1.5 操作系统概念

多数操作系统都使用某些基本概念和抽象，诸如进程、地址空间以及文件等，它们是需要理解的中心。作为引论，在下面的几节中，我们将较为简要地分析这些基本概念中的一些成分。在本书的后面，我们将详细地讨论它们。为了说明这些概念，我们有时将使用示例，这些示例通常源自UNIX。不过，类似的例子在其他的操作系统中也明显地存在，进而，我们将在第11章中具体讨论Windows Vista。

1.5.1 进程

在所有操作系统中，一个重要的概念是*进程*（process）。进程本质上是正在执行的一个程序。与每个进程相关的是进程的地址空间（address space），这是从某个最小值的存储位置（通常是零）到某个最大值存储位置的列表。在这个地址空间中，进程可以进行读写。该地址空间中存放有可执行程序、程序的数据以及程序的堆栈。与每个进程相关的还有资源集，通常包括寄存器（含有程序计数器和堆栈指针）、打开文件的清单、突出的报警、有关进程清单，以及运行该程序所需要的所有其他信息。进程基本上是容纳运行一个程序所需要所有信息的容器。

进程的概念将在第2章详细讨论，不过，对进程建立一种直观感觉的最便利方式是分析一个分时系统。用户会启动一个视频编辑程序，并指令它按照某个格式转换一小时的视频（有时会花费数小时），然后离开去Web上冲浪。同时，一个被周期性唤醒，用来检查进来的e-mail的后台进程会开始运行。这样，我们就有了（至少）三个活动进程：视频编辑器、Web浏览器以及e-mail接收器。操作系统周期性地挂起一个进程然后启动运行另一个进程。例如，在过去的一秒钟内，第一个进程已使用完分配给它的时间片。

一个进程暂时被这样挂起后，在随后的某个时刻里，该进程再次启动时的状态必须与先前暂停时完全相同，这就意味着在挂起时该进程的所有信息都要保存下来。例如，为了同时读入信息，进程打开了若干文件。同每个被打开文件有关的是指向当前位置的指针（即下一个将读出的字节或记录）。在一个进程暂时被挂起时，所有这些指针都必须保存起来，这样在该进程重新启动之后，所执行的读调用才能读到正确的数据。在许多操作系统中，与一个进程有关的所有信息，除了该进程自身地址空间的内容以外，均存放在操作系统的一张表中，称为*进程表*（process table），进程表是数组（或链表）结构，当前存在的每个进程都要占用其中一项。

所以，一个（挂起的）进程包括：进程的地址空间，往往称作*磁芯映像*（core image，纪念过去年代中使用的磁芯存储器），以及对应的进程表项，其中包括寄存器以及稍后重启动该进程所需要的许多其他信息。

与进程管理有关的最关键的系统调用是那些进行进程创建和进程终止的系统调用。考虑一个典型的例子。有一个称为命令解释器（command interpreter）或shell的进程从终端上读命令。此时，用户刚键入一条命令要求编译一个程序。shell必须先创建一个新进程来执行编译程序。当执行编译的进程结束时，它执行一条系统调用来终止自己。

若一个进程能够创建一个或多个进程（称为子进程），而且这些进程又可以创建子进程，则很容易得到进程树，如图1-13所示。合作完成某些作业的相关进程经常需要彼此通信以便同步它们的行为。这种通信称为*进程间通信*（interprocess communication），将在第2章中详细讨论。

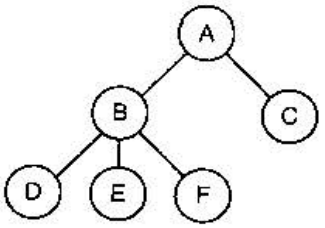

图1-13 一个进程树。进程A创建两个子进程B和C，进程B创建三个子进程D、E和F

其他可用的进程系统调用包括：申请更多的内存（或释放不再需要的内存）、等待一个子进程结束、用另一个程序覆盖该程序等。

有时，需要向一个正在运行的进程传送信息，而该进程并没有等待接收信息。例如，一个进程通过网络向另一台机器上的进程发送消息进行通信。为了保证一条消息或消息的应答不会丢失，发送者要求它所在的操作系统在指定的若干秒后给一个通知，这样如果对方尚未收到确认消息就可以进行重发。在设定该定时器后，程序可以继续做其他工作。