6.4.1 每种类型一个资源的死锁检测

我们从最简单的例子开始，即每种类型只有一个资源。这样的系统可能有一台扫描仪、一台CD刻录机、一台绘图仪和一台磁带机，但每种类型的资源都不超过一个，即排除了同时有两台打印机的情况。稍后我们将用另一种方法来解决两台打印机的情况。

可以对这样的系统构造一张资源分配图，如图6-3所示。如果这张图包含了一个或一个以上的环，那么死锁就存在。在此环中的任何一个进程都是死锁进程。如果没有这样的环，系统就没有发生死锁。

我们讨论一下更复杂的情况，假设一个系统包括A到G共7个进程，R到W共6种资源。资源的占有情况和进程对资源的请求情况如下：

1) A进程持有R资源，且需要S资源。

2) B进程不持有任何资源，但需要T资源。

3) C进程不持有任何资源，但需要S资源。

4) D进程持有U资源，且需要S资源和T资源。

5) E进程持有T资源，且需要V资源。

6) F进程持有W资源，且需要S资源。

7) G进程持有V资源，且需要U资源。

问题是："系统是否存在死锁？如果存在的话，死锁涉及了哪些进程？"

要回答这一问题，我们可以构造一张资源分配图，如图6-5a所示。可以直接观察到这张图中包含了一个环，如图6-5b所示。在这个环中，我们可以看出进程D、E、G已经死锁。进程A、C、F没有死锁，这是因为可把S资源分配给它们中的任一个，而且它们中的任一进程完成后都能释放S，于是其他两个进程可依次执行，直至执行完毕。（请注意，为了让这个例子更有趣，我们允许进程D每次请求两个资源。）

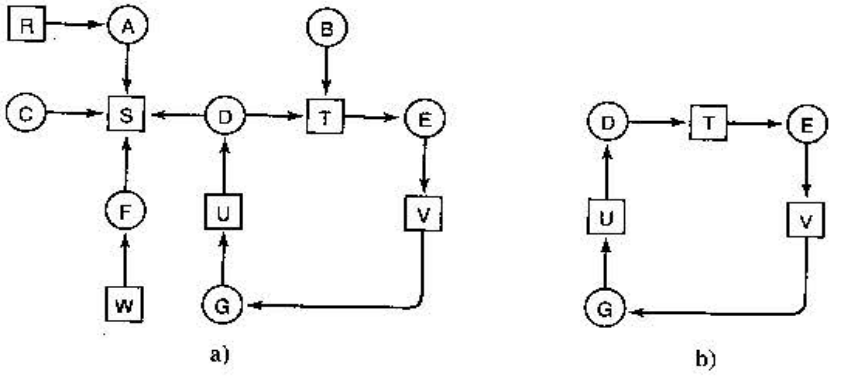

图6-5　a) 资源分配图；b) 从a中抽取的环

虽然通过观察一张简单的图就能够很容易地找出死锁进程，但为了实用，我们仍然需要一个正规的算法来检测死锁。众所周知，有很多检测有向图环路的方法。下面将给出一个简单的算法，这种算法对有向图进行检测，并在发现图中有环路存在或无环路时结束。这一算法使用了数据结构L，L代表一些节点的集合。在这一算法中，对已经检查过的弧（有向边）进行标记，以免重复检查。

通过执行下列步骤完成上述算法：

1) 对图中的每一个节点N，将N作为起始点执行下面5个步骤。

2) 将L初始化为空表，并清除所有的有向边标记。

3) 将当前节点添加到L的尾部，并检测该节点是否在L中已出现过两次。如果是，那么该图包含了一个环（已列在L中），算法结束。

4) 从给定的节点开始，检测是否存在没有标记的从该节点出发的弧（有向边）。如果存在的话，做第5步；如果不存在，跳到第6步。

5) 随机选取一条没有标记的从该节点出发的弧（有向边），标记它。然后顺着这条弧线找到新的当前节点，返回到第3步。

6) 如果这一节点是起始节点，那么表明该图不存在任何环，算法结束。否则意味着我们走进了死胡