电子科技大学

实
验
报
告

课程：计算机操作系统

姓名：李果念

学号：2013040203030

银行家算法程序

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。

实验源代码：

```
//这个实验参考刘乃琦、蒲晓蓉主编，高等教育出版社出版的《操作系统原理、设计及应用》这本灰皮书的第116页到第120页
#include <stdio.h>
#include <string.h>

#define KINDS 31
#define NUM 31

unsigned char p;
unsigned char r;

unsigned char available[KINDS];
unsigned char lock;

struct PROCESS
{
unsigned char max[KINDS];
unsigned char allocation[KINDS];
unsigned char need[KINDS];
unsigned char scheduled;
}process[NUM];

struct SCHEDULE
{
unsigned char processNO;
unsigned char work[KINDS];
unsigned char need[KINDS];
unsigned char allocation[KINDS];
unsigned char workplusallo[KINDS];
unsigned char finish;
}schedule[NUM];

void inputresources(void)
{
    unsigned char i=0;
    unsigned char j=0;
    printf("银行家算法程序\n 注意：所有的下标都是从0开始的。\n 如果要求输入多个数，那么数与数之间必须用空格隔开。\n");
    printf("请输入进程数量\n",i);
    scanf("%d",&p);
    printf("请输入资源数量\n",i);
    scanf("%d",&r);
    for (i=0;i<p;i++)
    {
        printf("请输入进程P%d需要的最大资源数max\n",i);
        for (j=0;j<r;j++)
            scanf("%d",&process[i].max[j]);
    }
    for (i=0;i<p;i++)
    {
        printf("请输入进程P%d已分配的资源数allocation\n",i);
        for (j=0;j<r;j++)
            scanf("%d",&process[i].allocation[j]);
    }
    printf("请输入剩余的资源数available\n",i);
    for (j=0;j<r;j++)
        scanf("%d",&available[j]);
```

```
    for (i=0;i<p;i++)
    {
        process[i].scheduled=0;
        for (j=0;j<r;j++)
        process[i].need[j]=process[i].max[j]-process[i].allocation[j];
    }
}

unsigned char isavailable(unsigned char p[], unsigned char q[])
{
    unsigned char res=0;
    unsigned char s=0;
    unsigned char t=0;
    for (t=0;t<r;t++)
    {
        if (p[t]<=q[t])
            res++;
    }
    s=res/r;
    return s;
}

int main()
{
    unsigned char a;
    unsigned char b=0;
    unsigned char i;
    unsigned char j;
    unsigned char k;
    lock=0;
    inputresources();
    for (i=0;i<p;i++)
    {
        a=isavailable(process[i].need,available);
        b=b+a;
        if (a)
            break;
    }
    if (b==0)
    {
        printf("会死锁！\n");
        lock=1;
        return 0;
    }
    else
    {
        process[i].scheduled=1;
        schedule[0].processNO=i;
        memcpy(schedule[0].work,available,r);
        memcpy(schedule[0].need,process[i].need,r);
        memcpy(schedule[0].allocation,process[i].allocation,r);
        for (j=0;j<r;j++)
            schedule[0].workplusallo[j]=schedule[0].work[j]+schedule[0].allocation[j];
        schedule[0].finish=1;
        memcpy(schedule[1].work,schedule[0].workplusallo,r);
    }
    for (k=1;k<p;k++)
    {
        a=0;
```

```
        b=0;
        for (i=0;i<p;i++)
        {
            if (!process[i].scheduled)
            {
            a=isavailable(process[i].need,schedule[k].work);
            b=b+a;
            if (a)
                break;
            }
            else ;
        }
        if (b==0)
        {
            printf("会死锁！\n");
            lock=1;
            return 0;
        }
        else
        {
            process[i].scheduled=1;
            schedule[k].processNO=i;
            memcpy(schedule[k].need,process[i].need,r);
            memcpy(schedule[k].allocation,process[i].allocation,r);
            for (j=0;j<r;j++)

    schedule[k].workplusallo[j]=schedule[k].work[j]+schedule[k].allocation[j];
            schedule[k].finish=1;
            if (k<NUM-1)
            memcpy(schedule[k+1].work,schedule[k].workplusallo,r);
        }
    }
    if (!lock)
    {
        printf("调度顺序为：\n");
        for (i=0;i<p;i++)
        {
            printf("%d",schedule[i].processNO);
            if (i<p-1)
                printf("→");
            else printf("\n");
        }
    }
    return 0;
}
```

运行结果：

银行家算法程序
注意：所有的下标都是从 0 开始的。
如果要求输入多个数，那么数与数之间必须用空格隔开。
请输入进程数量
4
请输入资源数量
3
请输入进程 P0 需要的最大资源数 max
3 2 2
请输入进程 P1 需要的最大资源数 max
6 1 3
请输入进程 P2 需要的最大资源数 max
3 1 4
请输入进程 P3 需要的最大资源数 max
4 2 2
请输入进程 P0 已分配的资源数 allocation
1 0 0
请输入进程 P1 已分配的资源数 allocation
6 1 2
请输入进程 P2 已分配的资源数 allocation
2 1 1
请输入进程 P3 已分配的资源数 allocation
0 0 2
请输入剩余的资源数 available
0 1 1
调度顺序为：
1→0→2→3

结论：

如果一个进程执行完了，那么它会释放所有它原先占有的资源。