编号：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 | 总评 | 教师签名 |
| 成绩 |  |  |  |  |  |  |  |  |  |  |  |  |

武汉大学计算机学院

课程实验(设计)报告

专业(班)： 计算机科学与技术1班

学 号： 2015301040209

姓 名： 张相宾

课程名称： 操作系统设计

任课教师： 宋伟

2017年 11 月 22 日

**银行家算法实现**

1. **实习题目**
2. 实习内容

编写实现银行家算法，实现资源的安全分配。

1. 实习目的

通过本实验熟悉银行家算法，对预防死锁有更深刻的认识。

1. 实习题目

初始状态下，设置数据结构存储可利用资源向量（Available），最大需求矩阵（MAX），分配矩阵（Allocation），需求矩阵（Need），输入待分配进程队列和所需资源。

设计安全性算法，设置工作向量表示系统可提供进程继续运行的可利用资源数目。

如果进程队列可以顺利执行打印输出资源分配情况，如果进程队列不能顺利执行打印输出分配过程，提示出现死锁位置。

1. **实习内容及设计思想**
2. 设计思路

银行家算法的实现思想如下：

设Request是进程Pi的请求向量，Requesti[j]=k表示进程Pi请求分配Rj类资源k个，当Pi发出资源请求后，系统按下述步骤进行检查：

（1）如果Requesti<=Needi，则转向步骤（2）；否则，会出错，因为进程所需要的资源数目已超过它所宣布的最大值。

（2）如果Requesti<=Available，则转向步骤（3）；否则，表示系统中尚无足够的资源满足进程Pi的申请，Pi必须等待。

（3）系统试着把申请的资源分配给进程Pi，并修改下面数据结构中的数值

Available-= Requesti

Allocationi += Requesti;

Needi -= Requesti;

（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，则正式资源将资源分配给进程Pi，已完成本次分配；否则，将试分配作废，恢复原来的资源分配状态，让进程Pi等待。

系统所执行的安全性算法描述如下：

（1）设置两个向量

Work和Finish

（2）从进程集合中找到一个能满足下述条件的进程：

Finish[i]==false;

Needi<=Work;

若找到，则执行步骤（3）；否则，执行步骤（4）。

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

Work=Work+Alloacationi

Finish[i]=true;

然后转向第（2）步。

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

1. 主要数据结构

进程数量：m

资源种类：n

可利用资源向量：Available[n]

最大需求矩阵：Max[m][n]

分配矩阵：Allocation[m][n]

需求矩阵：Need[m][n]

请求矩阵：Request[m][n]

空闲资源向量：Work[m]

是否可能运行完成序列：Finish[m]

安全序列：Sequence[m]

1. 主要代码结构

安全性算法

bool Safe(int m,int n) {

int i, j;

int seq = 0;

int Work[MAXRESOURCE];

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

Work[i] = Available[i];

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

Finish[i] = false;

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

{

if (Finish[i] == true)

continue;

else {

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

if (Need[i][j] > Work[j])

break;

}

if (j == n) {

Finish[i] = true;

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

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

Sequence[seq] = i;

seq++;

i = -1;

}

else {

continue;

}

if (seq == m) {

cout << "安全序列为：" << endl;

for (i = 0; i < (seq-1); i++) {

cout << "P"<<Sequence[i]+1;

cout << "-->";

}

cout <<"P"<< Sequence[i]+1 << endl;

return true;

}

}

}

cout << "不安全" << endl;

return false;

}

1. **上机实习所用平台及相关软件**
2. **调试过程**
3. 测试数据设计

计算机操作系统P96

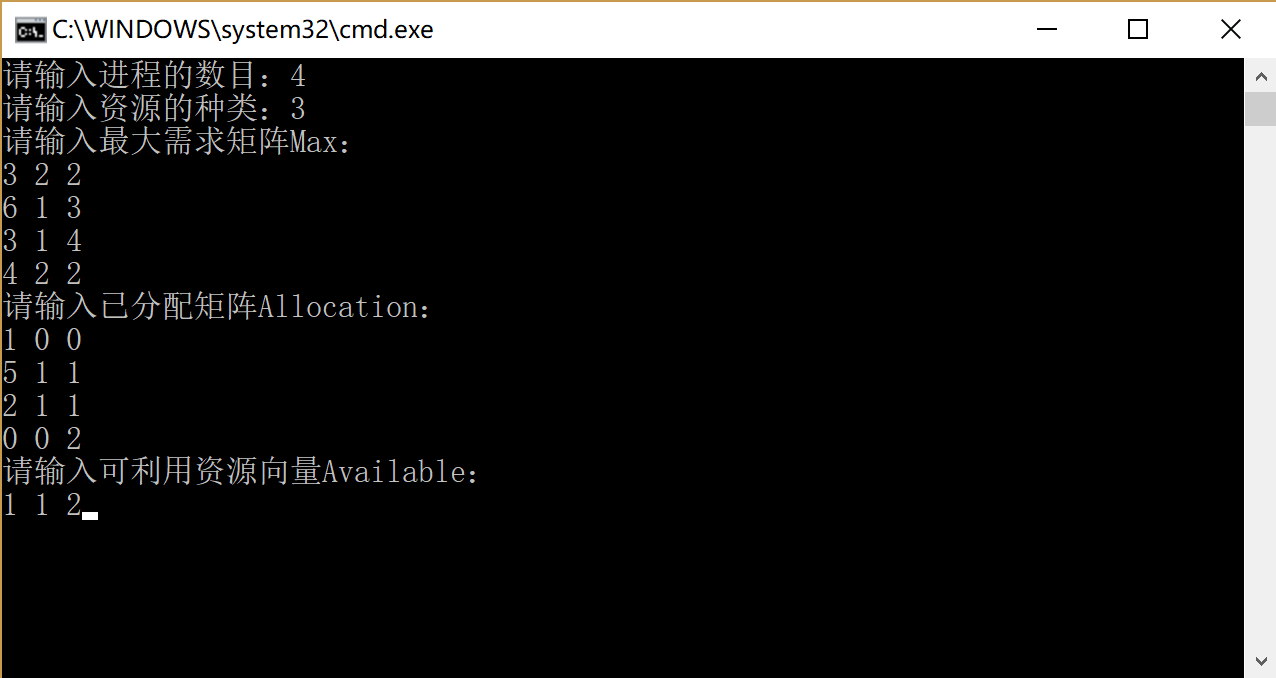
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Max | | | Allocation | | | Need | | | Available | | |
| R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
|  |  |  |  |  |  |  |  |  |  | 1 | 1 | 2 |
| P1 | 3 | 2 | 2 | 1 | 0 | 0 | 2 | 2 | 2 |  |  |  |
| P2 | 6 | 1 | 3 | 5 | 1 | 1 | 1 | 0 | 2 |  |  |  |
| P3 | 3 | 1 | 4 | 2 | 1 | 1 | 1 | 0 | 3 |  |  |  |
| P4 | 4 | 2 | 2 | 0 | 0 | 2 | 4 | 2 | 0 |  |  |  |

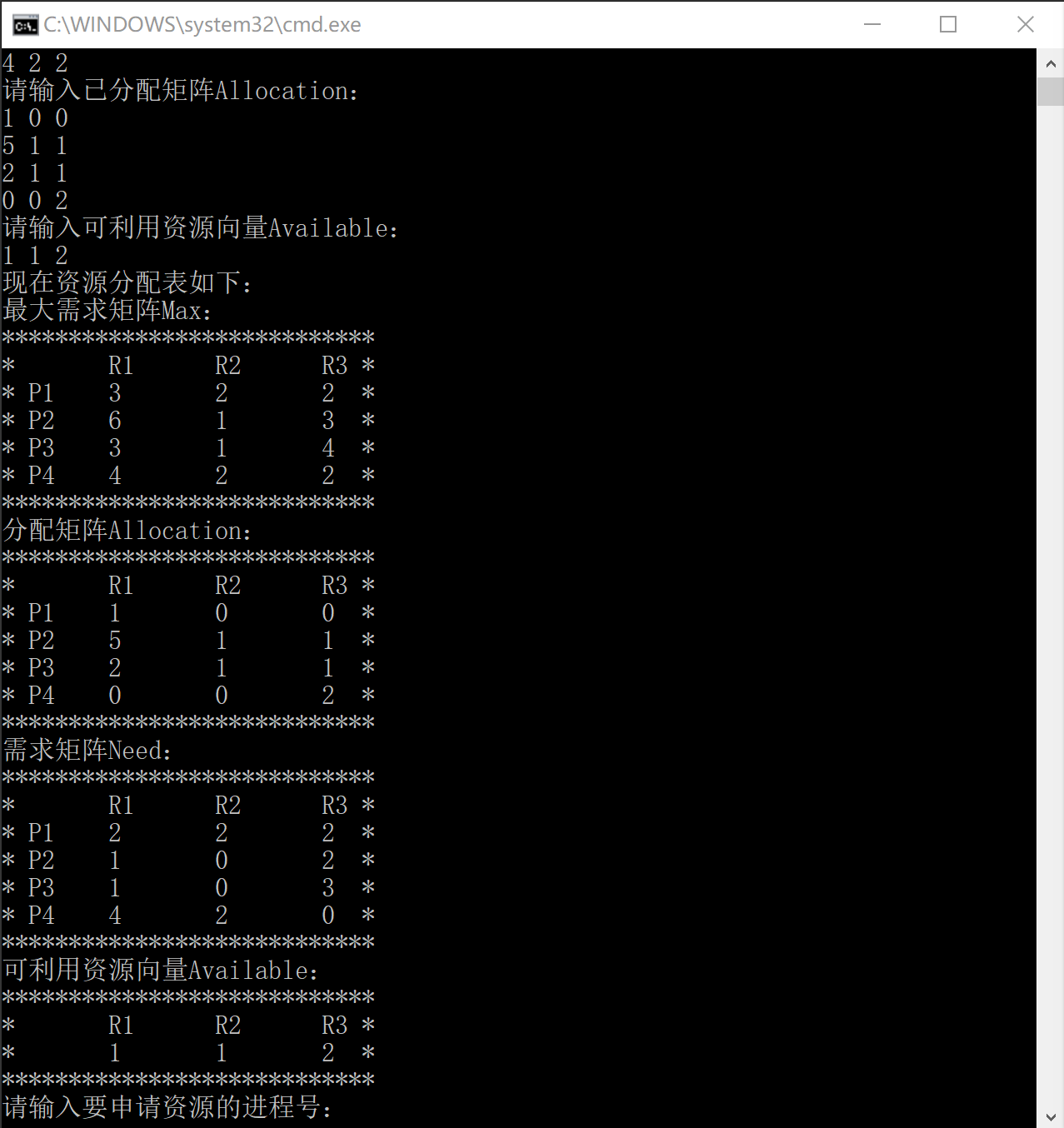
（1）P2请求资源（1,0,1）

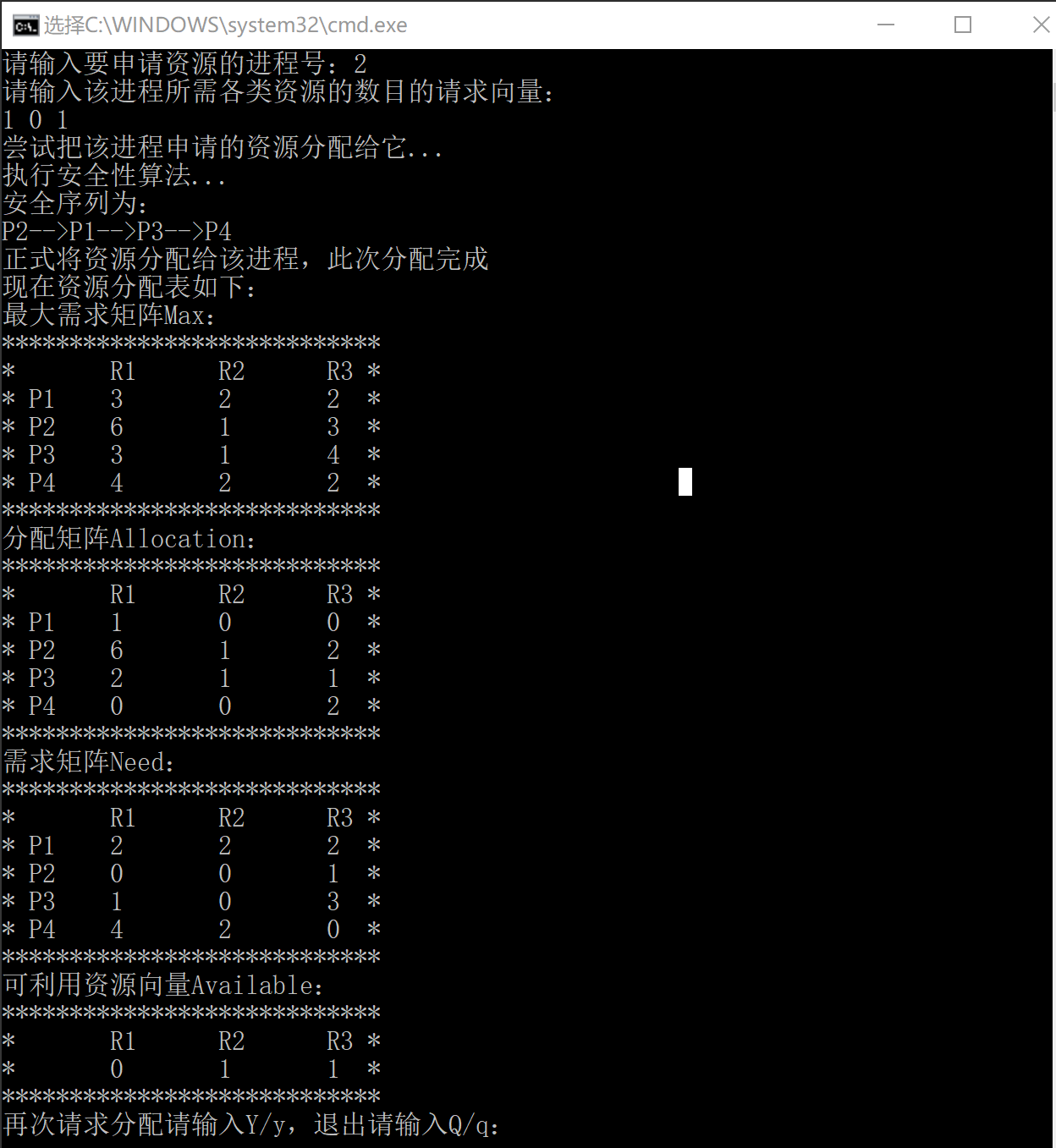
（2）P2请求资源后，P1请求资源（1,0,2）

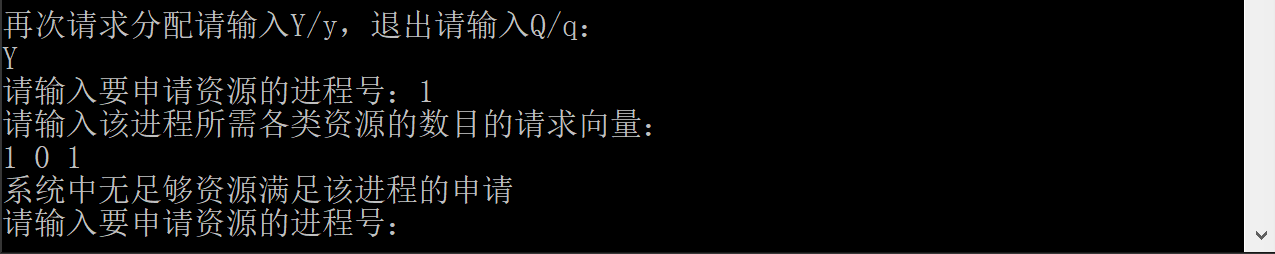
（3）之后，P3请求资源（0,0,1）

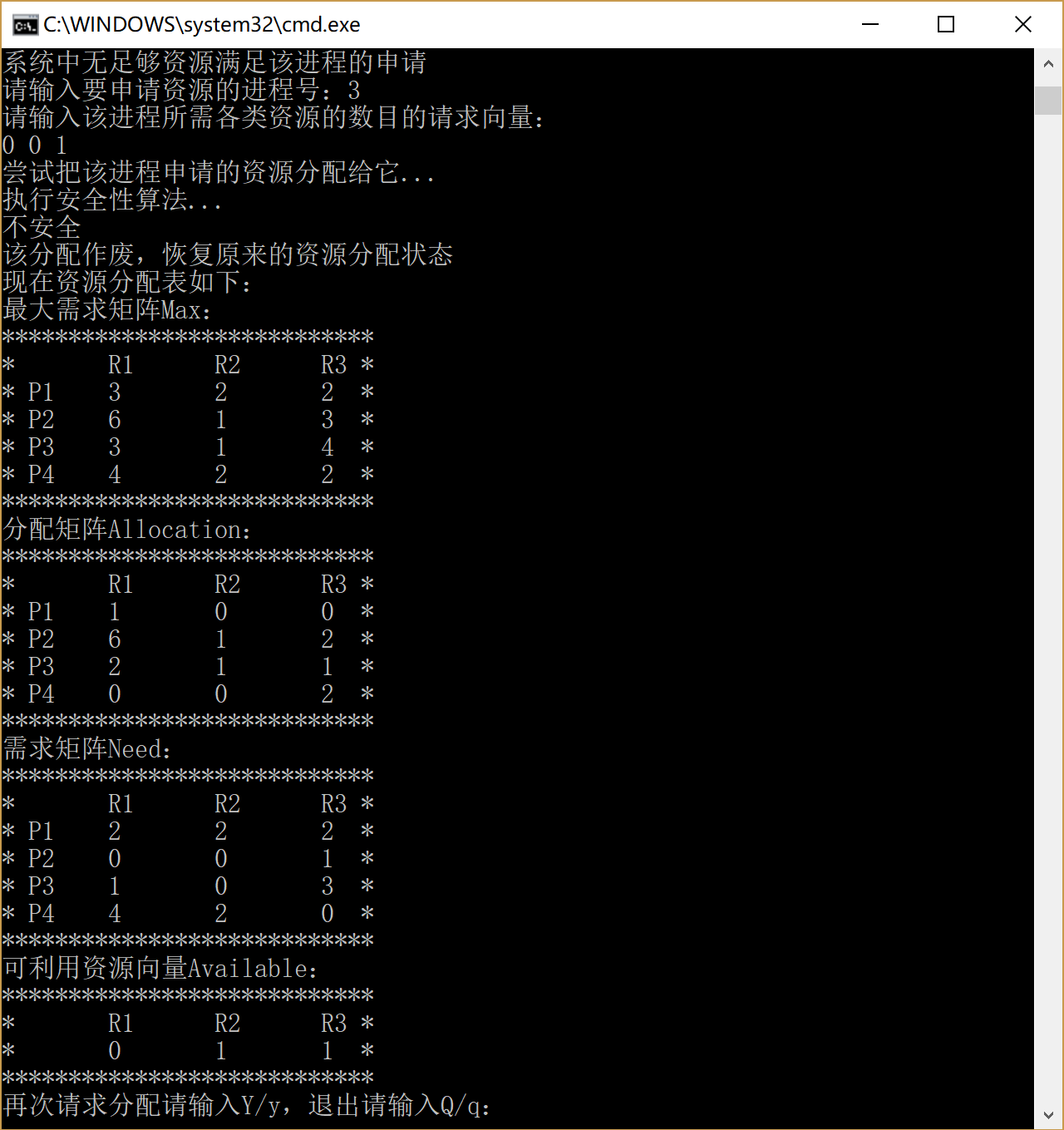
1. 测试结果分析











与预测结果相符

1. **总结**
2. 实习中遇到的问题及解决方法

（1）有的数组是二维矩阵，有的是一维数组，互相预算时尤其注意下标

（2）界面开始并不友好，通过灵活运用建表符是的界面易看

1. 实习体会及收获

对操作系统中的死锁问题有了更深理解，实践了银行家算法，对于系统安全性保证有了更深理解，可以自行设置资源种类和进程数目并自由分配，对多种情况有了更多了解。