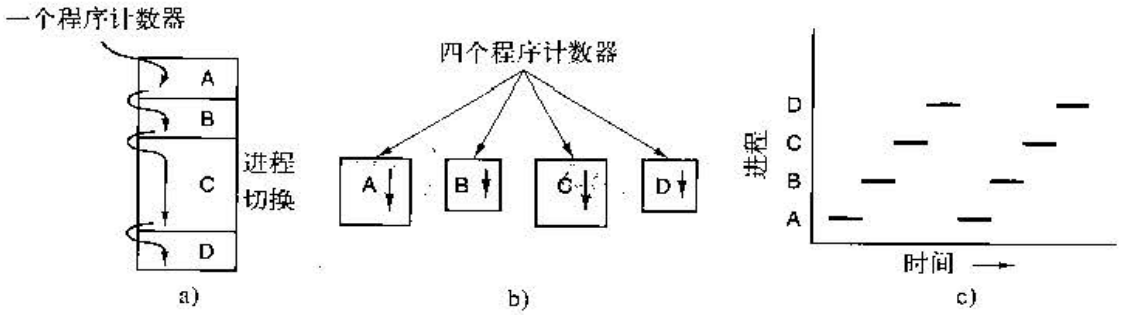

图2-1 a) 含有4道程序的多道程序；b) 4个独立的顺序进程的概念模型；c) 在任意时刻仅有一个程序是活跃的

在本章，我们假设只有一个CPU。然而，逐渐这个假设就不为真了，因为新的芯片经常是多核的，包含2个、4个或更多的CPU。我们将会在第8章介绍多核芯片以及多处理器，但是在现在，一次只考虑一个CPU会更简单一些。因此，当我们说一个CPU只能真正一次运行一个进程的时候，即使有2个核（或CPU），每一个核也只能一次运行一个进程。

由于CPU在各进程之间来回快速切换，所以每个进程执行其运算的速度是不确定的。而且当同一进程再次运行时，其运算速度通常也不可再现。所以，在对进程编程时决不能对时序做任何确定的假设。例如，考虑一个I/O进程，它用流式磁带机恢复备份的文件，它执行一个10 000次的空循环以等待磁带机达到正常速度，然后发出命令读取第一个记录。如果CPU决定在空循环期间切换到其他进程，则磁带机进程可能在第一条记录通过磁头之后还未被再次运行。当一个进程具有此类严格的实时要求时，也就是一些特定事件一定要在所指定的若干毫秒内发生，那么必须采取特殊措施以保证它们一定在这段时间中发生。然而，通常大多数进程并不受CPU多道程序设计或其他进程相对速度的影响。

进程和程序间的区别是很微妙的，但非常重要。用一个比喻可以使我们更容易理解这一点。想象一位有一手好厨艺的计算机科学家正在为他的女儿烘制生日蛋糕。他有做生日蛋糕的食谱，厨房里有所需的原料：面粉、鸡蛋、糖、香草汁等。在这个比喻中，做蛋糕的食谱就是程序（即用适当形式描述的算法），计算机科学家就是处理器（CPU），而做蛋糕的各种原料就是输入数据。进程就是厨师阅读食谱、取来各种原料以及烘制蛋糕等一系列动作的总和。

现在假设计算机科学家的儿子哭着跑了进来，说他的头被一只蜜蜂蜇了。计算机科学家就记录下他照着食谱做到哪儿了（保存进程的当前状态），然后拿出一本急救手册，按照其中的指示处理蜇伤。这里，我们看到处理机从一个进程（做蛋糕）切换到另一个高优先级的进程（实施医疗救治），每个进程拥有各自的程序（食谱和急救手册）。当蜜蜂蜇伤处理完之后，这位计算机科学家又回来做蛋糕，从他离开时的那一步继续做下去。

这里的关键思想是：一个进程是某种类型的一个活动，它有程序、输入、输出以及状态。单个处理器可以被若干进程共享，它使用某种调度算法决定何时停止一个进程的工作，并转而为另一个进程提供服务。

值得注意的是，如果一个程序运行了两遍，则算作两个进程。例如，我们可能经常两次去启动同一个字处理软件，或在有两个可用的打印机的情况下同时打印两个文件。像“两个进程恰好运行同一个程序”这样的事实其实无关紧要，因为它们是不同的进程。操作系统能够使它们共享代码，因此只有一个副本放在内存中，但那只是一个技术性的细节，不会改变有两个进程正在运行的概念。

2.1.2 创建进程

操作系统需要有一种方式来创建进程。一些非常简单的系统，即那种只为运行一个应用程序设计的系统（例如，微波炉中的控制器），可能在系统启动之时，以后所需要的所有进程都已存在。然而在通用系统中，需要有某种方法在运行时按需要创建或撤销进程。我们现在开始考察这个问题。

有4种主要事件导致进程的创建：

1) 系统初始化。

2) 执行了正在运行的进程所调用的进程创建系统调用。

3) 用户请求创建一个新进程。