**天津财经大学理工学院信息科学与技术系**

**《操作系统》实验报告**

实验 2 银行家算法

**班 级：软件 2201**

**学 号：2022113179**

**姓 名：钟奇林**

# 一、实验目的

通过本实验模拟实现操作系统管理有限资源，避免产生死锁的方法，从而加深对银行家算法的理解。

# 二、实验内容

编写银行家算法(单资源银行家算法、多资源银行家算法)的程序。判断系统是否处于安全状态，要求运行程序时，输入数据包括:进程数量，已分配资源向量(或矩阵),请求资源向量(或矩阵)和可用资源向量；程序运行结果是：如果系统当前状态是安全的，则输出资源分配的安全序列，如果系统当前状态是不安全的，则显示此处请求资源不能满足的提示即可。

# 三、实验结果

核心代码说明如下：

*// 获取安全序列的函数*

**bool** getSafeSeq() {

**int**\* tempRes = (**int**\*)malloc(nResources \* sizeof(**int**)); *// 为临时资源分配内存*

    for (**int** i = 0; i < nResources; i++)tempRes[i] = resources[i]; *// 将资源复制到临时资源数组*

**bool**\* finished = (**bool**\*)malloc(nResources \* sizeof(**bool**)); *// 为进程完成状态分配内存*

    for (**int** i = 0; i < nProcesses; i++)finished[i] = false; *// 初始化所有进程为未完成状态*

**int** nfinished = 0; *// 完成的进程数量初始化为0*

    while (nfinished < nProcesses) { *// 当还有未完成的进程时循环*

**bool** safe = false; *// 初始化安全状态为假*

        for (**int** i = 0; i < nProcesses; ++i) { *// 遍历所有进程*

            if (!finished[i]) { *// 如果当前进程未完成*

**bool** possible = true; *// 初始化能否运行状态为真*

                for (**int** j = 0; j < nResources; ++j) *// 遍历所有资源*

                    if (need[i][j] > tempRes[j]) { *// 如果需要的资源大于临时资源*

                        possible = false; *// 设置为不能运行*

                        break;

                    }

                if (possible) { *// 如果当前剩余资源满足当前进程需求*

                    for (**int** j = 0; j < nResources; ++j) *// 更新临时资源*

                        tempRes[j] += allocated[i][j];

                    safeSeq[nfinished] = i; *// 添加当前进程到安全序列*

                    finished[i] = true; *// 设置当前进程为完成状态*

                    ++nfinished; *// 完成的进程数量加1*

                    safe = true; *// 找到至少一个安全序列的节点，暂时处于安全*

                }

            }

        }

        if (!safe) { *// 如果未找到安全序列*

            for (**int** i = 0; i < nProcesses; i++)safeSeq[i] = -1; *// 将安全序列全部置为-1*

            return false; *// 返回失败*

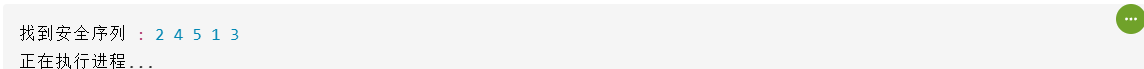
        }

    }

    return true; *// 返回成功*

}

运行结果如下：



# 四、实验总结

这次试验模拟银行家算法，模拟了操作系统是如何有效地预防了多个进程因资源竞争而导致的死锁问题。实际上整个程序比较简单的，这个程序的缺陷就是没有真正的模拟多个线程的执行，难点在于后面拓展模拟多个线程的执行，一个<pthread.h>头文件属于linux，一个<thread>是c++的，都没法使用纯c实现,最终模拟早期Unix系统的多线程实现，使用while循环。