想要的是渔船、游船还是战舰，并且当产品生产出来之后鲜有人会改变产品的用途。

第七，现代操作系统一般被设计成可移植的，这意味着它们必须运行在多个硬件平台上。它们还必须支持上千个I/O设备，所有这些I/O设备都是独立设计的，彼此之间没有关系。这样的差异可能会导致问题，一个例子是操作系统需要运行在小端机器和大端机器上。第二个例子经常在MS-DOS下看到，用户试图安装一块声卡和一个调制解调器，而它们使用了相同的I/O端口或者中断请求线。除了操作系统以外，很少有程序必须处理由于硬件部件冲突而导致的这类问题。

第八，也是最后一个问题，是经常需要与某个从前的操作系统保持向后兼容。以前的那个系统可能在字长、文件名或者其他方面有所限制，而在设计人员现在看来这些限制都是过时的，但是却必须坚持。这就像让一家工厂转而去生产下一年的汽车而不是这一年的汽车的同时，继续全力地去生产这一年的汽车。

13.2 接口设计

到现在读者应该清楚，编写一个现代操作系统并不容易。但是人们要从何处开始呢？可能最好的起点是考虑操作系统提供的接口。操作系统提供了一组抽象，主要是数据类型（例如文件）以及其上的操作（例如read）。它们合起来形成了对用户的接口。注意，在这一上下文中操作系统的用户是指编写使用系统调用的代码的程序员，而不是运行应用程序的人员。

除了主要的系统调用接口，大多数操作系统还具有另外的接口。例如，某些程序员需要编写插入到操作系统中的设备驱动程序。这些驱动程序可以看到操作系统的某些功能特性并且能够发出某些过程调用。这些功能特性和调用也定义了接口，但是与应用程序员看到的接口完全不同。如果一个系统要取得成功，所有这些接口都必须仔细地设计。

13.2.1 指导原则

有没有指导接口设计的原则？我们认为是有的。简而言之，原则就是简单、完备和能够有效地实现。

原则1：简单

一个简单的接口更加易于理解并且更加易于以无差错的方式实现。所有的系统设计人员都应该牢记法国先驱飞行家和作家Antoine de St. Exupéry的著名格言：

不是当没有东西可以再添加，而是当没有东西可以再裁减时，才能达到尽善尽美。

这一原则说的是少比多好，至少在操作系统本身中是这样。这一原则的另一种说法是KISS原则：Keep It Simple, Stupid（保持简朴无华）。

原则2：完备

当然，接口必须能够做用户需要做的一切事情，也就是说，它必须是完备的。这使我们想起了另一条著名的格言，Albert Einstein（阿尔伯特·爱因斯坦）说过：

万事都应该尽可能简单，但是不能过于简单。

换言之，操作系统应该不多不少准确地做它需要做的事情。如果用户需要存储数据，它就必须提供存储数据的机制；如果用户需要与其他用户通信，操作系统就必须提供通信机制；如此等等。1991年，CTSS和MULTICS的设计者之一Fernando Corbató在他的图灵奖演说中，将简单和完备的概念结合起来并且指出：

首先，重要的是强调简单和精练的价值，因为复杂容易导致增加困难并且产生错误。正如我们已经看到的那样。我对精练的定义是以机制的最少化和清晰度的最大化实现指定的功能。

此处重要的思想是机制的最少化（minimum of mechanism）。换言之，每一个特性、功能和系统调用都应该尽自己的本分。它应该做一件事情并且把它做好。当设计小组的一名成员提议扩充一个系统调用或者添加某些新的特性时，其他成员应该问这样的问题："如果我们省去它会不会发生可怕的事情？"如果回答是："不会，但是有人可能会在某一天发现这一特性十分有用"，那么请将其放在用户级的库中，而不是操作系统中，尽管这样做可能会使速度慢一些。并不是所有的特性都要比高速飞行的子弹还要快。目标是保持Corbató所说的机制的最少化。

让读者简略地看一看我亲身经历的两个例子：MINIX（Tanenbaum和Woodhull，2006）和Amoeba（Tanenbaum等人，1990）。实际上，MINIX具有三个系统调用：send、receive和sendrec。系统是作为一组进程的集合而构造的，内存管理、文件系统以及每个设备驱动程序都是单独的可调度的进程。作为