要有100卷书——基本上需要一整个书架来摆放。请设想一下有个维护操作系统的工作，第一天老板带你到装有代码的书架旁，说："去读吧。"而这仅仅是运行在内核中的部分代码。用户程序，如GUI、库以及基本应用软件（类似于Windows Explorer）等，很容易就能达到这个代码数量的10倍或20倍之多。

至于为什么操作系统的寿命较长，读者现在应该清楚了——操作系统是很难编写的。一旦编写完成，操作系统的所有者当然不愿意把它扔掉，再写一个。相反，操作系统会在长时间内进行演化。基本上可以把Windows 95/98/Me看成是一个操作系统，而Windows NT/2000/XP/Vista则是另外一个操作系统。对于用户而言，它们看上去很相像，因为微软公司努力使Windows 2000/XP与被替代的系统，如Windows 98，两者的用户界面看起来十分相似。无论如何，微软公司要舍弃Windows 98是有非常正当的原因的，我们将在第11章涉及Windows细节时具体讨论这一内容。

贯穿本书的其他主要例子（除了Windows）还有UNIX，以及它的变体和克隆版。UNIX，当然也演化了多年，如System V版、Solaris以及FreeBSD等都是来源于UNIX的原始版；不过尽管Linux非常像依照UNIX模式而仿制，并且与UNIX高度兼容，但是Linux具有全新的代码基础。本书将采用来自UNIX中的示例，并在第10章中具体讨论Linux。

本章将简要叙述操作系统的若干重要部分，内容包括其含义、历史、分类、一些基本概念及其结构。在后面的章节中，我们将具体地讨论这些重要内容。

1.1 什么是操作系统

很难给出操作系统的准确定义。操作系统是一种运行在内核态的软件——尽管这个说法并不总是符合事实。部分原因是操作系统执行两个基本上独立的任务，为应用程序员（实际上是应用程序）提供一个资源集的清晰抽象，并管理这些硬件资源，而不仅仅是一堆硬件。另外，还取决于从什么角度看待操作系统。读者多半听说过其中一个或另一个的功能。下面我们逐项进行讨论。

1.1.1 作为扩展机器的操作系统

在机器语言一级上，多数计算机的体系结构（指令集、存储组织、I/O和总线结构）是很原始的，而且编程是很困难的，尤其是对输入/输出操作而言。要更细致地考察这一点，可以考虑如何用NEC PD765控制器芯片来进行软盘I/O操作，多数基于Intel的个人计算机中使用了该控制器兼容芯片。（在本书中，术语"软盘"和"磁盘"是可互换的。）我们之所以使用软盘作为例子，是因为它虽然已经很少见，但是与现代硬盘相比则简单得多。PD765有16条命令，每一条命令向一个设备寄存器装入长度从1字节到9字节的特定数据。这些命令用于读写数据、移动磁头臂、格式化磁道，以及初始化、检测状态、复位、校准控制器及设备等。

最基本的命令是read和write。它们均需要13个参数，所有这些参数封装在9个字节中。这些参数所指定的信息有：欲读取的磁盘块地址、磁道的扇区数、物理介质的记录格式、扇区间隙以及对已删除数据地址标识的处理方法等。如果读者不懂这些"故弄玄虚"的语言，请不要担心，因为这正是关键所在——它们太玄秘了。当操作结束时，控制器芯片在7个字节中返回23个状态及出错字段。这样似乎还不够，软盘程序员还要注意保持步进电机的开关状态。如果电机关闭着，则在读写数据前要先启动它（有一段较长的启动延迟时间）。而电机又不能长时间处于开启状态，否则软盘片就会被磨坏。程序员必须在较长的启动延迟和可能对软盘造成损坏（和丢失数据）之间做出权衡。

现在不用再叙述读操作的具体过程了，很清楚，一般程序员并不想涉足软盘（或硬盘，更复杂）编程的这些具体细节。相反，程序员需要的是一种简单的、高度抽象的处理。在磁盘的情况下，典型的抽象是包含了一组已命名文件的一个磁盘。每个文件可以打开进行读写操作，然后进行读写，最后关闭文件。诸如记录是否应该使用修正的调频记录方式，以及当前电机的状态等细节，不应该出现在提供给应用程序员的抽象描述中。

抽象是管理复杂性的一个关键。好的抽象可以把一个几乎不可能管理的任务划分为两个可管理的部分。其第一部分是有关抽象的定义和实现，第二部分是随时用这些抽象解决问题。几乎每个计算机用户都理解的一个抽象是文件。文件是一种有效的信息片段，诸如数码照片、保存的电子邮件信息或Web页面等。处理数码照片、电子邮件以及Web页面等，要比处理磁盘的细节容易，这些磁盘的具体细节与前面叙述过的软盘一样。操作系统的任务是创建好的抽象，并实现和管理它所创建的抽象对象。本书中，