# 作业4

不限语言，编程实现银行家算法，实现书P121 的银行家算法之例。初始值为图3-15，输出检测T0状态是否安全的检查过程，在命令行显示每一步每一个进程的**Allocation**和此刻系统的可用资源Work的值，并给出安全序列

C++实现代码：

#include<iostream>

#define False 0

#define True 1

#define resourceNum 3

#define progressNum 5

using namespace std;

int Max[progressNum][resourceNum] = {

{

7,5,3},{

3,2,2},{

9,0,2},{

2,2,2},{

4,3,3}};

int Avaliable[resourceNum] = {

3,3,2};

char name[progressNum] = {

0,1,2,3,4};

int Allocation[progressNum][resourceNum] = {

{

0,1,0},{

2,0,0},{

3,0,2},{

2,1,1},{

0,0,2}};

int Need[progressNum][resourceNum] = {

{

7,4,3},{

1,2,2},{

6,0,0},{

0,1,1},{

4,3,1}};

int Request[resourceNum] = {

0};

int temp[progressNum]= {

0}; //记录安全序列

int Work[resourceNum] = {

0,0,0};

int Finsh[progressNum] = {

0};

/\* Max Need Allocation Avaliable P0 7 5 3 0 1 0 7 4 3 3 3 2 P1 3 2 2 2 0 0 1 2 2 P2 9 0 2 3 0 2 6 0 0 P3 2 2 2 2 1 1 0 1 1 P4 4 3 3 0 0 2 4 3 1 \*/

void showInfo(){

cout << "当前系统资源分配需求\t 剩余可分配资源为"<<endl;

for(int i = 0; i < resourceNum; i++) cout << Avaliable[i] << " ";

cout << "\n-------------------------------------------------" << endl;

cout << "name\tMax\tNeed\tAllocation\t" << endl;

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

cout << "P" << i << "\t";

for(int j = 0;j < resourceNum; j++) cout << Max[i][j] << " ";

cout << "\t";

for(int j = 0; j < resourceNum; j++) cout << Need[i][j] << " ";

cout <<"\t";

for(int j= 0; j < resourceNum; j++) cout << Allocation[i][j] << " ";

cout << "\n";

}

}

void ShowSafeInfo(int progress){

cout << "\n";

cout << "P" <<progress << "\t";

for(int j = 0;j < resourceNum; j++) cout << Max[progress][j] << " ";

cout << "\t";

for(int j = 0; j < resourceNum; j++) cout << Need[progress][j] << " ";

cout <<"\t";

for(int j= 0; j < resourceNum; j++) cout << Allocation[progress][j] << " ";

cout << "\t\t";

for(int j = 0; j < resourceNum; j++) cout << Work[j] << " ";

cout << "\n";

}

/\*检验当前进程能否分配\*/

bool CanAvaliable(int progress,int Avaliable[]){

if(Finsh[progress] == 1) return false;

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

if(Need[progress][i] > Avaliable[i]){

return false;

}

}

return true;

}

//进行银行家算法，进行资源分配

bool isSafe(){

Finsh[progressNum] = {

0};

int progress = 0; //记录操作的进程号

int finshProgress = 0;

int Serise = 0;

int flag = 0;

cout << "Name\tMax\tNeed\tAllocation\tAllocation+Work" << endl;

for(int i = 0; i < resourceNum; i++) Work[i] = Avaliable[i];

while(finshProgress != progressNum){

//如果可以将资源分配给该进程，则更改Work 并将Finsh 的值更改，记录安全序列

if(CanAvaliable(progress,Work)){

for(int i = 0; i < resourceNum; i++) Work[i] += Allocation[progress][i];

ShowSafeInfo(progress);

Finsh[progress] = 1;

temp[Serise++] = progress;

finshProgress++;

}

progress++;

if(progress >= progressNum){

progress = progress % progressNum;

if(flag == finshProgress) break; //检验当全部遍历第二遍之后完成的进程数没有改变，则表示只能进行finshprogress个进程的分配

else flag = finshProgress ;

}

}

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

if(Finsh[i] == 0) {

cout << "当前系统不安全" << endl;

return false;

}

}

return true;

}

int main(){

int p;

int Request[resourceNum] ;

showInfo();

//isSafe();

cout << "\n-------------------------------------------------" << endl;

cout << "请问是那个进程请求分配资源：" ;

cin >> p;

cout << "\n请问三个资源的分别请求的数量为：";

for(int i = 0; i < resourceNum; i++) cin >> Request[i];

//初步判断是否能够分配

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

if(Request[i] > Avaliable[i]) {

cout << "系统处于不安全状态，原因为：空闲资源量小于请求的资源数量" << endl;

break;

}

}

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

Avaliable[i] = Avaliable[i] - Request[i];

Allocation[p][i] += Request[i];

Need[p][i] = Need[p][i] - Request[i];

}

showInfo();

Finsh[resourceNum] = {

0};

cout << endl;

if(isSafe()){

cout << "\n系统处于安全状态，并且安全序列为：";

for(int i = 0; i < progressNum; i++) cout << temp[i] << " ";

}else{

cout << "系统处于不安全的状态，每一个进程不能都被分配";

}

}

运行结果：

