**广州大学学生实验报告**

**开课学院及实验室：**计算机科学与工程实验室 **2019年10月31日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | **计算机科学与教育软件学院** | **年级/专业/班** | **计科173** | **姓名** | 谢绍波 | **学号** | 1706100109 |
| **实验课程名称** | **操作系统实验** | | | | | **成绩** |  |
| **实验项目名称** | 实验二 银行家算法 | | | | | **指导老师** | 张汛涞 |

**一、实验目的及要求**

**二、实验设备与平台**

1. 实验设备：计算机；

2. 平台：Ubuntu虚拟机

**三、实验内容与步骤**

　一、实验指导

　　要解释银行家算法，必须先解释操作系统安全状态和不安全状态。

　　安全状态：如果存在一个由系统中所有进程构成的安全序列P1，…，Pn，则系统处于安全状态。安全状态一定是没有死锁发生。

　　不安全状态:不存在一个安全序列。不安全状态不一定导致死锁。

　　那么什么是安全序列呢？

　　安全序列：一个进程序列{P1，…，Pn}是安全的，如果对于每一个进程Pi(1≤i≤n），它以后尚需要的资源量不超过系统当前剩余资源量与所有进程Pj (j < i )当前占有资源量之和。

　　银行家算法：

　　按银行家制定的规则为进程分配资源，可以使系统保持在安全状态，具体方法如下：

·进程首次申请资源的分配

如果系统现存资源可以满足该进程的最大需求量，则按当前的申请量分配资源，否则推迟分配。

·进程在执行中继续申请资源的分配

若该进程已占用的资源与本次申请的资源之和不超过对资源的最大需求量，且现存资源能满足该进程尚需的最大资源量，则按当前申请量分配资源，否则推迟分配。

·至少一个进程能完成

在任何时刻保证至少有一个进程能得到所需的全部资源而执行到结束。

银行家算法通过动态地检测系统中资源分配情况和进程对资源的需求情况来决定如何分配资源，并能在确保系统处于安全状态时才把资源分配给申请者，从而避免系统发生死锁。

二、实验目的

掌握银行家算法思想，并能编程实现。

三、实验内容和要求

1、在Linux环境下编译运行程序；  
2、按照教材3.7.2节的算法编写；  
3、输入数据从文本文件中读出，不得从键盘录入；  
4、主要数据结构的变量名和教材中的一致，包括Available、Max、Allocation、Need、Request、Work、Finish。并作以下改动：在变量名后加下划杠和姓名缩写。例如，张三丰同学的变量名为： Available\_zsf， Max\_zsf， Allocation \_zsf……  
5、程序可支持不同个数的进程和不同个数的资源；  
6、验证教材中的“银行家算法示例”中的例子（包括可成功分配、不可分配）。

四、实验程序

#include <iostream>

#include <fstream>

#define PROGRESS 5

#define RESOURCE  3

using namespace std;

int Available\_xsb[RESOURCE];

int sign[PROGRESS];

int work[PROGRESS][RESOURCE];

int work\_Allocation\_xsb[PROGRESS][RESOURCE];

int Max\_xsb[PROGRESS][RESOURCE];

int Allocation\_xsb[PROGRESS][RESOURCE];

int Need\_xsb[PROGRESS][RESOURCE];

void init(){

        ifstream inFile1("output.txt",ios::in);

        if(!inFile1)

              cerr<<"File open error."<<endl;

        else {

                char c;

                while((c=inFile1.get())!=EOF)

                        cout<<c;

                cout<<endl;

                inFile1.close();

        }

        ifstream inFile2("blime.txt",ios::in);

        if(!inFile2)

              cerr<<"File open error."<<endl;

        else{

                int data;

                for(int j = 0;j < PROGRESS;j++) {

                for(int i = 0;i < RESOURCE;i++) {

                    inFile2>>data;

                    Max\_xsb[j][i]=data;

                }

                for(int i = 0;i < RESOURCE;i++) {

                    inFile2>>data;

                    Allocation\_xsb[j][i]=data;

                }

                for(int i = 0;i < RESOURCE;i++) {

                    inFile2>>data;

                    Need\_xsb[j][i]=data;

                }

                if(j==0) {

                    for(int i = 0;i < RESOURCE;i++) {

                        inFile2>>data;

                        Available\_xsb[i]=data;

                    }

                }

               }

                inFile2.close();

        }

}

int Safe\_inspection(int Available\_xsb[],int Need\_xsb[][RESOURCE],int Allocation\_xsb[][RESOURCE]) {

        int i=0,j=0,m=0,n=0;

        int Work[RESOURCE],Finish[PROGRESS] = {0,0,0,0,0};

        for(int r = 0;r < RESOURCE;r++)

                Work[r] = Available\_xsb[r];

        while(i < PROGRESS) {

                if(Finish[i] == 0){

                        while(j < RESOURCE && Need\_xsb[i][j] <= Work[j] )

                                j++;

                        if(j == RESOURCE) {

                                for(int k = 0;k < RESOURCE;k++){

                                        work[i][k] = Work[k];

                                        Work[k] = Work[k]+Allocation\_xsb[i][k];

                                        work\_Allocation\_xsb[i][k] = Work[k];

                                }

                                Finish[i]=1;

                                sign[m]=i;

                                i=-1;m++;

                        }

                }

                j=0;i++;

        }

        for(int p = 0;p < PROGRESS;p++){

               if(Finish[p] == 1)

                        n++;

        }

        return n;

}

int bank\_xsb (int i,int Request[][RESOURCE]){

        for(int m = 0;m < RESOURCE;m++) {

                if(Request[i][m] > Need\_xsb[i][m]){

                        cout<<"request resource over Max!"<<endl;

                        return 0;

                } else if(Request[i][m] > Available\_xsb[m]) {

                        cout<<"resource insufficient，p["<<i<<"] need wait!"<<endl;

                        return 0;

                }

        }

        for(int j = 0;j < RESOURCE;j++) {

                Available\_xsb[j] = Available\_xsb[j] - Request[i][j];

                Allocation\_xsb[i][j] = Allocation\_xsb[i][j] + Request[i][j];

                Need\_xsb[i][j] = Need\_xsb[i][j] - Request[i][j];

        }

        int n = Safe\_inspection(Available\_xsb,Need\_xsb,Allocation\_xsb);

        cout<<endl;

        if(n == PROGRESS) {

                cout<<"safe state，allocate resource to "<<"P["<<i<<"]!"<<endl;

        }else {

                cout<<"unsafe state，cannot allocate resource to "<<"P["<<i<<"]!"<<endl;

        }

        return n;

}

void Print(int result) {

        if(result == PROGRESS) {

                cout<<" progress\\Resources"<<" Work(A B C)"<<" Need\_xsb(A B C)"

                    <<" Allocation\_xsb(A B C)"<<" Work+Available\_xsb(A B C)"<<" Finish";

                cout<<endl;

                for(int i = 0;i < PROGRESS;i++) {

                        cout<<"    "<<"P["<<sign[i]<<"]  "<<'\t';

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

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

                        cout<<'\t'<<'\t';

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

                               cout<<Need\_xsb[sign[i]][j]<<" ";

                        cout<<'\t'<<'\t';

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

                               cout<<Allocation\_xsb[sign[i]][j]<<" ";

                        cout<<'\t'<<'\t';

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

                               cout<<work\_Allocation\_xsb[sign[i]][j]<<" ";

                        cout<<'\t'<<'\t';

                        cout<<"true"<<endl;

                }

                cout<<endl<<"Find safe sequence｛P["<<sign[0]<<"]";

                for(int m = 1;m < PROGRESS;m++){

                        cout<<", P["<<sign[m]<<"]";

                }

                cout<<"}"<<endl;

        } else {

                cout<<"   progress\\Resources "<<"  Allocation\_xsb(A B C)"<<"   Need\_xsb(A B C)"<<"   Available\_xsb(A B C)";

                cout<<endl;

                for(int k = 0;k < 5;k++){

                        cout<<'\t'<<"P["<<k<<"]"<<'\t'<<'\t';

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

                                cout<<Allocation\_xsb[k][j]<<" ";

                        cout<<'\t'<<'\t';

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

                                cout<<Need\_xsb[k][j]<<" ";

                        cout<<'\t'<<'\t';

                        if(k == 0) {

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

                                cout<<Available\_xsb[j]<<" ";

                        }

                        cout<<endl;

                }

        }

}

int main()

{

        puts("计科173谢绍波");

        init();

        int i,N=0;

        int Request[PROGRESS][RESOURCE];

        while(N!=-1) {

                cout<<endl<<"Please input request resource Request[PROGRESS i][resource j]:"<<endl;

                cout<<"PROGRESS i = ";

                cin>>i;

                cout<<"Resource (A B C) =  ";

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

                        cin>>Request[i][m];

                cout<<endl;

                int result = bank\_xsb(i,Request);

                cout<<endl<<"resource allocation table ："<<endl;

                Print(result);

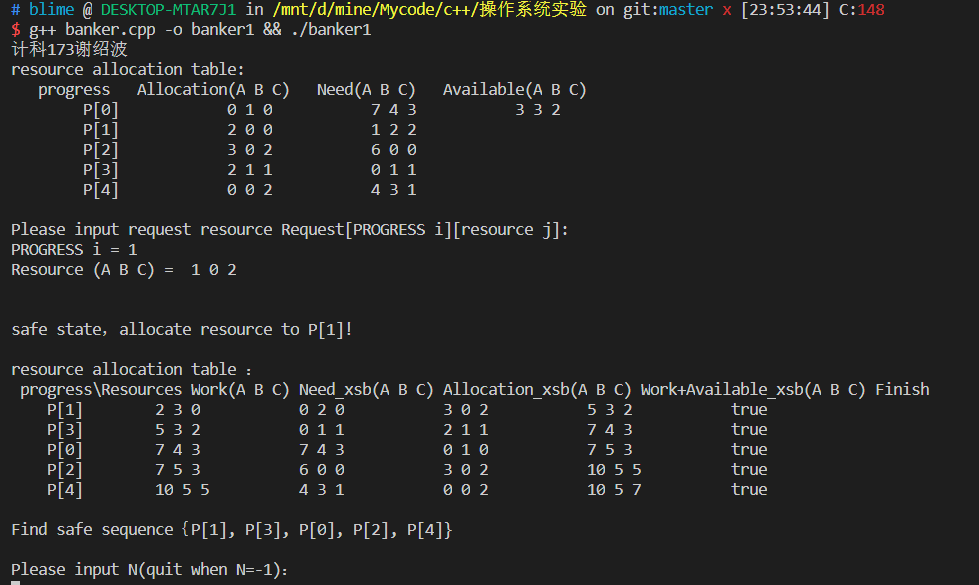
                cout<<endl<<"Please input N(quit when N=-1)："<<endl;

                cin>>N;

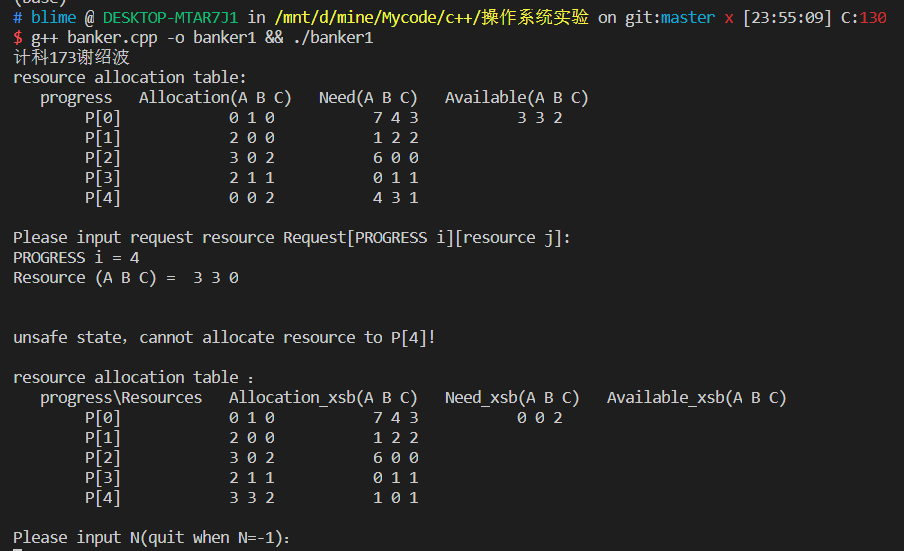
        }

}

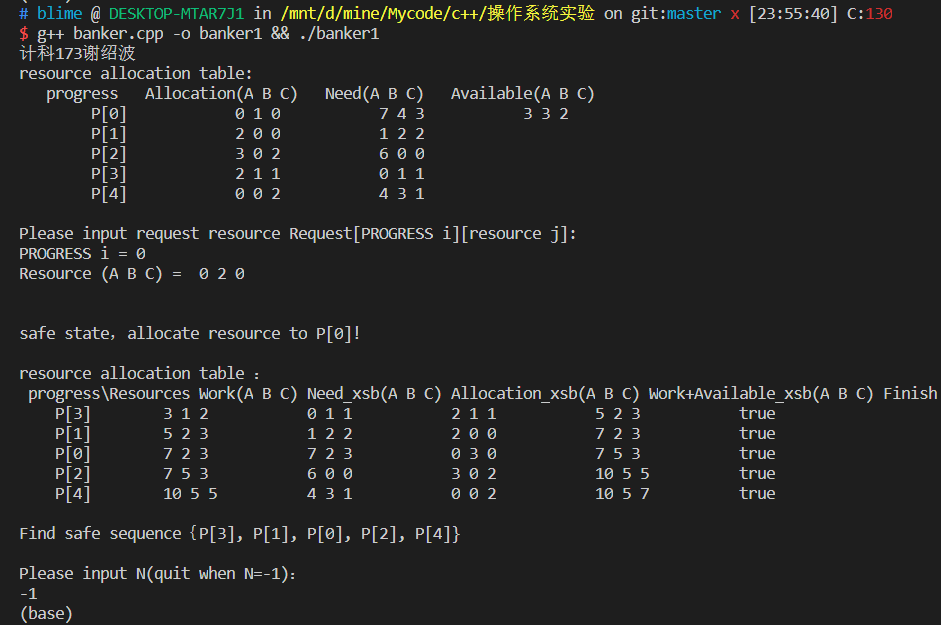
五、验证数据和运行结果



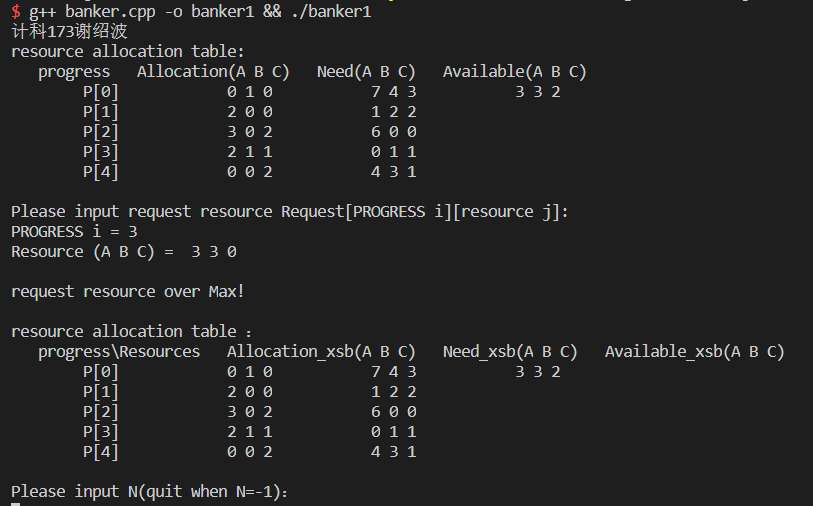
找到安全序列



不安全，不分配



找到安全序列



超过最大资源数

六、思考和分析

经过本次实验，认真学习了银行家算法。

首先是银行家算法中的进程：

包含进程Pi的需求资源数量（也是最大需求资源数量，MAX）

已分配给该进程的资源A（Allocation）

还需要的资源数量N（Need=M-A）

Available为空闲资源数量，即资源池（注意：资源池的剩余资源数量+已分配给所有进程的资源数量=系统中的资源总量）

假设资源P1申请资源，银行家算法先试探的分配给它（当然先要看看当前资源池中的资源数量够不够），若申请的资源数量小于等于Available，然后接着判断分配给P1后剩余的资源，能不能使进程队列的某个进程执行完毕，若没有进程可执行完毕，则系统处于不安全状态（即此时没有一个进程能够完成并释放资源，随时间推移，系统终将处于死锁状态）。

若有进程可执行完毕，则假设回收已分配给它的资源（剩余资源数量增加），把这个进程标记为可完成，并继续判断队列中的其它进程，若所有进程都可执行完毕，则系统处于安全状态，并根据可完成进程的分配顺序生成安全序列（如{P0，P3，P2，P1}表示将申请后的剩余资源Work先分配给P0–>回收（Work+已分配给P0的A0=Work）–>分配给P3–>回收（Work+A3=Work）–>分配给P2–>······满足所有进程）。如此就可避免系统存在潜在死锁的风险。