件，多年以前就已经提出来了，似乎比较显而易见。其他一些抽象概念，例如线程，还比较新鲜，就不那么成熟了。例如，如果一个多线程的进程有一个线程由于等待键盘输入而阻塞，那么由这个进程通过调用fork函数创建的新进程是否也包含一个等待键盘输入的线程？其他的抽象概念涉及同步、信号、内存模型、I/O的建模以及其他领域。

每一个抽象概念可以采用具体数据结构的形式实例化。用户可以创建进程、文件、信号量等。基本操作则处理这些数据结构。例如，用户可以读写文件。基本操作以系统调用的形式实现。从用户的观点来看，操作系统的核心是由抽象概念和其上的基本操作所构成的，而基本操作则可通过系统调用加以利用。

由于多个用户可以同时登录到一台计算机，操作系统需要提供机制将他们隔离。一个用户不可以干扰另一个用户。为了保护的目的，进程概念广泛地用于将资源集合在一起。文件和其他数据结构一般也是受保护的。确保每个用户只能在授权的数据上执行授权的操作是系统设计的关键目标。然而，用户还希望共享数据和资源，因此隔离必须是选择性的并且要在用户的控制之下。这就使问题更加复杂化了。电子邮件程序不应该弄坏Web浏览器程序，即使只有一个用户，不同的进程也应该隔离开来。

与这一要点密切相关的是需要隔离故障。如果系统的某一部分崩溃（最为一般的是一个用户进程崩溃），不应该使系统的其余部分随之崩溃。系统设计应该确保系统的不同部分良好地相互隔离。从理想的角度看，操作系统的各部分也应该相互隔离，以便使故障独立。

最后，操作系统必须管理硬件。特别地，它必须处理所有低级芯片，例如中断控制器和总线控制器。它还必须提供一个框架，从而使设备驱动程序得以管理更大规模的I/O设备，例如磁盘、打印机和显示器。

13.1.2 设计操作系统为什么困难

摩尔定律表明计算机硬件每十年改进100倍，但却没有一个定律宣称操作系统每十年改进100倍。甚至没有人能够宣称操作系统每十年在某种程度上会有所改善。事实上，可以举出事例，一些操作系统在很多重要的方面（例如可靠性）比20世纪70年代的UNIX版本7还要糟糕。

为什么会这样？大部分责任常常归咎于惯性和渴望向后兼容，不能坚持良好的设计原则也是问题的根源。但是还不止这些。操作系统在特定的方面根本不同于计算机商店以49美元销售的小型应用程序。我们下面就看一看使设计一个操作系统比设计一个应用程序要更加困难的8个问题。

第一，操作系统已经成为极其庞大的程序。没有一个人能够坐在一台PC机前在几个月内匆匆地完成一个严肃的操作系统。UNIX的所有当前版本都超过了300万行代码，Windows Vista有超过500万行的内核代码（全部代码超过7亿行）。没有一个人能够理解300万到500万行代码，更不必说7亿行代码。当你拥有一件产品，如果没有一名设计师能够有望完全理解它时，结果经常远没有达到最优也就不难预料了。

操作系统不是世界上最复杂的系统，例如，航空母舰就要复杂得多，但是航空母舰能够更好地分成相互隔离的部分。设计航空母舰上的卫生间的人员根本不必关心雷达系统，这两个子系统没有什么相互作用。而在操作系统中，文件系统经常以意外的和无法预料的方式与内存系统相互作用。

第二，操作系统必须处理并发。系统中往往存在多个用户和多个设备同时处于活动状态。管理并发自然要比管理单一的顺序活动复杂得多。竞争条件和死锁只是出现的问题中的两个。

第三，操作系统必须处理可能有敌意的用户——想要干扰系统的用户或者做不允许做的事情（例如偷窃另一个用户的文件）的用户。操作系统需要采取措施阻止这些用户不正当的行为，而字处理程序和照片编辑程序就不存在这样的问题。

第四，尽管事实上并非所有的用户都相信其他用户，但是许多用户确实希望与经过选择的其他用户共享他们的信息和资源。操作系统必须使其成为可能，但是要以确保怀有恶意的用户不能妨害的方式。而应用程序就不会面对类似这样的挑战。

第五，操作系统已经问世很长时间了。UNIX已经历了四分之一个世纪，Windows面世也已经超过二十年并且还没有消退的迹象。因此，设计人员必须思考硬件和应用程序在遥远的未来可能会发生的变化，并且考虑为这样的变化做怎样的准备。紧密地局限于世界的一个特定视野的系统通常不会存世太久。

第六，操作系统设计人员对于他们的系统将怎样被人使用实际上并没有确切的概念，所以他们需要提供相当程度的通用性。UNIX和Windows在设计时都没有把电子邮件或Web浏览器放在心上，然而许多运行这些系统的计算机却很少做其他的事情。人们在告诉一名轮船设计师建造一艘轮船时，却会指明他