旁注　关于术语的注释

各种异常类型的术语根据系统的不同而有所不同。处理器ISA规范通常会区分异步“中断”和同步“异常”，但是并没有提供描述这些非常相似的概念的概括性的术语。为了避免不断地提到“异常和中断”以及“异常或者中断”，我们用单词“异常”作为通用的术语，而且只有在必要时才区别异步异常(中断)和同步异常(陷阱、故障和终止)。正如我们提到过的，对于每个系统而言，基本的概念都是相同的，但是你应该意识到一些制造厂商的手册会用“异常”仅仅表示同步事件引起的控制流的改变。

8.2　进程

异常是允许操作系统内核提供进程(process)概念的基本构造块，进程是计算机科学中最深刻、最成功的概念之一。

在现代系统上运行一个程序时，我们会得到一个假象，就好像我们的程序是系统中当前运行的唯一的程序一样。我们的程序好像是独占地使用处理器和内存。处理器就好像是无间断地一条接一条地执行我们程序中的指令。最后，我们程序中的代码和数据好像是系统内存中唯一的对象。这些假象都是通过进程的概念提供给我们的。

进程的经典定义就是一个执行中程序的实例。系统中的每个程序都运行在某个进程的上下文(context)中。上下文是由程序正确运行所需的状态组成的。这个状态包括存放在内存中的程序的代码和数据，它的栈、通用目的寄存器的内容、程序计数器、环境变量以及打开文件描述符的集合。

每次用户通过向shell输入一个可执行目标文件的名字，运行程序时，shell就会创建一个新的进程，然后在这个新进程的上下文中运行这个可执行目标文件。应用程序也能够创建新进程，并且在这个新进程的上下文中运行它们自己的代码或其他应用程序。

关于操作系统如何实现进程的细节的讨论超出了本书的范围。反之，我们将关注进程提供给应用程序的关键抽象：

- 一个独立的逻辑控制流，它提供一个假象，好像我们的程序独占地使用处理器。
- 一个私有的地址空间，它提供一个假象，好像我们的程序独占地使用内存系统。

让我们更深入地看看这些抽象。

8.2.1　逻辑控制流

即使在系统中通常有许多其他程序在运行，进程也可以向每个程序提供一种假象，好像它在独占地使用处理器。如果想用调试器单步执行程序，我们会看到一系列的程序计数器(PC)的值，这些值唯一地对应于包含在程序的可执行目标文件中的指令，或是包含在运行时动态链接到程序的共享对象中的指令。这个PC值的序列叫做*逻辑控制流*，或者简称*逻辑流*。

考虑一个运行着三个进程的系统，如图8-12所示。处理器的一个物理控制流被分成了三个逻辑流，每个进程一个。每个竖直的条表示一个进程的逻辑流的一部分。在这个例子中，三个逻辑流的

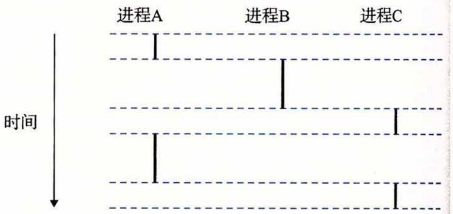

图8-12　逻辑控制流。进程为每个程序提供了一种假象，好像程序在独占地使用处理器。每个竖直的条表示一个进程的逻辑控制流的一部分