**操作系统课程实验指导书**

**信息工程学院**

**1、银行家算法的设计与实现**

对I/O系统中死锁问题求解的主要方法是银行家算法，单种资源的银行家算法和多种资源的银行家算法的解决思路一致，要求设计实现多种银行家算法，并要求所涉及的模型最少能够满足如下要求：

1. 程序能够根据进程的请求进行判断，给出系统是否安全的的提示，如果安全，要求能够显示一组进程执行的安全序列；
2. 能够根据需要，显示当前系统中各种资源的分配情况；

**算法实现思路：**

1）各种资源总数向量resource  
是一个含有m个元素的数组，其中的每一个元素代表一类系统可提供的的资源数目总数。如果resource[j]=K，则表示系统中现有Rj类资源K个。  
2）最大需求矩阵Claim  
这是一个n×m的矩阵，它定义了系统中n个进程中的每一个进程对m类资源的最大需求。如果Claim[i,j]=K，则表示进程i需要Rj类资源的最大数目为K。  
3）分配矩阵allocation  
这也是一个n×m的矩阵，它定义了系统中每一类资源当前已分配给每一进程的资源数。如果allocation[i,j]=K，则表示进程i当前已分得Rj类资源的 数目为K。  
4）还可使用的资源数available向量  
这是一个含有m个元素的数组，每个元素代表一类资源还能够使用的数目。  
**算法的执行流程很简单：**  
（1）先用resource[i]-sum[i](各个进程占用某个资源的总数)求出available[i]数组  
（2）然后遍历n\*m的矩阵，只要有一个进程对各类资源的需求均小于available数组，就将其认为是安全的，并且结束后将其占用的资源释放，即available[j]+=alloc[i][j];

（3）相反，在上一个步骤中，如果找不到这样的一个序列，我们就认为是不安全序列，不允许分配，程序结束

代码：

#include <iostream>

#include <string>

#include <cstring>

#include <vector>

using namespace std;

#define M 4 //一共有M种资源

#define N 3 //一共有N个进程

int resource[M]; //各种资源的总数 如果resource[i]=K，则表示系统中现有Ri类资源K个

int available[M]; //可获得的资源

int claim[N][M]; //每个进程对资源的最大需求

int alloc[N][M]; //每个进程已经分配的资源

int vis[N]; //

vector<int> ans; //

int main()

{

cout<<"请输入每个进程对资源的最大值矩阵"<<endl;

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

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

cin>>claim[i][j];

cout<<"请输入每个进程已经分配的各个资源"<<endl;

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

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

cin>>alloc[i][j];

cout<<"请输入各个资源的总数"<<endl;

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

cin>>resource[i];

int sum[M];

memset(sum,0,sizeof(sum));

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

{

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

sum[j]+=alloc[i][j];

}

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

available[i]=resource[i]-sum[i];

memset(vis,0,sizeof(vis));

ans.clear();

int is\_ok=0;

while(1)

{

if(ans.size()==N)

{

is\_ok=1;

break;

}

int index;

int flag=0;

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

{

int judge=1;

if(!vis[i])

{

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

{

if(available[j]<claim[i][j]-alloc[i][j])

{

judge=0;

break;

}

}

}

else

continue;

if(judge)

{

index=i;

flag=1;

break;

}

}

if(flag)

{

vis[index]=1;

ans.push\_back(index);

for(int k=0;k<M;k++)

available[k]+=alloc[index][k];

}

else

{

cout<<"出现死锁，无法分配!"<<endl;

return 0;

}

}

if(is\_ok)

{

cout<<"不会发生死锁，安全序列是："<<endl;

for(int i=0;i<ans.size();i++)

cout<<ans[i]<<" ";

cout<<endl;

}

return 0;

}