

阵显示了使各进程完成运行所需的各种资源数。这些矩阵就是图6-6中的C和R。和一个资源的情况一样，各进程在执行前给出其所需的全部资源量，所以在系统的每一步中都可以计算出右边的矩阵。

图6-12最右边的三个向量分别表示现有资源E、已分配资源P和可用资源A。由E可知系统中共有6台磁带机、3台绘图仪、4台打印机和2台CD-ROM驱动器。由P可知当前已分配了5台磁带机、3台绘图仪、2台打印机和2台CD-ROM驱动器。该向量可通过将左边矩阵的各列相加获得，可用资源向量可通过从现有资源中减去已分配资源获得。

进程	磁带机	绘图仪	打印机	CD-ROM
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

已分配资源

进程	磁带机	绘图仪	打印机	CD-ROM
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

仍然需要的资源

$E = (6342)$   
 $P = (5322)$   
 $A = (1020)$

图6-12 多个资源的银行家算法

检查一个状态是否安全的算法如下：

1) 查找右边矩阵中是否有一行，其没有被满足的资源数均小于或等于A。如果不存在这样的行，那么系统将会死锁，因为任何进程都无法运行结束（假定进程会一直占有资源直到它们终止为止）。

2) 假若找到这样一行，那么可以假设它获得所需的资源并运行结束，将该进程标记为终止，并将其资源加到向量A上。

3) 重复以上两步，或者直到所有的进程都标记为终止，其初始状态是安全的；或者所有进程的资源需求都得不到满足，此时就是发生了死锁。

如果在第1步中同时有若干进程均符合条件，那么不管挑选哪一个运行都没有关系，因为可用资源或者会增多，或者至少保持不变。

图6-12中所示的状态是安全的，若进程B现在再请求一台打印机，可以满足它的请求，因为所得系统状态仍然是安全的（进程D可以结束，然后是A或E结束，剩下的进程相继结束）。

假设进程B获得两台可用打印机中的一台以后，E试图获得最后一台打印机，假若分配给E，可用资源向量会减到(1000)，这时会引起死锁。显然E的请求不能立即满足，必须延迟一段时间。

银行家算法最早由Dijkstra于1965年发表。从那之后几乎每本操作系统的专著都详细地描述它，很多论文的内容也围绕该算法讨论了它的不同方面。但很少有作者指出该算法虽然很有意义但缺乏实用价值，因为很少有进程能够在运行前就知道其所需资源的最大值。而且进程数也不是固定的，往往在不断地变化（如新用户的登录或退出），况且原本可用的资源也可能突然间变成不可用（如磁带机可能会坏掉）。因此，在实际中，如果有，也只有极少的系统使用银行家算法来避免死锁。

## 6.6 死锁预防

通过前面的学习我们知道，死锁避免从本质上来说是不可能的，因为它需要获知未来的请求，而这些请求是不可知的。那么实际的系统又是如何避免死锁的呢？我们回顾Coffman等人（1971）所述的四个条件，看是否能发现线索。如果能够保证四个条件中至少有一个不成立，那么死锁将不会产生（Havender, 1968）。

### 6.6.1 破坏互斥条件

先考虑破坏互斥使用条件。如果资源不被一个进程所独占，那么死锁肯定不会产生。当然，允许两个进程同时使用打印机会造成混乱，通过采用假脱机打印机（spooling printer）技术可以允许若干个进程同时产生输出。该模型中惟一真正请求使用物理打印机的进程是打印机守护进程，由于守护进程决不会请求别的资源，所以不会因打印机而产生死锁。

假设守护进程被设计为在所有输出进入假脱机之前就开始打印，那么如果一个输出进程在头一轮打印之后决定等待几个小时，打印机就可能空置。为了避免这种现象，一般将守护进程设计成在完整的输出文件就绪后才开始打印。例如，若两个进程分别占用了可用的假脱机磁盘空间的一半用于输出，而任何一个也没有能够完成输出，那么会怎样？在这种情形下，就会有两个进程，其中每一个都完成了部分的输出，但不是它们的全部输出，于是无法继续进行下去。没有一个进程能够完成，结果在磁盘上出现了死锁。