w

电子科技大学

计算机专业类课程

实验报告

课程名称：计算机操作系统

学　　院：计算机科学与工程学院

专　　业：信息安全

学生姓名：郑啸宇

学　　号：2013060202013

指导教师：薛瑞尼

日　　期：　2016 年　6 月　8 日

电 子 科 技 大 学

实　验　报　告

实验二

一、实验名称：银行家算法程序

二、实验学时：2

三、实验内容和目的：

（1）实验内容：

- 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。

（2）实验目的：

熟练掌握银行家算法的原理，能利用银行家算法的思想来避免死锁，并能顺利用代码实现所要求的内容。

四、实验原理：

1.银行家算法的思路

先对用户提出的请求进行合法性检查，即检查请求的是不大于需要的，是否不大于可利用的。若请求合法，则进行试分配。最后对试分配后的状态调用安全性检查算法进行安全性检查。若安全，则分配，否则，不分配，恢复原来状态，拒绝申请。

2.银行家算法中用到的主要数据结构

可利用资源向量　Available[j]　j 为资源的种类。

最大需求矩阵　　Max[i][j]　 i 为进程的数量。

分配矩阵　　Allocation[i][j]

需求矩阵　　need[i][j]= Max[i][j]- Allocation[i][j]

申请各类资源数量　Request i[j]　i 进程申请 j 资源的数量

工作向量　　Work[i]　 Finish[j]

3．银行家算法

进程 i 发出请求申请 k 个 j 资源，Request i[j]=k

(1)检查申请量是否不大于需求量：Request i[j]<=need[i,j],若条件不符重新输入，

不允许申请大于需求量。

(2)检查申请量是否小于系统中的可利用资源数量：Request i[j]<=available[i,j]，若条件不符就申请失败，阻塞该进程，用 goto 语句跳转到重新申请资源。

(3)若以上两个条件都满足，则系统试探着将资源分配给申请的进程，并修改下面数据结构中的数值：

Available[i,j]= Available[i,j]- Request i[j]；

Allocation[i][j]= Allocation[i][j]+ Request i[j]；

need[i][j]= need[i][j]- Request i[j]；

(4)试分配后，执行安全性检查，调用 safe()函数检查此次资源分配后系统是否处于安全状态。若安全，才正式将资源分配给进程；否则本次试探分配作废，恢复原来的资源分配状态，让该进程等待。

4．安全性检查算法

(1)设置两个向量：工作向量 Work，它表示系统可提供给进程继续运行所需的各类资源数目，在执行安全性算法开始时，Work= Available。 Finish=false 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]=0；

当有足够的资源分配给进程时，再令 Finish[i]=true。

(2)在进程中查找符合以下条件的进程： 条件 1：Finish[i]=false； 条件 2：

need[i][j]<=Work[j]

若找到，则执行步骤(3)否则，执行步骤(4)

(3)当进程获得资源后，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[j]= Work[j]+ Allocation[i][j]；

Finish[i]=true；

goto step 2；

(4)如果所有的 Finish[i]=true 都满足，则表示系统处于安全状态，否则，处于不

安全状态。

五、实验器材（设备、元器件）

操作系统：Windows10

六、实验数据及结果分析：

1、为了好进行检验，直接用书上的数据，先设置 5 个进程，然后设置三类资源，各种资源的最大数量分别为 10、5、7

各类进程对资源的最大需求 max 以及已经分配的进程 allocation 的数量如下：

```
# of processes:5
# of resources:3
total of each resource:10 5 7
#1 process's max demands to every resouce:7 5 3
#1 process's allocation already to every resouce:0 1 0
#2 process's max demands to every resouce:3 2 2
#2 process's allocation already to every resouce:2 0 0
#3 process's max demands to every resouce:9 0 2
#3 process's allocation already to every resouce:3 0 2
#4 process's max demands to every resouce:2 2 2
#4 process's allocation already to every resouce:2 1 1
#5 process's max demands to every resouce:4 3 3
#5 process's allocation already to every resouce:0 0 2
(1, 3, 0, 2, 4)
>>>
```

运行结果为 (1，3，0，2，4)与书上的结果一致，不会出现死锁。

分析:因为 1 获得资源，运行结束，将会释放各类资源为 7 5 3 ，这些资源便能满足 3 进程所需要的，3 获得了这些资源，运行结束后还会释放自身的所有资源，依次运行，各进程均能成功运行。

2、再运行一个错误的结果

5 个进程，3 类资源，各类资源的最大数量为 10、10、10

各类进程对资源的最大需求 max 以及已经分配的进程 allocation 的数量如下：

```
# of processes:5
# of resources:3
total of each resource:10 10 10
#1 process's max demands to every resouce:12 10 10
#1 process's allocation already to every resouce:0 0 0
#2 process's max demands to every resouce:9 9 7
#2 process's allocation already to every resouce:1 2 1
#3 process's max demands to every resouce:7 6 10
#3 process's allocation already to every resouce:2 1 1
#4 process's max demands to every resouce:9 9 9
#4 process's allocation already to every resouce:1 1 1
#5 process's max demands to every resouce:8 10 8
#5 process's allocation already to every resouce:0 0 1
no result
>>>
```

运行结果为 no result ，因为没有一个进程在得到资源后释放出来的总共资源可满足其他进程，无论哪个进程先获得资源，最后总有一个进程因为资源不足而阻塞，进而导致其他进程也无法获得资源来运行，造成死锁。

七、实验结论

顺利地完成了该实验，能按要求得到正确的输出，达到了该实验的要求。

八、心得体会

通过这次实验，让我对银行家算法的相关内容有了更进一步的理解，知道了什么是安全序列，也了解到了如何才能避免死锁，解决了之前理论学习中的一些困惑，提升了自己的动手能力。

九、改进建议：

暂无

报告评分：

指导教师签字：