实上，支持同时运行使用不同操作系统的应用程序是赞成虚拟机技术的一个重要理由。

同时，虚拟机的一个重要应用是软件开发。一个程序员想要确保他的软件在Windows 98、Windows 2000、Windows XP、Windows Vista、多种Linux版本、FreeBSD、OpenBSD、NetBSD和Mac OS X上都可以正常运行，他不需要有一打的计算机，以及在不同的计算机上安装不同的操作系统。相反，他只需要在一台物理机上创建一些虚拟机，然后在每个虚拟机上安装不同的操作系统。当然，这个程序员可以给他的磁盘分区，然后在每个分区上安装不同的操作系统，但是这种方法太过困难。首先，不论磁盘的容量有多大，标准的PC机只支持四个主分区。其次，尽管在引导块上可以安装一个多引导程序，但要运行另一个操作系统就必须重启计算机。使用虚拟机，所有的操作系统可以同时运行，因为它们都只是美妙的进程。

8.3.1 虚拟化的条件

我们在第1章中看到，有两种虚拟化的方法。一种管理程序（hypervisor），又称为I型管理程序（或虚拟机监控器），如图1-29a所示。实质上，它就是一个操作系统，因为它是惟一一个运行在内核态的程序。它的工作是支持真实硬件的多个副本，也称作虚拟机（virtual machine），与普通操作系统所支持的进程类似。相反，II型管理程序，如图1-29b所示，是一种完全不同的类型。它只是一个运行在诸如Windows或Linux平台上，能够“解释”机器指令集的用户程序，它也创建了一个虚拟机。我们把“解释”二字加上引号是因为通常代码块是以特殊的方式进行处理然后缓存并且直接执行从而获得性能上的提升，但是在原理上，完全解释也是可行的，虽然速度很慢。两种情况下，运行在管理程序上的操作系统都称为客户操作系统（guest operating system）。在II型管理程序的情况下，运行在硬件上的操作系统称为宿主操作系统（host operating system）。

在两种情况下，虚拟机都必须像真实机器一样工作，认识到这一点非常重要。也就是说，必须能够像真实机器那样启动虚拟机，像真实的机器那样在其上安装任意的操作系统。管理程序的任务就是提供这种错觉，并且尽量高效（不能完全解释执行）。

虚拟机有两种类型的原因与Intel 386体系结构的缺陷有关，而这些缺陷在20年间以向后兼容的名义被盲目地不断推进到新的CPU中。简单地说，每个有内核态和用户态的处理器都有一组只能在内核态执行的指令集合，比如I/O指令、改变MMU状态的指令等。Popek和Goldberg（1974）两人在他们的经典虚拟化工作中称这些指令为敏感指令（sensitive instruction）。还有一些指令如果在用户态下执行会引起陷入。Popek和Goldberg称它们是特权指令（privileged instruction）。在他们的论文中首次论述指出，当且仅当敏感指令是特权指令的子集时，机器才是可虚拟化的。简单地说，如果你想做一些在用户态下不能做的工作，硬件应该陷入。IBM/ 370具有这种特性，但是与它不同，386体系结构不具有这种特性。有一些敏感的386指令如果在用户态下执行就会被忽略。举例来说，POPF指令替换标志寄存器，会改变允许/禁止中断的标志位。但是在用户态下，这个标志位不被改变。所以，386体系结构和它的后代都是不可虚拟化的，也就是说它们不能支持I型管理程序。

事实上，情况比上面描述的还要更糟糕一些。除了某些指令在用户态不能陷入之外，还有一些指令可以在用户态读取敏感状态而不引起陷入。比如，在Pentium处理器上，一个程序可以读取代码段选择子（selector）的值从而判断它是运行在用户态还是内核态上。如果一个操作系统做同样的事情，然后发现它运行在用户态，那么就可能据此作出不正确的判断。

从2005年开始，Intel和AMD公司在它们的处理器上引进了虚拟化技术，从而使问题得到了解决。在Intel Core 2CPU上，这种技术称为VT（Virtualization Technology）。在AMD Pacific CPU上，这种技术称为SVM（Secure Virtual Machine）。在下文里，我们一般使用VT这个词来代表。它们的灵感都来自于IBM VM/370，但是也有一些细微的不同之处。基本的思想是创建容器使得虚拟机可以在其内运行。当一个客户操作系统在一个容器内启动，它将继续运行直到它引发了异常而陷入到管理程序。例如，执行一条I/O指令。陷入操作由管理程序通过硬件位图集来管理。有了这些扩展，经典的“陷入-仿真”类型的虚拟化方法才成为可能。

8.3.2 I型管理程序

可虚拟化是一个重要的问题，所以让我们来更仔细地研究一下。在图8-26中，我们可以看到一个支