 ****

**计算机科学与技术学院**

**《操作系统》实验报告**

（ 2019 / 2020 学年 第 二 学期）

**学生姓名： XXX**

**学生专业： 计算机科学与技术**

**学生班级： 计算机172002**

**学生学号： 20172XXXXXX**

2020 年　6 月　3 日

# 实验四 银行家算法模拟

实验学时：2

实验类型：验证

实验要求：必修

**一、实验目的**

（1）进一步理解利用银行家算法避免死锁的问题；

（2）在了解和掌握银行家算法的基础上，编制银行家算法通用程序，将调试结果显示在计算机屏幕上，再检测和笔算的一致性。

（3）理解和掌握安全序列、安全性算法

**二、实验要求**

（1）了解和理解死锁；

（2）理解利用银行家算法避免死锁的原理；

（3）使用某种编程语言，编制银行家算法通用程序。

（4）撰写上机报告。

**三、实验内容**

(一)安全状态

指系统能按照某种顺序如<P1, P2, …, Pn>(称为<P1, P2, …, Pn>序列为安全序列)，为每个进程分配所需的资源，直至最大需求，使得每个进程都能顺利完成。

(二)银行家算法

假设在进程并发执行时进程i提出请求j类资源k个后，表示为Requesti[j]=k。系统按下述步骤进行安全检查：

（1）如果Requesti≤Needi则继续以下检查，否则显示需求申请超出最大需求值的错误。

（2）如果Requesti≤Available则继续以下检查，否则显示系统无足够资源，Pi阻塞等待。

（3）系统试探着把资源分配给进程Pi，并修改下面数据结构中的数值：

Available［j］=Available［j］-Requesti［j］;

Allocation［i,j］=Allocation［i,j］+Requesti［j］;

Need［i,j］=Need［i,j］-Requesti［j］;

（4）系统执行安全性算法，检查此次资源分配后，系统是否处于安全状态。若安全，才正式将资源分配给进程Pi，以完成本次分配；否则，将本次的试探分配作废，恢复原来的资源分配状态，让进程Pi等待。

（三）安全性算法

(1) 设置两个向量：

① 工作向量Work: 它表示系统可提供给进程继续运行所需的各类资源数目，它含有*m*个元素，在执行安全算法开始时，Work∶=Available;

② Finish: 它表示系统是否有足够的资源分配给进程，使之运行完成。开始时先做Finish［i］∶=false; 当有足够资源分配给进程时， 再令Finish［i］∶=true。

(2) 从进程集合中找到一个能满足下述条件的进程：

① Finish［i］=false;

② Need［i,j］≤Work［j］； 若找到， 执行步骤(3)， 否则，执行步骤(4)。

(3) 当进程Pi获得资源后，可顺利执行，直至完成，并释放出分配给它的资源，故应执行：

Work［j］=Work［i］+Allocation［i,j］;

Finish［i］=true;

go to step 2;

（4）如果所有进程的Finish［i］=true都满足，则表示系统处于安全状态；否则，系统处于不安全状态。

**四、实验算法**

（1）参考图1-1所示流程图编写安全性算法。

（2）编写统一的输出格式。

每次提出申请之后输出申请成功与否的结果。如果成功还需要输出变化前后的各种数据，并且输出安全序列。

（3）参考图1-2所示流程图编写银行家算法。

（4）编写主函数来循环调用银行家算法。

**五、思考题**

（1）在编程中遇到了哪些问题？你是如何解决的？

（2）在安全性算法中，为什么不用变量Available，而又定义一个临时变量work？

结束

否

是

申请失败。

以上分配作废，恢复原来的分配状态：

Available[j] = Available[j] + Request[i][j]

Allocation[i][j]= Allocation[i][j]－Request[i][j]

Need[i][j] = Need[i][j]+Request[i][j]

N

Y

N

Y

Request[i][j]> Need[i][j]

出错返回：return(error)

Request[i][j]> Available[j]

出错返回：(进程阻塞)

return(error)

Available[j] = Available[j] – Request[i][j]

Allocation[i][j]= Allocation[i][j] + Request[i][j]

Need[i][j] = Need[i][j] – Request[i][j]

假定分配：

输入初始参数（资源分配及请求情况）

开始

假定分配之后，系统安全吗？

申请成功。输出各种数据的变化

图1-1银行家算法流程图

1. **源程序清单：**

// 银行家算法.cpp : 定义控制台应用程序的入口点。

//

#include "stdafx.h"

#include <iostream>

#include<stdio.h>

#include<stdlib.h>

using namespace std;

const unsigned short c=3;// 资源类数

const unsigned short t=5;// 进程数

void print();// 用于打印输出表格的函数

void input();// 用于输入的函数

void tryfenpei(int i);// 试分配函数；

void refenpei(int i);// 恢复数据函数

void checksafe(int s);// 安全检测函数

int temp[t];

int work[c];

// 定义初始化数组

int need[t][c],request[c],available[c];

int max1 [t][c]={3, 5, 7 ,9 ,11,6 ,8 ,2 ,9, 5,6 ,3 ,5 ,7 ,4};

int allocation[t][c]={1 ,2 ,5 ,4, 8,5 ,4, 1 ,8 ,3 ,3 ,2 ,4, 3, 1};

int total[c]={17,21,25};

int in;// 用户选择的进程号

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

{

int i;

char ch='Y';

int l=0,m=0,a;

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

{

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

need[i][j]=max1[i][j]-allocation[i][j];

}

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

{

a=0;

for(int l=0;l<t;l++)

a+=allocation[l][m];

available[m]=total[m]-a;

}

do {

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

{

cout<<"ok, 现在开始进入实验…… "<<endl;

cout<<" 请输入需要请求的进程号 (0-4):";

while(cin>>in)

{

if(!(0<=in&&in<=4))

{

cout<<" 这里没有该进程，请重新输入～ "<<endl;

}

else break;

}

cout<<" 您输入的是 "<<"p["<<in<<"]"<<" 进程 "<<endl;

cout<<" 该进程需求量为： ";

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

{

need[in][i]=max1[in][i]-allocation[in][i];

cout<<need[in][i]<<" ";}

cout<<endl;

cout<<endl;

cout<<" 请输入请求资源向量 :";// 输入格式为ｘ

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

{

while(cin>>request[i])

{

if(request[i]<0) cout<<"sorry, 输入的数字无效～ "<<endl;

else if(request[i]>need[in][i])

cout<<" 超出进程需求量～ "<<endl<<endl;

if(request[i]>available[i])

cout<<" 系统没有足够多的可用资源量满足进程需要～ "<<endl<<endl;

else break;

}

}

cout<<" 输入成功～，您输入的是： "<<request[0]<<" "

<<request[1]<<" "<<request[2];

cout<<" 等待已久的银行家算法开始执行～ "<<endl;

tryfenpei(in);// 分配函数

cout<<" 试分配完成～ "<<endl;

cout<<" 现在进入安全性检测…… "<<endl;

checksafe(in);// 安全性检测函数

cout<<" 您还想继续银行家算法的实验吗～？（ y －继续 n 终止） ";

}

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

{

cout<<" 感谢您的使用， Bye ～ "<<endl<<" 退出 ing …… "<<endl;

break;

}

else

cout<<" 输出无效！请重新输入 "<<endl;

}

while (cin>>ch);

return 0;

}

/\*------- 输出函数 -------\*/

void print()

{

int i,j;

cout<<" 更新数据中 ..."<<endl;

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

cout<<"|-------| 最大需求矩阵 | 已分配矩阵 -|- 需求矩阵 -| 可利用资源 -|"<<endl;

cout<<"| 资源 | Max | Allocation| Need | available |"<<endl;

cout<<"| | A B C | A B C | A B C | A B C |"<<endl;

cout<<"| 进程 | | | | |"<<endl;

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

for(i=0;i<5;i++)

{

cout<<"| p"<<i<<" | ";

for(j=0;j<3;j++) {

cout<<max1[i][j]<<" "; }

cout<<" | ";

for(j=0;j<3;j++) {

cout<<" "<<allocation[i][j]; }

cout<<" | ";

for(j=0;j<3;j++) {

cout<<" "<<need[i][j]; }

cout<<" |";

if(i==0) {

for(j=0;j<3;j++) {

cout<<" "<<available[j]; }

cout<<" |"; }

if(i>0)

{cout<<" |"; }

cout<<endl; }

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

}

/\*-------- 试分配函数 -------\*/

void tryfenpei(int i)

{

for(int f=0;f<c;f++)

{

available[f]=available[f]-request[f];

allocation[i][f]=allocation[i][f]+request[f];

need[i][f]=need[i][f]-request[f];

}

}

/\*-------- 恢复数据函数 -------\*/

void refenpei(int i)

{

for(int f=0;f<c;f++)

{

available[f]=available[f]+request[f];

allocation[i][f]=allocation[i][f]-request[f];

need[i][f]=need[i][f]+request[f];

}

}

int com(int \*p,int \*q)

{

int i;

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

if(p[i]>q[i])

return 0;

return 1;

}

/\*-------- 安全检测函数 ---------\*/

void checksafe(int s)

{

int flag,temp[t],i,j,l,k=0;

bool finish[t];

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

finish[i]=false;

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

work[j]=available[j];

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

cout<<"| resource |-Work+Allocation-|--Finish--|"<<endl;

cout<<"| | A B C | T/F |"<<endl;

cout<<"|programme | | |"<<endl;

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

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

{

l=0;

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

{

if(need[i][j]>work[j])

l=1;

break;

}

if(finish[i]==false&&l==0)

{

cout<<"| p"<<i<<" | ";

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

{

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

if(work[j]>9)

cout<<" "<<work[j]<<" ";

else

cout<<" "<<work[j]<<" ";

}

cout<<" ";

cout<<"|";

cout<<" ";

finish[i]=true;

cout<<"true ";

cout<<"|";

temp[k]=i;//cout<<'temp="<<temp[k]<<endl;

k++;

i=-1; // 从用户选择的进程开始对每个进程都要检测

cout<<endl;

}

}

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

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

{

if(finish[i]==false)

flag=1;

if(flag==1)

{

cout<<" 系统不安全！本次资源申请不成功感！ "<<endl;

cout<<" 正在恢复原来的数据 ..."<<endl;

refenpei(in);

cout<<" 恢复数据成功 ! 正在打印输出 ..."<<endl;

print();

}

else

{

cout<<" 找到一个安全系列： ";

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

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

cout<<endl<<" 已通过安全性测试 !"<<endl<<endl<<endl;

cout<<" 开始给第 "<<"p]"<<in<<"]"<<" 进程分配资源 ..."<<endl;

cout<<" 分配完成 ! 打印输出 ..."<<endl<<endl;

print();

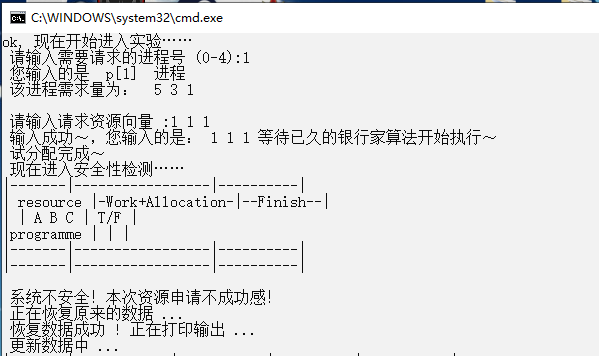
cout<<endl;

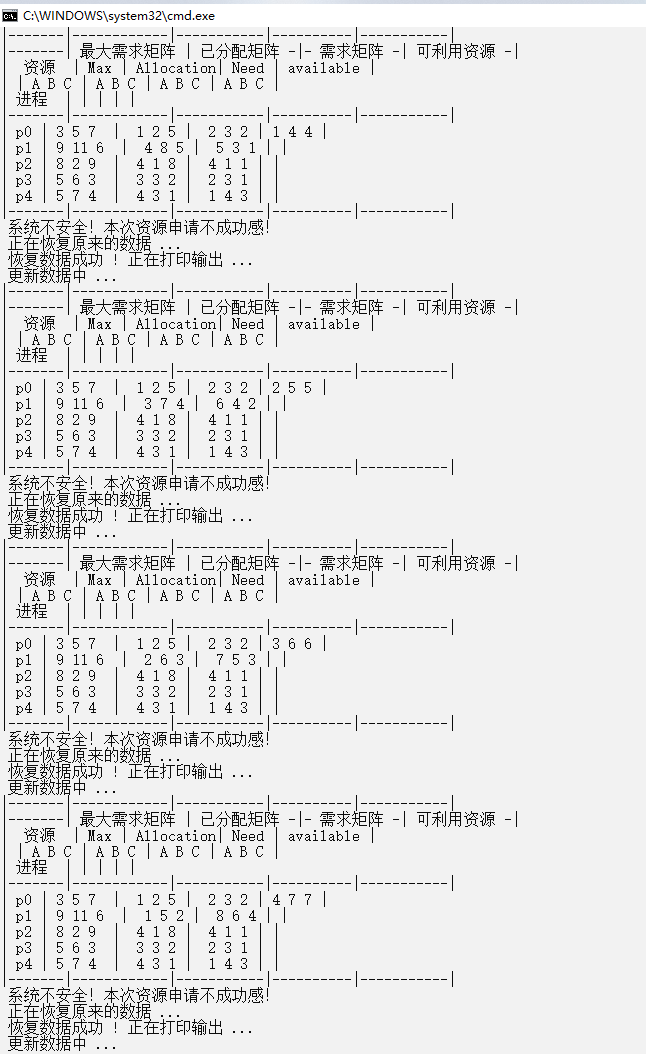
}

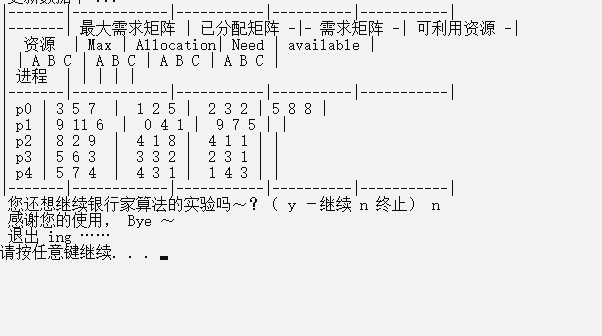
}

}

**七、测试结果**

****

****

****

**八、实验总结**

经过这次做实验，我体会颇多，学到了很多东西。我加强了对C++程序设计这门课程的认识，进一步理解利用银行家算法避免死锁的问题；理解和掌握了安全序列、安全性算法。这些都使我对计算机语言的学习有了更深入的认识。总之，通过这次课程设计，我收获颇丰，相信会为自己以后的学习带来很大的好处。锻炼了我们挑战难题，学会用已掌握的知识去解决具体问题的能力，进一.步培养了独立思考问题和解决问题的能力。当然，同学的帮助也是不可忽视的，他们给了我许多提示和帮助，教会了我编译复杂程序的方法。

通过本次实验，我进一步学习和掌握了对程序的设计和编写，从中体会到了面向对象程序设计的方便和巧妙。懂得了在进行编写一个程序之前，要有明确的目标和整体的设计思想。另外某些具体的细节内容也是相当的重要。这些宝贵的编程思想和从中摸索到的经验都是在编程的过程中获得的宝贵财富。这些经验对我以后的编程会有很大的帮助的，我要好好利用。