

吉林大学

软件学院

操作系统课程设计实验报告

班级	6		
学号	55210629		
姓名	张耕儒		

题目：基于死锁避免动态策略的资源分配银行家算法的模拟实现

一、实验内容

(1)设计银行家算法所需的数据结构设置（界面）

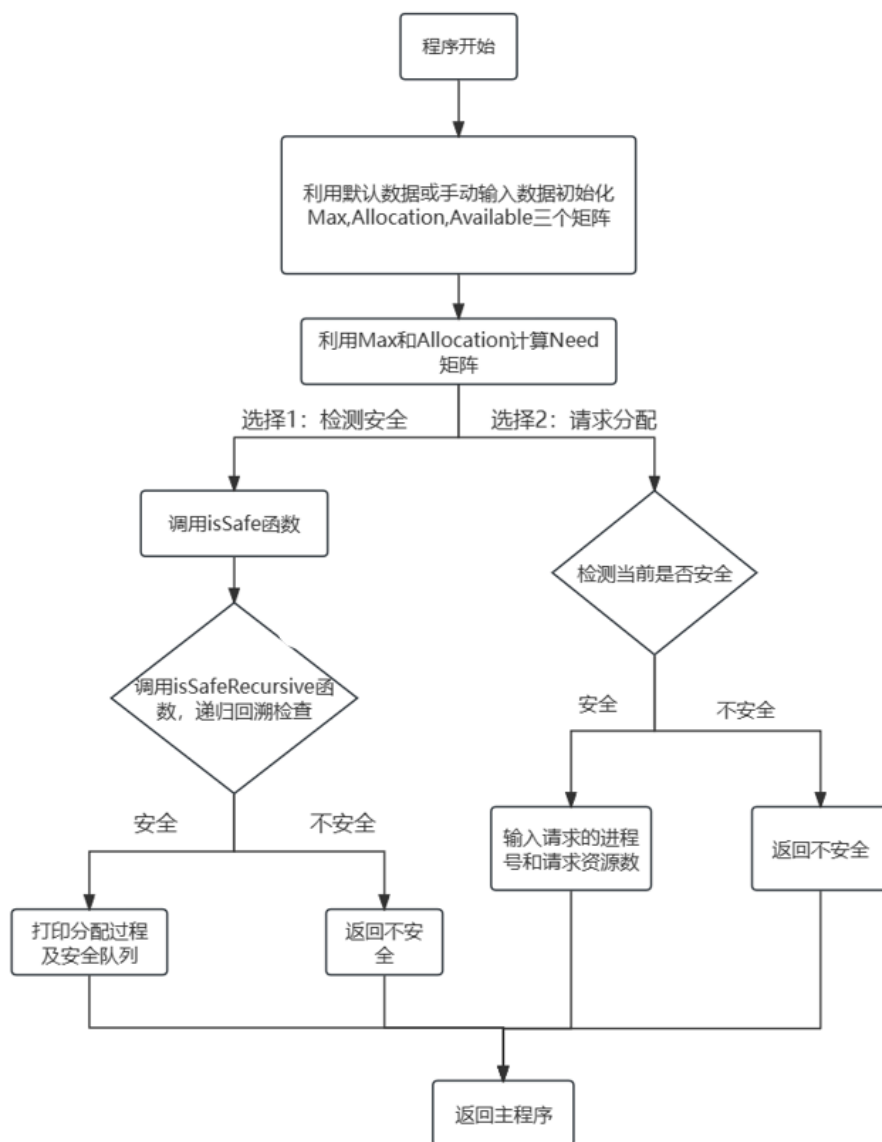
(2)运行结果（界面）

- ① 显示当前所设置的数据状态下，系统是否是安全状态的提示信息；
- ② 如果系统是安全状态，要求给出此状态下所有的安全进程序列；
- ③ 模拟在当前状态下，给出任意进程对某资源类的申请，并按银行家显示是否可接受的提示信息；如果该申请是可接受的，给出系统接受申请后的系统资源情况

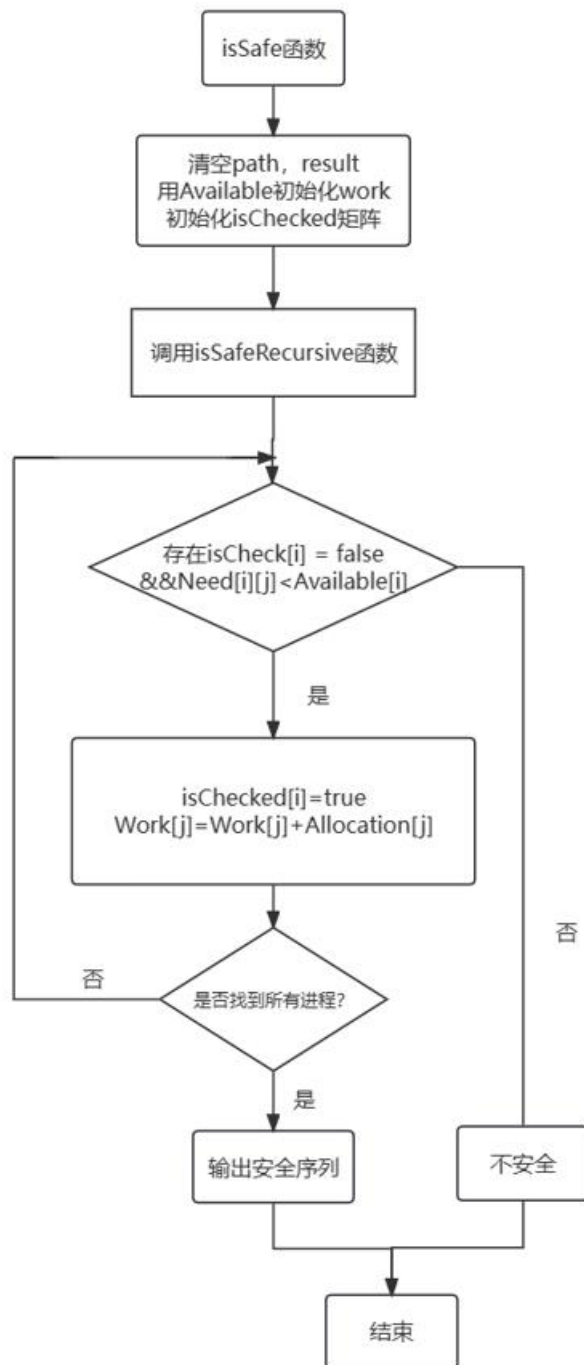
张耕儒完成（1），（2）①，②，③

二、实验设计

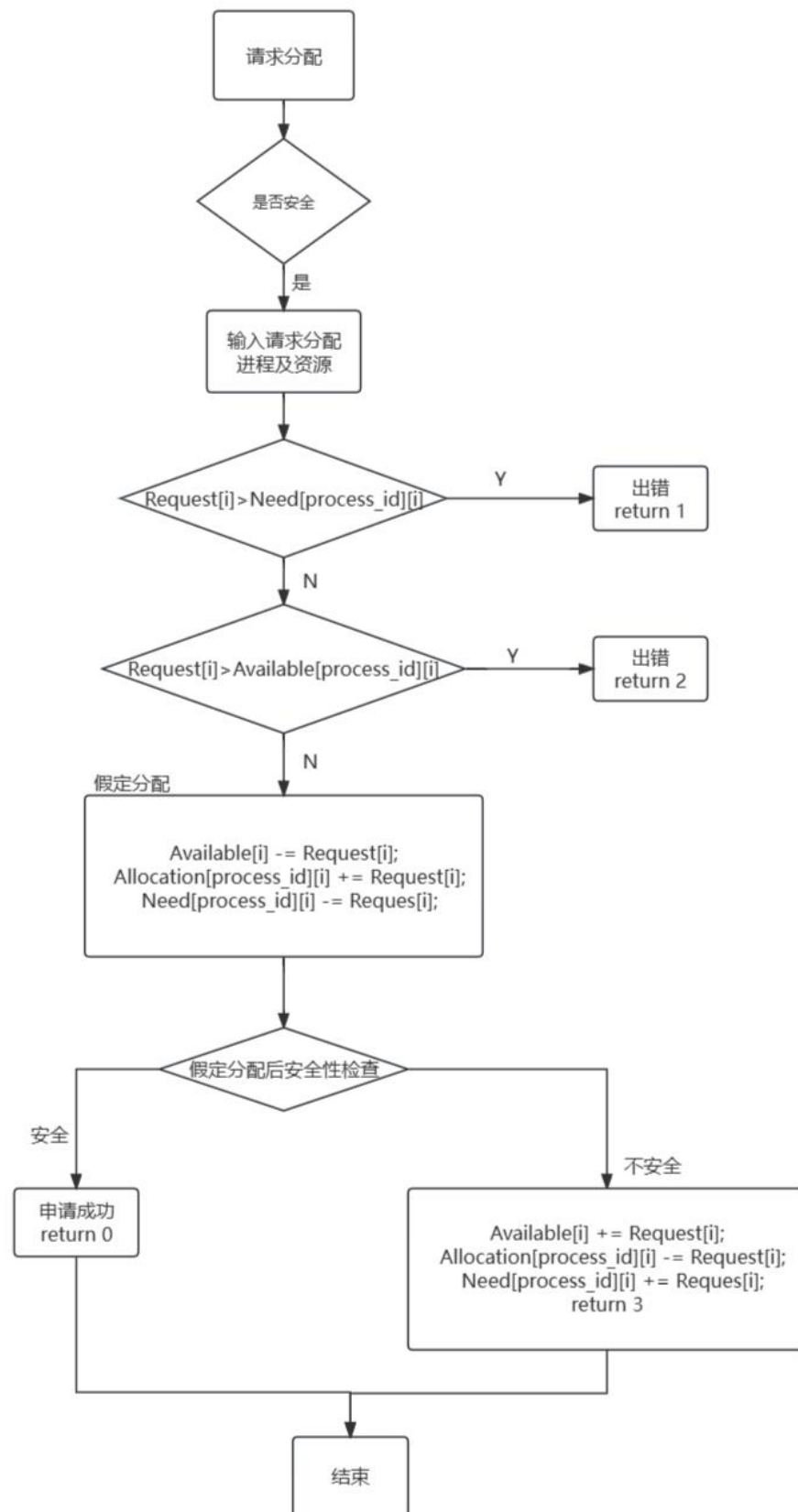
1. 主程序：



2.安全性检查:



3.请求资源



三、编码实现

检测是否安全代码：

```
bool Banker::isSafe()
{
    vector<int> work;
    work = Available;
    vector<bool> isChecked(processNum, false);
    isSafeRecursive(work, isChecked);
    if (result.empty())
        return false;
    return true;
}

void Banker::isSafeRecursive(vector<int>& work, vector<bool>& isChecked)
{
    bool hasFound = false;
    if (path.size() == processNum)
    {
        result.push_back(path);
        safePrint(print_num);
        add_printNum();
        return;
    }
    for (int i = 0; i < processNum; i++)
    {
        if (!isChecked[i])
        {
            for (int j = 0; j < resourceNum; j++)
            {
                if (Need[i][j] > work[j])
                    break;
                if (j == resourceNum - 1)
                {
                    for (int k = 0; k < resourceNum; k++)
                        work[k] += Allocation[i][k];
                    isChecked[i] = true;
                    hasFound = true;
                    path.push_back(i);
                    resourcePrint(work, i, isChecked);
                    isSafeRecursive(work, isChecked);
                    path.pop_back();
                    isChecked[i] = false;
                    for (int k = 0; k < resourceNum; k++)
                        work[k] -= Allocation[i][k];
                }
            }
        }
    }
}
```

```

        }
    }
}
if (!hasFound)
    return;
}

分配资源代码:
int Banker::requestResource(int process_id, vector<int> req)
{
    cout << "开始分配: " << endl << endl;
    for (int i = 0; i < resourceNum; i++)
        if (req[i] > Need[process_id][i])
            return 1;
    for (int i = 0; i < resourceNum; i++)
        if (req[i] > Available[i])
            return 2;
    for (int i = 0; i < resourceNum; i++)    //分配资源
    {
        Available[i] -= req[i];
        Allocation[process_id][i] += req[i];
        Need[process_id][i] -= req[i];
    }
    resourcePrint();
    if (!isSafe())
    {
        for (int i = 0; i < resourceNum; i++)    //回收资源
        {
            Available[i] += req[i];
            Allocation[process_id][i] -= req[i];
            Need[process_id][i] += req[i];
        }
        return 3;
    }
    else
        for (int i = 0; i < resourceNum; i++)    //恢复
        {
            Available[i] += req[i];
            Allocation[process_id][i] -= req[i];
            Need[process_id][i] += req[i];
        }
    return 0;
}

```

```

int Banker::requestResource(int process_id, vector<int> req)
{
    cout << "开始分配: " << endl << endl;
    for (int i = 0; i < resourceNum; i++)
        if (req[i] > Need[process_id][i])
            return 1;
    for (int i = 0; i < resourceNum; i++)
        if (req[i] > Available[i])
            return 2;
    for (int i = 0; i < resourceNum; i++)    //分配资源
    {
        Available[i] -= req[i];
        Allocation[process_id][i] += req[i];
        Need[process_id][i] -= req[i];
    }
    resourcePrint();
    if (!isSafe())
    {
        for (int i = 0; i < resourceNum; i++)    //回收资源
        {
            Available[i] += req[i];
            Allocation[process_id][i] -= req[i];
            Need[process_id][i] += req[i];
        }
        return 3;
    }
    //下属代码看需求，如果是连续分配，则不需要恢复，如果只是尝试单次，即可恢复
    /*else
        for (int i = 0; i < resourceNum; i++)    //恢复
        {
            Available[i] += req[i];
            Allocation[process_id][i] -= req[i];
            Need[process_id][i] += req[i];
        }
    return 0;*/
}

```

四、实验结果

1.程序开始:

```
C:\Users\12404\Documents\G  ×  +  v
进程分配表
进程\资源      Max      Allocation      Need
P0              7 5 3      0 1 0      7 4 3
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

Available
0 0 0

      请选择你要进行的操作:
      1.对当前资源表进行安全性检查。
      2.请求资源。
      0.退出
请输入: |
```

2.安全性检查

①分配过程

```
C:\Users\12404\Documents\G  ×  +  v
分配给进程 1
未分配资源进程
进程\资源      Max      Allocation      Need
P0              7 5 3      0 1 0      7 4 3
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
分配后Available:
5 3 2
-----
分配给进程 3
未分配资源进程
进程\资源      Max      Allocation      Need
P0              7 5 3      0 1 0      7 4 3
P2              9 0 2      3 0 2      6 0 0
P4              4 3 3      0 0 2      4 3 1
分配后Available:
7 4 3
-----
分配给进程 0
未分配资源进程
进程\资源      Max      Allocation      Need
P2              9 0 2      3 0 2      6 0 0
P4              4 3 3      0 0 2      4 3 1
分配后Available:
7 5 3
-----
分配给进程 2
未分配资源进程
```

②所有安全序列

```
安全序列为:
1 -> 3 -> 0 -> 2 -> 4
1 -> 3 -> 0 -> 4 -> 2
1 -> 3 -> 2 -> 0 -> 4
1 -> 3 -> 2 -> 4 -> 0
1 -> 3 -> 4 -> 0 -> 2
1 -> 3 -> 4 -> 2 -> 0
1 -> 4 -> 3 -> 0 -> 2
1 -> 4 -> 3 -> 2 -> 0
3 -> 1 -> 0 -> 2 -> 4
3 -> 1 -> 0 -> 4 -> 2
3 -> 1 -> 2 -> 0 -> 4
3 -> 1 -> 2 -> 4 -> 0
3 -> 1 -> 4 -> 0 -> 2
3 -> 1 -> 4 -> 2 -> 0
3 -> 4 -> 1 -> 0 -> 2
3 -> 4 -> 1 -> 2 -> 0

请按任意键继续. . . |
```


③不安全状态

```
C:\Users\12404\Documents\G x + v
进程分配表
进程\资源    Max      Allocation    Need
P0           7 5 3      0 1 0        7 4 3
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

Available
0 0 0

请选择你要进行的操作：
1.对当前资源表进行安全性检查。
2.请求资源。
0.退出
请输入：1

不安全
请按任意键继续. . .
```

3. 请求资源：

①不安全状态

```
C:\Users\12404\Documents\G x + v
当前数据状态不安全，无法请求资源！
进程分配表
进程\资源    Max      Allocation    Need
P0           7 5 3      0 1 0        7 4 3
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

Available
1 3 2

请选择你要进行的操作：
1.对当前资源表进行安全性检查。
2.请求资源。
0.退出
请输入：|
```

②安全

I. 请求资源超过最大请求值

```
C:\Users\12404\Documents\G x + v
进程分配表
进程\资源    Max      Allocation    Need
P0           7 5 3      0 1 0        7 4 3
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

Available
3 3 2

请选择你要进行的操作：
1.对当前资源表进行安全性检查。
2.请求资源。
0.退出
请输入：2

安全，可以请求，请输入请求资源号：1
请输入请求资源数： x...x 的形式(1x3矩阵)：1 3 2

开始分配：

分配失败！请求的资源数超过最大值！
请按任意键继续. . .
```

II. Available 小于请求资源

```
C:\Users\12404\Documents\G >
进程分配表
进程\资源    Max      Allocation    Need
P0           7 5 3      0 1 0      7 4 3
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

Available
3 3 2

请选择你要进行的操作：
1.对当前资源表进行安全性检查。
2.请求资源。
0.退出
请输入：2

安全，可以请求,请输入请求资源号：0
请输入请求资源数： x...x 的形式(1x3矩阵)： 3 3 3

开始分配：

分配失败！系统中尚无足够的资源满足P0的请求
请按任意键继续. . .
```

III. 假定分配后，无法通过安全性检查

```
C:\Users\12404\Documents\G >
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

Available
3 3 2

请选择你要进行的操作：
1.对当前资源表进行安全性检查。
2.请求资源。
0.退出
请输入：2

安全，可以请求,请输入请求资源号：4
请输入请求资源数： x...x 的形式(1x3矩阵)： 3 3 1

开始分配：

进程分配表
进程\资源    Max      Allocation    Need
P0           7 5 3      0 1 0      7 4 3
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      3 3 3      1 0 0

Available
0 0 1

分配失败！假定分配后，无法通过安全性检查！
请按任意键继续. . .
```

IV. 分配成功

```
C:\Users\12404\Documents\G >
安全，可以请求,请输入请求资源号：4
请输入请求资源数： x...x 的形式(1x3矩阵)： 1 0 0

开始分配：

进程分配表
进程\资源    Max      Allocation    Need
P0           7 5 3      0 1 0      7 4 3
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      1 0 2      3 3 1

Available
2 3 2

安全序列为：
1 -> 3 -> 2 -> 0 -> 4
1 -> 3 -> 2 -> 4 -> 0
1 -> 3 -> 4 -> 0 -> 2
1 -> 3 -> 4 -> 2 -> 0
1 -> 4 -> 3 -> 0 -> 2
1 -> 4 -> 3 -> 2 -> 0
3 -> 1 -> 2 -> 0 -> 4
3 -> 1 -> 2 -> 4 -> 0
3 -> 1 -> 4 -> 0 -> 2
3 -> 1 -> 4 -> 2 -> 0
3 -> 4 -> 1 -> 0 -> 2
3 -> 4 -> 1 -> 2 -> 0

分配成功！
请按任意键继续. . .
```

五、思考与体会

这次课程设计实现了操作系统第五次实验银行家算法，让我更加深刻的理解了进程死锁这一概念，也学习了如何进行死锁预防和死锁避免，而操作系统课程设计银行家算法正是通过死锁避免来实现的，通过自己编写代码，大大加深了对于操作系统第五次实验银行家算法细节的理解，理解了如何通过安全性算法来检查系统状态是否安全，也更加理解了如何通过资源请求算法来请求资源分配和释放未成功分配的资源。