Chapter 1

引论

现代计算机系统由一个或多个处理器、主存、磁盘、打印机、键盘、鼠标、显示器、网络接口以及各种其他输入/输出设备组成。一般而言，现代计算机系统是一个复杂的系统。如果每位应用程序员都不得不掌握系统的所有细节，那就不可能再编写代码了。而且，管理这些部件并加以优化使用，是一件挑战性极强的工作。所以，计算机安装了一层软件，称为操作系统，它的任务是为用户程序提供一个更好、更简单、更清晰的计算机模型，并管理刚才提到的所有设备。本书的祖逖就是操作系统。

多数读者都会对诸如 Windows、Linux、FreeBSD 或 OX X 等某个操作系统有些体验，但表面现象是会骗人的。用户与之交互的程序，基于文本的通常称为 shell，而基于图标的则称为图形用户界面（Graphical User Interface，GUI），它们实际上并不是操作系统的一部分，尽管这些程序使用操作系统来完成工作。

图给出了这里所讨论的主要部件的一个简化视图。图的底部是硬件。硬件包括芯片、电路板、磁盘、键盘、显示器以及类似的设备。在硬件的顶层是软件。多数计算机有两种运行模式：内核态和用户态。软件中最基础的部分是操纵系统，它运行在内核态（也称为管态、和心态）。在这个模式中，操嘴系统具有对所有硬件的完全访问权，可以执行机器能够运行的任何指令。软件的其余部分运行在用户态下。在用户态下，只是用了机器指令中的一个子集。特别地，那些会影响机器的控制或可进行 I/O 操作的指令，在用户态中的程序里是禁止的。本书中，我们会不断地讨论内核态和用户态之间的差别，这些差别在操作系统的运行中扮演着极其重要的角色。

用户接口程序（shell 或者 GUI）处于用户态程序中的最低层次，允许用户运行其他程序，诸如 web 浏览器、电子邮件阅读器或音乐播放器等。这些程序也大量使用操作系统。

操嘴系统所在的位置如图所示。它运行在逻辑之上，为所有其他软件提供基础的运行环境。

操作系统和普通软件（用户态）之间的主要区别是，如果用户不喜欢某个特定的电子邮件阅读器，他可以自由选择另一个，或者自己写一个，但是不能自己写一个属于操作系统一部分的时钟中断处理程序。这个程序由硬件保护，防止用户试图对其进行修改。

然而，有时在嵌入式系统（该系统没有内核态）或解释系统（如基于 Java 的操作系统，它采用解释方式而非硬件方式区分组件）中，上述区别是模糊的。

另外，在许多系统中，一些在用户态下运行的程序协助操嘴系统完成特权功能。例如，经常有一个程序供用户修改其口令之用。但是这个程序不是操作系统的一部分，也不在内核态下运行，不过它明显地带有敏感功能，并且必须以某种方式给予保护。在有些系统中，这种想法被推向了极致，一些传统上被认为是操嘴系统的部分（诸如文件系统）在用户空间中运行。在这类系统中，很难划分出一条明显的界限。在内核态中运行的当然是操作系统的一部分，但是一些在内核外运行的程序也有争议地被认为是操作系统的一部分，或者至少与操作系统密切相关。

操作系统与用户（即应用）程序的差异并不仅仅在于他们所处的地位。特别地，操作系统更加大型、复杂和长寿。Linux 或 Windows 操作胸膛呢搞得源代码有 500 万行甚至更高的数量级。要理解这个数量的含义，请考虑具有 500 万行的一套书，每页 50 行，每卷 1000 页。为了比书的大小列出一个操作系统，需要 100 卷的书——基本上需要一整个书架来摆放。

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 看成一个操作系统，而 Windows NT/2000/XP/Vista/Windows 7 则是另外一个操作系统。对于用户而言，它们看上去很像，因为微软公司努力使 Windows 2000/XP/Vista/Windows 7 与被替代系统的用户界面看起来十分相似。无论如何，微软公司要放弃 Windows 98 是有非常正当的原因的，我们将在第 11 章 Windows 细节时具体讨论这一内容。

除了 Windows 以外，贯穿本书的其他主要例子还有 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 非常想依照 UNIX 模式仿制而成，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、一些基本概念以及结构。在后面的章节中，我们将具体讨论这些重要内容。

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有两个基本上独立的任务，即为应用程序员（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。为了更细致地考察这一点，我们以大多数电脑使用的更现代的 SATA（Serial ATA）硬件为例。曾有一本描述早期版本硬件接口的书（Anderson，2007），它的页数超过 450 页。自 2007 年起，接口又被修改过很多次，因而比当时更为复杂。显然，没有任何理智的程序员想要在硬件层面上与硬件打交道。相反，他们使用

了一些叫作硬盘驱动（disk driver）的软件来和硬件交互。这类软件提供了读写硬件块的接口，而不用深入细节。操作系统包含了很多用于控制输入/输出设备的驱动。

但是就算是在这个层面，对于大多数应用来说还是太底层了。因此，所有的操作系统都提供了使用硬盘的又一层抽象：文件。使用该抽象，程序能创建、读写文件，而不用处理硬件实际工作中那些恼人的细节。

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。其第一部分是有关于抽象地定义和实现，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一个抽象是文件，正如上文所提到的。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、歌曲或 web 页面等。处理数码照片、电子邮件、歌曲以及 web 页面等，要比处理 SATA 硬盘的细节容易，这些磁盘的具体细节与前面叙述过的硬盘一样。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。本书中，我们将研究许多有关抽象地内容，因为这是理解操作系统的关键。

上述观点是非常重要的，所以值得用不同的表达方式再次叙述。

在本书中，我们将具体讨论提供给应用程序的抽象，不过很少涉及用户界面。尽管用户界面是一个巨大和重要的课题，但是它们毕竟只和操作系统的外围有关。

1.1.2 作为资源管理者的操作系统

把操作系统看作向应用程序提供基本抽象地概念，是一种自顶向下的观点，按照另一种自底向上的观点，操作系统则用来管理一个复杂系统的各个部分。现代计算机包含处理器、存储器、时钟、磁盘、鼠标、网络接口、打印机以及许多其他设备。从这个角度看，操作系统的任务是在相互竞争的程序之间有序地控制对处理器、存储器以及其他 I/O 接口设备的分配。

现代操作系统允许同时在内存中运行多道程序。假设在一台计算机上运行的三个程序试图同时在同一台打印机上输出计算结果，那么开始的几行可能是程序 1 的输出，接着几行是程序 2 的输出，然后又是程序 3 的输出等，最终结果将是一团糟。采用将打印结果宋代磁盘上缓冲区的方法，操作系统可以把潜在的混乱有序化。在一个程序结束后，操作系统可以将暂存在磁盘上的文件送到打印机输出，同时其他程序可以继续产生更多的输出结果，很明显，这些程序的输出还没有真正送到打印机。

当一个计算机有多个用户时，管理和保护存储器、I/O 设备以及其他资源的需求变得强烈起来，因为用户间可能会相互干扰。另外，用户通常不仅共享硬件，还有共享信息。简而言之，操作系统的这种观点认为，操作系统的主要任务是记录哪个程序在使用那些资源，对资源请求进行分配，评估使用代价，并且为不同的程序和用户调解相互冲突的资源请求。

资源管理包括用以下两种不同方式实现多路复用资源：在时间上复用和在空间上复用。当一种资源在时间上复用时，不同的程序或用户轮流使用它。

在图形设计、专业数码摄像以及专业数字视频制作的创意世界里，Macintosh 得到广泛的应用，这些用户对苹果公司及 Macintosh 有着极大的热情。

Chapter 2

进程与线程

从本章开始，我们将深入考察操作系统是如何设计和构造的。操作系统中最核心的概念是进程：这是对正在运行的程序的一个抽象。操作系统的其他所有内容都是围绕着进程的概念展开的，所以，让操作系统的设计者（及学生）尽快并透彻地理解进程是非常重要的。

进程是操作系统提供的最古老的也是最重要的抽象概念之一。既是可以使用的 CPU 只有一个，但它们也具有支持（伪）并发操作的能力，它们将一个单独的 CPU 变换成多个虚拟的 CPU。没有进程的抽象，现代计算机将不复存在。本章会通过大量的细节去探究进程，以及他们的第一个亲戚——线程。

2.1 进程

所有现代的计算机经常会在同一时间做许多件事。习惯于在个人计算机上工作的人么也许不会十分注意这个事实，因此列举一个例子可以更清楚地说明这一问题。