深入解析银行家算法

侯　刚
（潍坊学院，山东　潍坊　261061）

摘　要：银行家算法是操作系统中采用避免死锁策略来解决死锁问题的一种算法。本文首先讲述了银行家算法流程，然后深入解析了银行家算法的根本原理。这在国内外操作系统教材中是第一次揭示银行家算法的根本原理。

关键词：银行家算法；死锁策略；数据结构
中图分类号：TP301.6　文献标识码：A　文章编号：1671-4288（2006）02-0046-03

1. 引言

死锁是操作系统学科中的一个重要研究内容。解决死锁问题可以采取三种策略：预防死锁、避免死锁、检测与解除死锁，银行家算法是避免死锁的算法。

银行家通过发放贷款而获取利润，要获取利润必须按时收回贷款本金和利息，即贷款企业要按时还本付息，而只有各企业能不断获得所需资金最终完成项目才能还本付息。要避免的情况是：在某个时刻，各并行推进项目的企业均得到了一部分贷款，要使项目顺利推进还需要贷款，而银行家已经没有多余资金，从而导致各企业间循环等待对方释放其占有的资金而使项目中断，造成一种僵滞局面，银行家因收不回贷款而破产。

操作系统的资源分配类似于银行家贷款。操作系统就像一个银行家，系统临界资源就像银行家的贷款本金，并发进程就像需要贷款的企业。因此可把银行家规避风险的算法引入操作系统的资源分配，解决资源分配中的死锁问题。这就是银行家算法这一名称的由来。

在操作系统的资源分配问题中要解决的是：一组并发进程 {P1,P2, … ,Pn} 共享一组临界资源 {R1,R2, … ,Rm}，Ri 指一类资源。当前，有一个进程 Pi 向操作系统提出了资源请求 $Request_i$=（r_1, r_2, … ,r_m）。若系统当前的可分配资源 Available=（a_1,a_2, … ,a_m）不能满足当前进程的资源请求，即不符合条件 $Request_i \leq$ Available，则拒绝当前进程的资源请求，阻塞该进程；若系统当前的可分配资源 Available 能满足当前进程的资源请求，即符合条件 $Request_i \leq$ Available，是否分配给该进程所需资源呢？

2. 银行家算法基本思想

在银行家算法中，为了决定是否对当前申请资源的进程进行资源分配，将系统状态划分为安全状态和不安全状态。若为当前申请资源的进程分配资源后系统进入安全状态，则接受进程请求为其分配资源，否则拒绝进程请求不为其分配资源。

安全状态：从当前时刻起，若系统能按某种进程顺序（P1,P2, … ,Pn）逐个分配所需全部剩余资源，使各进程依次运行完毕，则称此时刻系统处于安全状态，上述进程序列称为安全序列。否则系统处于不安全状态。注：定义的前提是：假设从当前时刻起，进程一次性申请全部剩余资源，而且一直保持当时已经占有的资源直至运行完毕。

3. 银行家算法中的数据结构

设有 n 个进程 P1、P2、……、Pn，共享 m 类资源 R1、R2、……、Rm。

⑴可利用资源向量 Available=(a_1,a_2, … ,a_m)

含有 m 个元素，每个元素表示某类资源可利用的单位数，初值为系统中所配置的该类资源全部单位数，其值随该类资源的分配和回收而动态改变。

⑵最大需求矩阵 Max

Max(i,j)=k，表示进程 Pi 总共需要 k 个单位的 Rj 类资源

收稿日期：2005-11-13
作者简介：侯刚（1970 —），男，山东高密人，潍坊学院计算机与通信工程学院讲师。

⑶分配矩阵 Allocation

Allocation（i,j）=k，表示进程 Pi 已占有 k 个单位的 Rj 类资源。

⑷需求矩阵 Need

Need（i,j）=k，表示进程 Pi 要完成工作还需要 k 个单位的 Rj 类资源，即剩余资源需求。

上述三个矩阵之间存在下述关系：（见图 1）

Need（i,j）= Max（i,j）— Allocation（i,j）

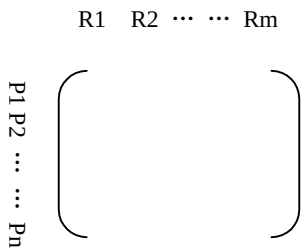

图 1：矩阵 Max、Allocation、Need

4. 算法流程

设 $Request_i$是当前运行的进程 Pi 的请求向量，若 $Request_i[j]=k$，表示进程 Pi 当前请求 k 个单位的 Rj 类资源。当进程 Pi 发出资源请求后，系统按下述步骤进行检查：

⑴若 $Request_i \geq Need[i]$，则为非法请求，进行相应处理，因为它需要的资源数超过它开始宣布的最大值；否则进行下一步。

⑵若 $Request_i \geq Available$，则表示系统现在可利用资源不能满足进程需求，阻塞进程；否则进行下一步。

⑶系统试探地把要求的资源分配给进程 Pi，即不进行实际的资源分配，而只是修改有关数据结构：

$Available=Available-Request_i$

$Allocation[i]=Allocation[i]+ Request_i$

$Need[i]=Need[i]- Request_i$

⑷执行安全性检测算法，检测此次资源分配后，系统是否处于安全状态。若安全，则正式将资源分配给进程 Pi，否则，将试探分配作废，恢复原来的资源分配状态，让进程 Pi 等待。

安全性检测算法

设置两个临时向量（又称为工作向量）Work、Finish，Work 表示系统当前可利用的各类资源数，初始值为 Available，目的是在 work 上进行模拟演算，而保存 Available 使其不被破坏。Finish 表示各进程能否运行完成。

```
/* 初始化工作向量 */
Work=Available;
for (i=1;i<n+1;i++)
 Finish[i]=FALSE；
/* 依次寻找能获得所需资源从而完成的进程，即求安全序列 */
for (j=1;j<n+1;j++)
for (i=1;i<n+1;i++)
  if (Need[i]<=Work&&!Finish[i])      /* 进程 Pi 能获得所需资源运行完成 */
   {
    Work=Work+Allocation[i]; /* 进程 Pi 完成后，释放所占有的资源 */
     Finish[i]=TRUE;     /* 修改完成标志 */
   }
/* 检测是否存在安全序列 */
c=0;
for (i=1;i<n+1;i++)
 if (Finish[i]) c++;
if (c==n)            /* 所有 Finish[i]=TRUE，则所有进程都能运行完成 */
 系统处于安全状态；
else
 系统处于不安全状态；
```

5. 银行家算法根本原理

⑴安全状态与不安全状态均指某一时刻的系统状态。安全状态并不意味着系统不会陷入死锁；不安全状态意味着系统可能陷入死锁，但是不一定。如果在系统的整个运行过程中，一直运用银行家算法保证系统始终处于安全状态，则肯定不会陷入死锁。否则，一个时刻的安全状态是毫无意义的，而且可能从安全状态转变成不安全状态。

⑵安全状态与不安全状态两个概念的引入使许多人产生误解，容易认为安全状态意味着系统不会陷入死锁，不安全状态意味着系统肯定会陷入死锁。其实，安全状态与不安全状态基于安全序列而定义，其实质是指是否存在一个安全序列。

安全序列的意义不在于从某时刻起系统按安全序列的顺序分配资源使各进程依次执行完毕。在某时刻确定的一个安全序列预示着，从此刻起，一定可以找到一种避免死锁的资源分配方案。一种特殊方案就是按照安全序列的顺序依次

满足各进程的后续资源需求，使各进程依次顺序执行完毕。

银行家算法基于这样一个原理：如果在系统运行的整个过程中，始终存在一种避免死锁的资源分配方案，则系统肯定能顺利推进直至所有进程都运行完毕。可以反证法来证明这一结论：如果系统陷入死锁，则肯定不存在一种避免死锁的资源分配方案。银行家算法的作用就是使系统始终存在一个安全序列，即始终存在一种资源分配方案。存在一种资源分配方案不意味着一定按此方案进行实际的资源分配。

⑶如果所有进程的最大资源需求都小于或等于系统资源总额，则在银行家算法的作用下，系统能始终保持安全状态（即时刻都有一个安全序列）。深入分析可知：系统的初始状态肯定是安全的。如果在某时刻存在着一个安全序列 S1，则意味着在银行家算法的作用下以后系统不会拒绝所有进程的资源请求而陷入死锁，总会有一个进程的资源请求能得到满足而继续推进，至少安全序列 S1 中的第一个进程的资源请求能得到满足（这点很容易证明），且该次资源分配后系统仍然保持安全状态（即存在一个安全序列 S2，S2 可能不同于上一个安全序列 S1）。依次递推，则系统会一直处于安全状态。

⑷如果在某时刻系统处于不安全状态，即不存在一个安全序列，则意味着到最后可能（不一定）有一些进程的资源请求永远得不到满足，相互之间循环等待而产生死锁。一个特例就是从当前时刻起，进程一次性申请全部剩余资源，而且一直保持当时已经占有的资源直至运行完毕才释放其占有的全部资源。在这种情况下，如果不存在一个安全序列，则最后肯定会有些进程陷入死锁。

⑸银行家算法是一种比较谨慎的资源分配方法。在银行家算法的作用下，如果满足当前进程 P 的资源请求后系统处于安全状态（即存在一个安全序列），那么就为 P 分配资源，否则拒绝 P 的资源请求。安全序列意味着一种资源分配方案，但不一定是唯一的资源分配方案，可能有另外的资源分配方案，只不过很难寻找。不存在安全序列并不意味着不存在一种可以避免死锁的资源分配方案。那么按银行家算法实施资源分配，就有可能拒绝一些不会导致死锁的资源请求，从而阻滞了某些进程的执行，而且降低了资源利用率。

其他的资源分配方案难找是因为无法预知各进程以后申请资源的情况：分多少次申请，每次申请各类资源的数量是多少。

参考文献：

[1] 汤子瀛 . 计算机操作系统 (修订版)[M]. 西安 : 西安电子科技大学出版社 ,2001

[2] 庞丽萍 . 操作系统原理 (第二版)[M]. 武汉 : 华中理工大学出版社 ,1994

Deep analysis of banker algorithm

HOU Gang

(Weifang University,Weifang 261061,China)

Abstract: Banker algorithm is one of algorithms on solving deadlock problem using avoiding deadlock policy. In this article, first, I descript the flow of banker algorithm, then, I analyze the basic principle of banker algorithm. In all textbook about operating system in the world, I analyze the basic principle of banker algorithm first.

Keywords: deadlock, avoiding deadlock, banker algorithm （责任编辑：肖恩忠）