旁。然后，该进程被挂起（标记为不可运行）。接着，打印机被分配给另一个进程。当那个进程结束后，堆在一旁的文档再被重新放回原处，原进程可重新继续工作。

在不通知原进程的情况下，将某一资源从一个进程强行取走给另一个进程使用，接着又送回，这种做法是否可行主要取决于该资源本身的特性。用这种方法恢复通常比较困难或者说不太可能。若选择挂起某个进程，则在很大程度上取决于哪一个进程拥有比较容易收回的资源。

2. 利用回滚恢复

如果系统设计人员以及主机操作员了解到死锁有可能发生，他们就可以周期性地对进程进行检查点检查（checkpointed）。进程检查点检查就是将进程的状态写入一个文件以备以后重启。该检查点中不仅包括存储映像，还包括了资源状态，即哪些资源分配给了该进程。为了使这一过程更有效，新的检查点不应覆盖原有的文件，而应写到新文件中。这样，当进程执行时，将会有一系列的检查点文件被累积起来。

一旦检测到死锁，就很容易发现需要哪些资源。为了进行恢复，要从一个较早的检查点上开始，这样拥有所需要资源的进程会回滚到一个时间点，在此时间点之前该进程获得了一些其他的资源。在该检查点后所做的所有工作都丢失。（例如，检查点之后的输出必须丢弃，因为它们还会被重新输出。）实际上，是将该进程复位到一个更早的状态，那时它还没有取得所需的资源，接着就把这个资源分配给一个死锁进程。如果复位后的进程试图重新获得对该资源的控制，它就必须一直等到该资源可用时为止。

3. 通过杀死进程恢复

最直接也是最简单的解决死锁的方法是杀死一个或若干个进程。一种方法是杀掉环中的一个进程。如果走运的话，其他进程将可以继续。如果这样做行不通的话，就需要继续杀死别的进程直到打破死锁环。

另一种方法是选一个环外的进程作为牺牲品以释放该进程的资源。在使用这种方法时，选择一个要被杀死的进程要特别小心，它应该正好持有环中某些进程所需的资源。比如，一个进程可能持有一台绘图仪而需要一台打印机，而另一个进程可能持有一台打印机而需要一台绘图仪，因而这两个进程是死锁的。第三个进程可能持有另一台同样的打印机和另一台同样的绘图仪而且正在运行着。杀死第三个进程将释放这些资源，从而打破前两个进程的死锁。

有可能的话，最好杀死可以从头开始重新运行而且不会带来副作用的进程。比如，编译进程可以被重复运行，由于它只需要读入一个源文件和产生一个目标文件。如果将它中途杀死，它的第一次运行不会影响到第二次运行。

另一方面，更新数据库的进程在第二次运行时并非总是安全的。如果一个进程将数据库的某个记录加1，那么运行它一次，将它杀死后，再次执行，就会对该记录加2，这显然是错误的。

6.5 死锁避免

在讨论死锁检测时，我们假设当一个进程请求资源时，它一次就请求所有的资源（见图6-6中的矩阵R）。不过在大多数系统中，一次只请求一个资源。系统必须能够判断分配资源是否安全，并且只能在保证安全的条件下分配资源。问题是：是否存在一种算法总能做出正确的选择从而避免死锁？答案是肯定的，但条件是必须事先获得一些特定的信息。本节我们会讨论几种死锁避免的方法。

6.5.1 资源轨迹图

避免死锁的主要算法是基于一个安全状态的概念。在描述算法前，我们先讨论有关安全的概念。通过图的方式，能更容易理解。虽然图的方式不能被直接翻译成有用的算法，但它给出了一个解决问题的直观感受。

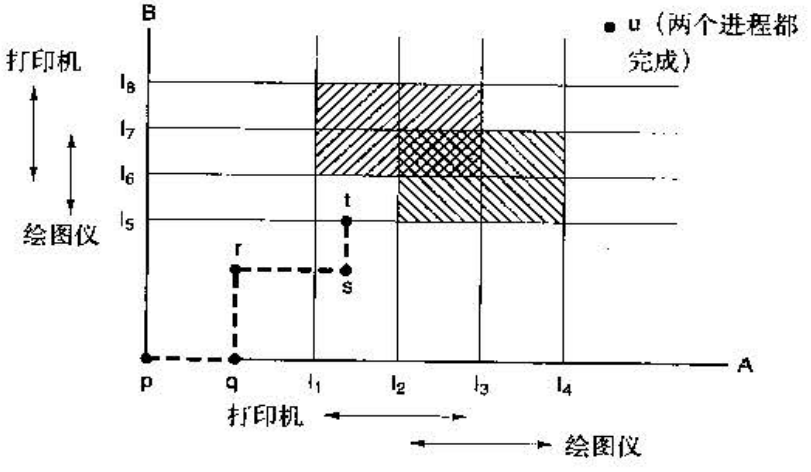

图6-8 两个进程的资源轨迹图

在图6-8中，我们看到一个处理两