实验报告

【实验名称】 银行家算法实现死锁的避免

【实验目的】

为了了解系统的资源分配情况，假设系统的任何一种资源在任意时刻只能被一个进程使用。任何进程已经占用的资源只能由进程自己释放，而不能由其他进程抢占。当进程申请的资源不能满足时必须等待。银行家算法就是对系统进行资源调度的程序。一个是随机动态的进行资源分配的模拟程序，即只要系统当前剩余资源满足进程的当前请求，就立即将资源分配给进程，以观察死锁产生的情况。

【实验原理】

设计3-4个并发进程，共享系统的10个同类不可抢占的的资源。各进程动态进行资源的申请和释放

【实验内容】

#include <iostream>

#include <iomanip>

#include <fstream>

using namespace std;

#define MaxNumber 20

static int n;

static int m;

static int Available[MaxNumber];

static int Max[MaxNumber][MaxNumber];

static int Allocation[MaxNumber][MaxNumber];

static int Need[MaxNumber][MaxNumber];

static int Request[MaxNumber];

static int SafeOrder[MaxNumber];

static bool Finish[MaxNumber];

static bool isDisplayAvaliable = true;

static char sourceName[] = {'A','B','C','D','E','F','G','H','I','J','K'}; //资源名称

void display()

{

char processName[] = {'1','2','3','4','5','6'};

cout<<"----------------------------------------------------------------------"<<endl;

cout<<"当前进程个数为 n = "<<n<<endl;

cout<<"当前资源个数为 m = "<<m<<endl;

cout<<"系统可利用资源数情况如下："<<endl;

for (int i=0;i<m;i++)

{

cout<<setw(5)<<sourceName[i]<<" ";

}

cout<<endl;

cout<<setw(5)<<Available[0]<<" ";

cout<<setw(5)<<Available[1]<<" ";

cout<<setw(5)<<Available[2]<<" "<<endl;

cout<<"------------------------------------------------------------------------"<<endl;

cout<<"processName";

cout<<setw(10)<<"Max[][]"<<" ";

cout<<setw(15)<<"Allocation[][]"<<" ";

cout<<setw(10)<<"Need[][]"<<" ";

if (isDisplayAvaliable)

{

cout<<setw(15)<<"available[][]";

}

cout<<endl;

cout<<setw(15);

for (int i=0;i<m;i++)

{

cout<<sourceName[i]<<setw(3);

}

cout<<setw(5);

for (int i=0;i<m;i++)

{

cout<<sourceName[i]<<setw(3);

}

cout<<setw(10);

for (int i=0;i<m;i++)

{

cout<<sourceName[i]<<setw(3);

}

if (isDisplayAvaliable)

{

cout<<setw(5);

for (int i=0;i<m;i++)

{

cout<<sourceName[i]<<setw(3);

}

}

cout<<endl;

//设置矩阵输出形式

bool OnlyOneLineAvaliable = true; //Avaliable数组只输出第一行

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

{

cout<<setw(5)<<"P"<<processName[i];

cout<<setw(10);

for (int j = 0;j<m;j++)

{

cout<<Max[i][j]<<setw(3);

}

cout<<setw(5);

for (int j = 0;j<m;j++)

{

cout<<Allocation[i][j]<<setw(3);

}

cout<<setw(10);

for (int j = 0;j<m;j++)

{

cout<<Need[i][j]<<setw(3);

}

if (isDisplayAvaliable&&OnlyOneLineAvaliable)

{

cout<<setw(5);

for (int i=0;i<m;i++)

{

cout<<Available[i]<<setw(3);

}

OnlyOneLineAvaliable = false;

}

cout<<endl;

}

}

void input()

{

ifstream inData;

inData.open("C:\\Users\\Administrator\\Desktop\\操作系统实验\\银行家算法\\data.txt");

//读取数据

inData>>n;

inData>>m;

for (int i=0;i<m;i++)

{

inData>>Available[i];

}

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

{

for (int j=0;j<m;j++)

{

inData>>Allocation[i][j];

}

}

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

{

for (int j=0;j<m;j++)

{

inData>>Need[i][j];

}

}

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

{

for (int j=0;j<m;j++)

{

Max[i][j] = Need[i][j] + Allocation[i][j];

}

}

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*程序开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

display();

}

bool isSystemSafe()

{

int work[MaxNumber];

for (int i=0;i<m;i++) //m是资源个数A,B,C

{

work[i] = Available[i];

}

for (int i=0;i<n;i++) //n是进程个数

{

Finish[i] = false;

SafeOrder[i] = -1; //初始化安全序列

}

int FinishNumebr = 0;

int isSafe;

int i =0,j;

while(i<n)

{

isSafe = 0;

for(j = 0;j<m;j++)

{

if (Finish[i]==false && Need[i][j]<=work[j])

{

isSafe++;

}

else

break;

}

if (isSafe == m) //当且仅当进程对应的所有资源的数量都满足的时候才成立

{

Finish[i] = true;

SafeOrder[FinishNumebr] = i;

FinishNumebr++;

for (j = 0;j<m;j++)

{

work[j] += Allocation[i][j];

}

i=0; //找到满足条件的进程后，从头开始再进行寻找

}

else

i++;

if (FinishNumebr==n)

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Safe!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"对应的安全序列为："<<endl;

cout<<"P"<<SafeOrder[0];

for (int i=1;i<n;i++)

{

cout<<"-->"<<"P"<<SafeOrder[i] ;

}

cout<<endl;

return true;

}

}

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*unSafe!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

return false;

}

void bankerAlgorithm()

{

int chooseProcess;

char isContinue;

while(true)

{

bool isRequestNeedOK = true;

bool isRequestAvailableOK = true;

cout<<"请输入要申请资源的进程号（注意：第一个进程为0号，第二个进程为1号，以此类推！）"<<endl<<"chooseProcess=";

cin>>chooseProcess;

cout<<"请输入进程所请求的各类资源的数量：(A B C)"<<endl;

for (int i=0;i<m;i++)

{

cin>>Request[i];

}

//输入错误判断

for (int i=0;i<m;i++)

{

if (Request[i]>Need[chooseProcess][i])

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*当前运行结果\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"您输入的请求进程所对应的资源数量超过最大需求量，请重新输入！"<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

isRequestNeedOK = false;

continue;

}

if (Request[i]>Available[i])

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*当前运行结果\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"您输入的请求进程的资源数量超过系统所供给的最大资源数量pi必须等待，请重新输入！"<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

isRequestAvailableOK = false;

continue;

}

}

for (int j = 0;j<m;j++)

{

Available[j] -=Request[j];

Allocation[chooseProcess][j] += Request[j];

Need[chooseProcess][j] -= Request[j];

}

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

if (!isSystemSafe()) //如何不满足系统安全性算法，将本次试探作废，恢复到原来的值

{

for (int j = 0;j<m;j++)

{

Available[j] +=Request[j];

Allocation[chooseProcess][j] -= Request[j];

Need[chooseProcess][j] += Request[j];

}

if (isRequestAvailableOK&&isRequestNeedOK)

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*当前运行结果\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"当前可利用资源已经不能满足任何进程的需求，故系统进入不安全状态，系统不分配资源！"<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

}

}

cout<<"尝试根据p(i)发出的请求向量后的资源情况："<<endl;

display();

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"是否继续输入请求变量request进行测试，是（Y）,否（N）"<<endl;

cout<<"isContinue = ";

cin>>isContinue;

if (isContinue=='Y'||isContinue=='y')

{

//input();//实现多次请求（保留上一次请求的状态）

continue;

}

else if (isContinue=='N'||isContinue=='n')

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*程序结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

break;

}

}

}

int main(int argc, char\* argv[])

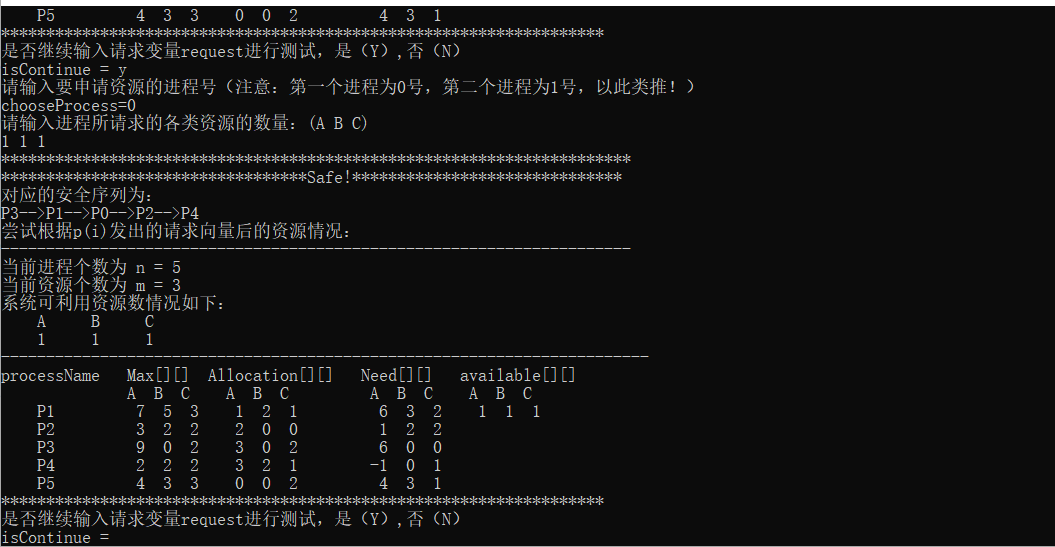
{

input();

bankerAlgorithm();

return 0;

}



【实验总结】

由银行家算法我们可以清楚的了解资源如何分配给进程的过程，构造一个算法确保系统决不进入不安全状态，在进行资源请求算法之前要先对系统进行安全性检查，如何将Available、Max、Allocation、Need矩阵进行转化等等这些都是在进行程序实现之后我们更容易理解与掌握，此外，通过实现银行家算法，我们也对自己的c++编程能力有了提高。