

渤海大学学生实验报告

(信息科学与技术学院)

实验课课程名称: 操作系统

实验室 房间号	I科楼C504	日期 时间	2022 年 10 月 21 日 第 (3.4) 节		
年级、班	20级国经	学号	20012349	姓名	孙达明
实验项目 名称	银行家算法			指导教师	孙德才
实验环境	PC兼容机, Windows 10, Dev-C++			成绩	
实验目的	用高级语言编写一个银行家的模拟算法 通过本实验对预防死锁和银行家算法更深刻的认识				

【实验内容】(算法、程序、步骤、数据记录与计算、实验结果和讨论等)

实验内容与步骤

编写程序模拟银行家算法的计算过程:

1. 设置数据结构

包括可利用资源向量(Available)、最大需求矩阵(Max)、分配矩阵(Allocation)、需求矩阵(Need)

2. 设计安全性算法

设置工作向量Work表示系统可提供进程继续运行可利用资源项目, Finish表示系统是否有足够的资源分配给进程, 编写算法模拟计算。

3. 以书上121页的例子作为数据, 模拟计算过程

4. Banketm.c 代码已撰写了本实验的主函数以验一些辅助函数关键代码 banker函数未全部提供, 请补充完整。

主要程序:

```
#include <stdio.h>
```

```
#define N 5
```

```
#define M 3
```

```
Void showSystemStatus();
```

```
Void Banker(int, int*);
```

```
int SafeCheck();
```

```
Void showMenu();
```

```
int Available[M] = {3, 3, 2};
```

```
int Max[N][M] = {{7, 5, 3}, {3, 2, 2}, {9, 0, 2}, {2, 2, 2}, {4, 3, 3}};
```

```
int Allocation[N][M] = {{0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}};
```

```
int Need[N][M];
```



```

void Banker (int P, int *R) {
    int i, j;
    printf ("Request P %d", P);
    for (i=0; i<M; i++) { printf ("%d", R[i]); }
    for (i=0; i<M; i++) { if (R[i] > Need[P][i]) {
        printf ("(b) > Need P %d", P);
        for (i=0; i<M; i++) {
            printf ("%d", Need[P][i]);
        }
        printf ("(b); \n");
        printf ("申请失败"); return i; }
        printf ("(b) <= Need P %d", P);
        for (i=0; i<M; i++) { printf ("%d", Need[P][i]); }
        printf ("(b); \n");
    }
    printf ("Request P %d", P);
    for (i=0; i<M; i++) { printf ("%d", R[i]); }
    for (j=0; j<M; j++) { if (R[j] > Available[j]) {
        printf ("(b) > Available");
        for (i=0; i<M; i++) {
            printf ("%d", Available[i]);
        }
        printf ("(b); \n");
        printf ("申请失败"); return i; }
        printf ("(b) <= Available");
        for (i=0; i<M; i++) { printf ("%d", Available[i]); }
        printf ("(b); \n");
        printf ("分配: \n");
        for (j=0; j<M; j++) { Available[j] = Available[j] - R[j];
            Allocation[P][j] = Allocation[P][j] + R[j];
            Need[P][j] = Need[P][j] - R[j]; }
        showSystemStatus();
        if (safeCheck() == 0) {
            for (j=0; j<M; j++) {
                Available[j] = Available[j] + R[j];
                Allocation[P][j] = Allocation[P][j] - R[j];
                Need[P][j] = Need[P][j] + R[j]; } else
                printf ("申请成功");

```

教师签字:

年 月 日