第3章 存储管理

内存（RAM）是计算机中一种需要认真管理的重要资源。就目前来说，虽然一台普通家用计算机的内存容量已经是20世纪60年代早期全球最大的计算机IBM 7094的内存容量的10 000倍以上，但是程序大小的增长速度比内存容量的增长速度要快得多。正如帕金森定律所指出的："不管存储器有多大，程序都可以把它填满"。在这一章中，我们将讨论操作系统是怎样对内存创建抽象模型以及怎样管理内存的。

每个程序员都梦想拥有这样的内存：它是私有的、容量无限大的、速度无限快的，并且是永久性的存储器（即断电时不会丢失数据）。当我们期望这样的内存时，何不进一步要求它价格低廉呢？遗憾的是，目前的技术还不能为我们提供这样的内存。也许你会有解决方案。

除此之外的选择是什么呢？经过多年探索，人们提出了"分层存储器体系"（memory hierarchy）的概念，即在这个体系中，计算机有若干兆（MB）快速、昂贵且易失性的高速缓存（cache），数千兆（GB）速度与价格适中且同样易失性的内存，以及几兆兆（TB）低速、廉价、非易失性的磁盘存储，另外还有诸如DVD和USB等可移动存储装置。操作系统的工作是将这个存储体系抽象为一个有用的模型并管理这个抽象模型。

操作系统中管理分层存储器体系的部分称为*存储管理器*（memory manager）。它的任务是有效地管理内存，即记录哪些内存是正在使用的，哪些内存是空闲的；在进程需要时为其分配内存，在进程使用完后释放内存。

本章我们会研究几个不同的存储管理方案，涵盖非常简单的方案到高度复杂的方案。由于最底层的高速缓存的管理由硬件来完成，本章将集中介绍针对编程人员的内存模型，以及怎样优化管理内存。至于永久性存储器——磁盘——的抽象和管理，则是下一章的主题。我们会从最简单的管理方案开始讨论，并逐步深入到更为缜密的方案。

3.1 无存储器抽象

最简单的存储器抽象就是根本没有抽象。早期大型计算机（20世纪60年代之前）、小型计算机（20世纪70年代之前）和个人计算机（20世纪80年代之前）都没有存储器抽象。每一个程序都直接访问物理内存。当一个程序执行如下指令：

```
MOV REGISTER1, 1000
```

计算机会将位置为1000的物理内存中的内容移到REGISTER1中。因此，那时呈现给编程人员的存储器模型就是简单的物理内存：从0到某个上限的地址集合，每一个地址对应一个可容纳一定数目二进制位的存储单元，通常是8个。

在这种情况下，要想在内存中同时运行两个程序是不可能的。如果第一个程序在2000的位置写入一个新的值，将会擦掉第二个程序存放在相同位置上的所有内容，所以同时运行两个程序是根本行不通的，这两个程序会立刻崩溃。

不过即使存储器模型就是物理内存，还是存在一些可行选项的。图3-1展示了三种变体。在图3-1a中，操作系统位于RAM（随机访问存储器）的底部；在图3-1b中，操作系统位于内存顶端的ROM（只读存储器）中；而在图3-1c中，设备驱动程序位于内存顶端的ROM中，而操作系统的其他部分则位于下面的RAM的底部。第一种方案以前被用在大型机和小型计算机上，现在很少使用了。第二种方案被用在一些掌上电脑和嵌入式系统中。第三种方案用于早期的个人计算机中（例如运行MS-DOS的计算机），在ROM中的系统部分称为BIOS（Basic Input Output System，基本输入输出系统）。第一种方案和第三种方案的缺点是用户程序出现的错误可能摧毁操作系统，引发灾难性后果（比如篡改磁盘）。

当按这种方式组织系统时，通常同一个时刻只能有一个进程在运行。一旦用户键入了一个命令，操