实验二　银行家算法　　学时:2

(一)实验类型：验证型

(二)实验类别：专业实验

(三)实验要求：必修

(四)实验目的

银行家算法是避免死锁的一种重要方法，通过编写一个简单的银行家算法程序，加深了解有关资源申请、避免死锁等概念，并体会和了解死锁和避免死锁的具体实施方法。

(五)实验内容

先对用户提出的请求进行合法性检查，即检查请求是否大于需要的，是否大于可利用的。若请求合法，则进行预分配，对分配后的状态调用安全性算法进行检查。若安全，则分配；若不安全，则拒绝申请，恢复到原来的状态，拒绝申请。

实验使用 C 语言模拟实现银行家算法，写出程序，并正确运行程序。

(六)实验方法、步骤及结果测试。

1、银行家算法步骤:

（1）如果 Requesti＜or =Need,则转向步骤(2)；否则，认为出错，因为它所需要的资源数已超过它所宣布的最大值。

（2）如果 Request＜or=Available,则转向步骤（3）；否则，表示系统中尚无足够的资源，进程必须等待。

（3）系统试探把要求的资源分配给进程 Pi,并修改下面数据结构中的数值：

```
Available=Available-Request[i];
Allocation=Allocation+Request;
Need=Need-Request;
```

（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。

2、安全性算法步骤:

（1）设置两个向量

①工作向量 Work。它表示系统可提供进程继续运行所需要的各类资源数目，执行安全算法开始时，Work=Allocation;

②布尔向量 Finish。它表示系统是否有足够的资源分配给进程，

使之运行完成，开始时先做 Finish[i]=false，当有足够资源分配给进程时，令 Finish[i]=true。

（2）从进程集合中找到一个能满足下述条件的进程：

①Finish[i]=false

②Need<or=Work

如找到，执行步骤（3）；否则，执行步骤（4）。

（3）当进程 P 获得资源后，可顺利执行，直至完成，并释放出分配给它的资源，故应执行：

Work=Work+Allocation;

Finish[i]=true;

转向步骤（2）。

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

3、流程图:

系统主要过程流程图，如图 3 所示。

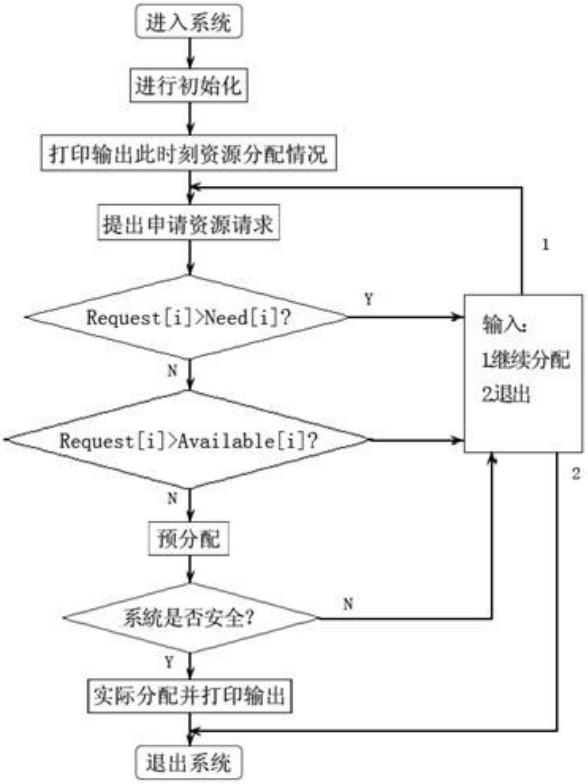

图 3　系统主要过程流程图

银行家算法流程图，如图 4 所示：

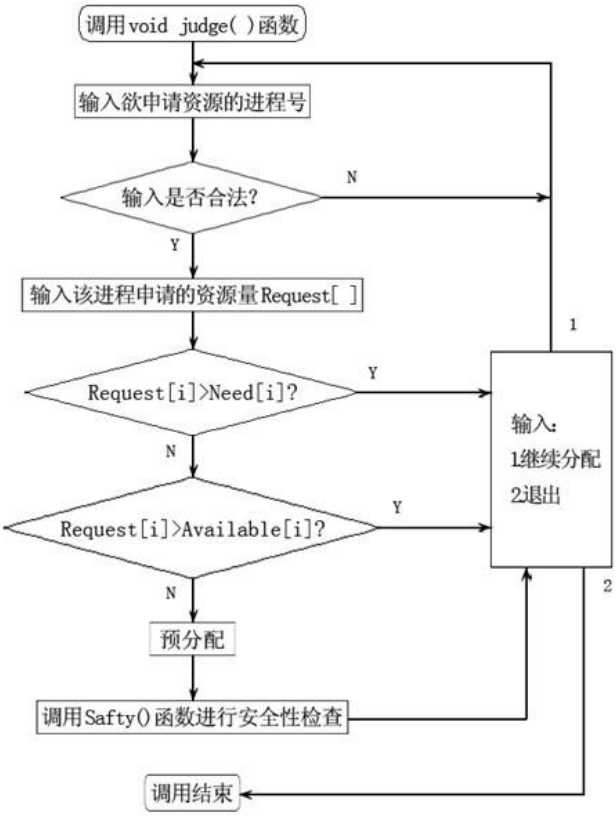

图 4　银行家算法流程图

安全性算法流程图，如图 5 所示：

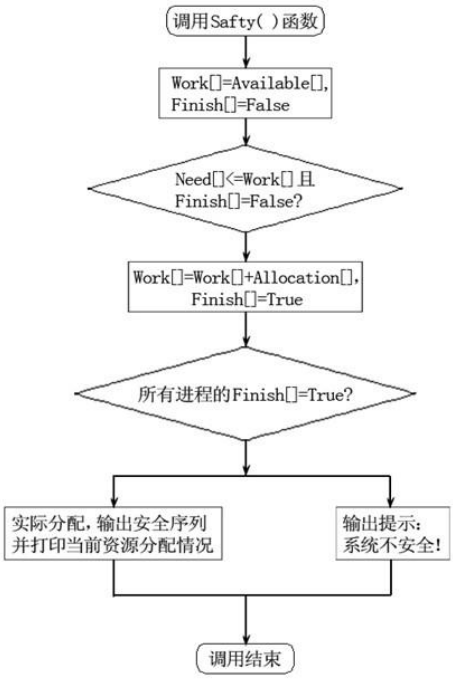

图 5　安全性算法流程图