拥有这些新技术的典型机器是CDC 6600，该机器于1964年发布，在多年之内始终是世界上最快的计算机。用户可以通过指定名称的方式创建所谓“永久文件”，希望这个名称还没有被别人使用，比如“data”就是一个适合于文件的名称。这个系统使用单层目录。后来在大型机上开发出了复杂的多层文件系统，MULTICS文件系统可以算是多层文件系统的顶峰。

接着小型计算机投入使用，该机型最后也有了硬盘。1970年在PDP-11上引入了标准硬盘，RK05磁盘，容量为2.5MB，只有IBM RAMAC一半的容量，但是这个磁盘的直径只有40厘米，5厘米高。不过，其原型也只有单层目录。随着微型计算机的出现，CP/M开始成为操作系统的主流，但是它也只是在（软）盘上支持单目录。

4. 虚拟内存

虚拟内存（安排在第3章中讨论），通过在RAM和磁盘中反复移动信息块的方式，提供了运行比机器物理内存大的程序能力。虚拟内存也经历了类似的历程，首先出现在大型机上，然后是小型机和微型机。虚拟内存还使得程序可以在运行时动态地链接库，而不是必须在编译时链接。MULTICS是第一个可以做到这点的系统。最终，这个思想传播到所有的机型上，现在广泛用于多数UNIX和Windows系统中。

在所有这些发展过程中，我们看到，在一种环境中出现的思想，随着环境的变化被抛弃（汇编语言设计，单道程序处理，单层目录等），通常在十年之后，该思想在另一种环境下又重现了。由于这个原因，本书中，我们将不时回顾那些在今日的G字节PC机中过时的思想和算法，因为这些思想和算法可能会在嵌入式计算机和智能卡中再现。

1.6 系统调用

我们已经看到操作系统具有两种功能：为用户程序提供抽象和管理计算机资源。在多数情形下，用户程序和操作系统之间的交互处理的是前者，例如，创建、写入、读出和删除文件。对用户而言，资源管理部分主要是透明和自动完成的。这样，用户程序和操作系统之间的交互主要就是处理抽象。为了真正理解操作系统的行为，我们必须仔细地分析这个接口。接口中所提供的调用随着操作系统的不同而变化（尽管基于的概念是类似的）。

这样我们不得不在如下的可能方式中进行选择：（1）含混不清的一般性叙述（“操作系统提供读取文件的系统调用”）；（2）某个特定的系统（“UNIX提供一个有三个参数的read系统调用：一个参数指定文件，一个说明数据应存放的位置，另一个说明应读出多少个字节”）。

我们选择后一种方式。这种方式需要更多的努力，但是它能更多地洞察操作系统具体在做什么。尽管这样的讨论会涉及专门的POSIX（International Standard 9945-1），以及UNIX、System V、BSD、Linux、MINIX3等，但是多数现代操作系统都有实现相同功能的系统调用，尽管它们在细节上差别很大。由于引发系统调用的实际机制是非常依赖于机器的，而且必须用汇编代码表达，所以，通过提供过程库使C程序中能够使用系统调用，当然也包括其他语言。

记住下列事项是有益的。任何单CPU计算机一次只能执行一条指令。如果一个进程正在用户态中运行一个用户程序，并且需要一个系统服务，比如从一个文件读数据，那么它就必须执行一个陷阱或系统调用指令，将控制转移到操作系统。操作系统接着通过参数检查，找出所需要的调用进程。然后，它执行系统调用，并把控制返回给在系统调用后面跟随着的指令。在某种意义上，进行系统调用就像进行一个特殊的过程调用，但是只有系统调用可以进入内核，而过程调用则不能。

为了使系统调用机制更清晰，让我们简要地考察read系统调用。如上所述，它有三个参数：第一个参数指定文件，第二个指向缓冲区，第三个说明要读出的字节数。几乎与所有的系统调用一样，它的调用由C程序完成，方法是调用一个与该系统调用名称相同的库过程：read。由C程序进行的调用可有如下形式：

```
count = read(fd, buffer, nbytes);
```

系统调用（以及库过程）在count中返回实际读出的字节数。这个值通常和nbytes相同，但也可能更小，例如，如果在读过程中遇到了文件尾的情形就是如此。