

电子科技大学

计算机专业类课程

实验报告

课程名称：操作系统

学 院：计算机科学与工程

专 业：计算机科学与技术

学生姓名：韩会彬

学 号：2013060105004

指导教师：薛瑞尼

日 期： 2016 年 06 月 05 日

电子科技大学

实验报告

实验二

一、实验名称：银行家算法

二、实验学时：4

三、实验内容和目的：

编写程序实现银行家算法，要求：

- 输入
 - p: 进程数量
 - r: 资源数量
 - 各进程的 max, allocation
- 输出
 - 若产生死锁，打印提示：死锁状态。
 - 否则，给出一种调度顺序

四、实验原理：

多个进程在运行过程中因争夺资源而造成的一种僵局（Deadly-Embrace），当进程处于这种僵持状态时，若无外力作用，它们都将无法再向前推进。

产生死锁的原因

- 资源不足导致的资源竞争：多个进程所共享的资源不足，引起它们对资源的竞争而产生死锁。
- 并发执行的顺序不当：进程运行过程中，请求和释放资源的顺序不当，而导致进程死锁。

当用户申请资源时，系统判断如果把这些资源分出去，系统是否还处于安全状态。

- 若是，就可以分配这些资源；
- 否则，暂时不分配，阻塞进程。

安全序列：一个进程序列 $\{P_1, \dots, P_n\}$ 是安全的，如果对于每一个进程 $P_i (1 \leq i \leq n)$ ，它尚需要的资源量不超过系统当前剩余资源量与所有进程 $P_j (j < i)$ 当前占有资源量之和。

设系统中有 m 种不同资源， n 个进程

- Resource[j], j 资源的数量
- Available 向量: 系统中尚未分配的每种资源的总量
 - Available[j]: 尚未分配的资源 j 的数量
- Max 矩阵: 各个进程对每种资源的最大需求量(进程事先声明)
 - Max[i, j](Claim[i, j]): 进程 i 对资源 j 的最大需求量
- Allocation 矩阵: 当前资源分配状况
 - Allocation[i, j]: 进程 i 获得的资源 j 的数量
- Need 矩阵: 每个进程还需要的剩余资源的数量
 - Need[i, j]: 进程 i 尚需的资源 j 的数量

五、实验过程和结果:

实验代码:

```
#include<stdio.h>

#include<malloc.h>
#include<math.h>

int main() {
    int p = 0, r = 0;
    int** max = NULL;    //最大需求
    int** all = NULL;    //已分配
    int* ava = NULL;     //剩余资源
    int* sign = NULL;    //标志是否已分配
    int** need = NULL;
    int* que = NULL;

    printf("enter number of process:");
    scanf_s("%d", &p);
    printf("enter number of resources:");
    scanf_s("%d", &r);

    max = (int **)malloc(sizeof(int *)*p);
    all = (int **)malloc(sizeof(int *)*p);
    need = (int **)malloc(sizeof(int *)*p);
    ava = (int *)malloc(sizeof(int)*r);
    sign = (int *)malloc(sizeof(int)*(p+1));
    que = (int *)malloc(sizeof(int)*(p + 1));

    for (int i = 0; i < p; i++) {
        *(max + i) = (int *)malloc(sizeof(int)*r);
        *(all + i) = (int *)malloc(sizeof(int)*r);
        *(need + i) = (int *)malloc(sizeof(int)*r);
    }

    for (int jp = 0; jp < p; jp++) {
```

```

        for (int jr = 0; jr < r; jr++){
            printf("enter max[%d][%d]:", jp, jr);
            scanf_s("%d", &max[jp][jr]);
        }
    }
    for (int jp = 0; jp < p; jp++){
        for (int jr = 0; jr < r; jr++){
            printf("enter all[%d][%d]:", jp, jr);
            scanf_s("%d", &all[jp][jr]);
        }
    }
    for (int jp = 0; jp < p; jp++){
        for (int jr = 0; jr < r; jr++){
            need[jp][jr] = max[jp][jr] - all[jp][jr];
        }
    }
    for (int jp = 0; jp < p; jp++){
        sign[jp] = 0;
    }
    for (int jr = 0; jr < r; jr++){
        printf("enter ava[%d]", jr);
        scanf_s("%d", &ava[jr]);
    }
    int mnum = p;
    int si = 0;
    int pn = 0;
    int ds = 0;
    int pss = 1;
    while (mnum != 0) {

        for (int kp = 0; kp < p; kp++){
            si = 0;
            ds = ds + 1;
            if (sign[kp] == 0) {
                for (int kr = 0; kr < r; kr++){
                    if (need[kp][kr] > ava[kr]) {
                        si = 1;
                    }
                }
            }
            if (si == 0) {
                ds = 0;
                mnum = mnum - 1;
                sign[kp] = 1;
                que[pn] = kp+1;
            }
        }
    }

```

```

        pn = pn + 1;
        for (int kk = 0; kk < r; kk++) {
            ava[kk] = ava[kk] + all[kp][kk];
        }
    }
}

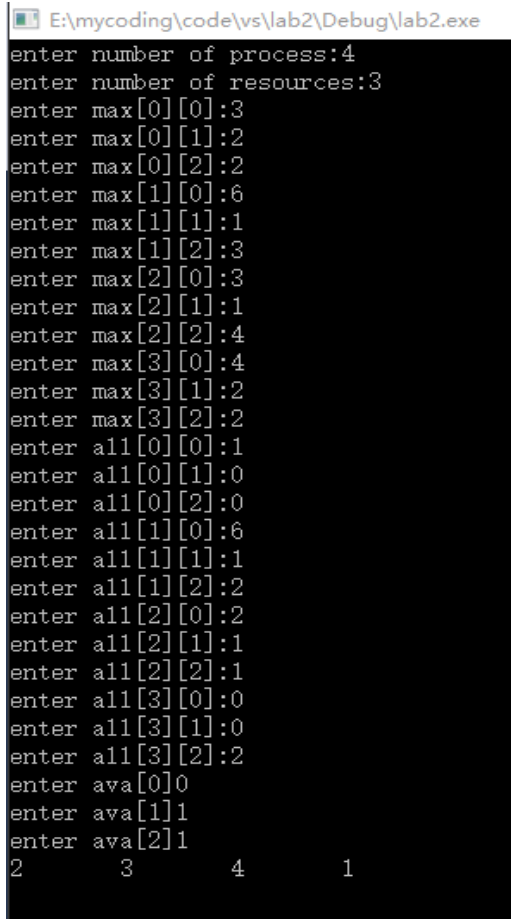
if (ds == p) {
    pss = 0;
    printf("死锁状态");
    break;
}
}

if (pss == 1) {
    for (int pp = 0; pp < p; pp++) {
        printf("%d\t", que[pp]);
    }
}

printf("over");
return 0;
}

```

没有死锁时运行结果:

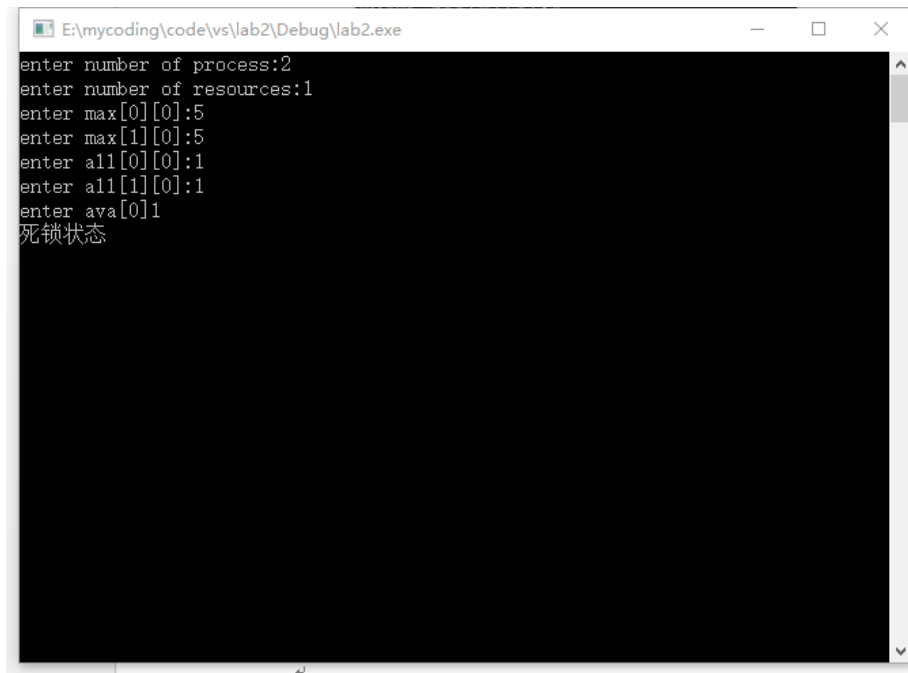


```

E:\mycoding\code\vs\lab2\Debug\lab2.exe
enter number of process:4
enter number of resources:3
enter max[0][0]:3
enter max[0][1]:2
enter max[0][2]:2
enter max[1][0]:6
enter max[1][1]:1
enter max[1][2]:3
enter max[2][0]:3
enter max[2][1]:1
enter max[2][2]:4
enter max[3][0]:4
enter max[3][1]:2
enter max[3][2]:2
enter all[0][0]:1
enter all[0][1]:0
enter all[0][2]:0
enter all[1][0]:6
enter all[1][1]:1
enter all[1][2]:2
enter all[2][0]:2
enter all[2][1]:1
enter all[2][2]:1
enter all[3][0]:0
enter all[3][1]:0
enter all[3][2]:2
enter ava[0]0
enter ava[1]1
enter ava[2]1
2      3      4      1

```

有死锁时运行结果：



```
E:\mycoding\code\vs\lab2\Debug\lab2.exe
enter number of process:2
enter number of resources:1
enter max[0][0]:5
enter max[1][0]:5
enter all[0][0]:1
enter all[1][0]:1
enter ava[0]1
死锁状态
```

六、心得和体会：

通过本次实验进一步掌握了死锁的原因和预防死锁的原理和方法,进一步熟悉了银行家算法的原理。