第 8 章

C H A P T E R 8

异常控制流

从给处理器加电开始，直到你断电为止，程序计数器假设一个值的序列

$$a_0, a_1, \cdots, a_{n-1}$$

其中，每个 a_k 是某个相应的指令 I_k 的地址。每次从 a_k 到 a_{k+1} 的过渡称为控制转移(control transfer)。这样的控制转移序列叫做处理器的控制流(flow of control 或 control flow)。

最简单的一种控制流是一个“平滑的”序列，其中每个 I_k 和 I_{k+1} 在内存中都是相邻的。这种平滑流的突变(也就是 I_{k+1} 与 I_k 不相邻)通常是由诸如跳转、调用和返回这样一些熟悉的程序指令造成的。这样一些指令都是必要的机制，使得程序能够对由程序变量表示的内部程序状态中的变化做出反应。

但是系统也必须能够对系统状态的变化做出反应，这些系统状态不是被内部程序变量捕获的，而且也不一定要和程序的执行相关。比如，一个硬件定时器定期产生信号，这个事件必须得到处理。包到达网络适配器后，必须存放在内存中。程序向磁盘请求数据，然后休眠，直到被通知说数据已就绪。当子进程终止时，创造这些子进程的父进程必须得到通知。

现代系统通过使控制流发生突变来对这些情况做出反应。一般而言，我们把这些突变称为异常控制流(Exceptional Control Flow，ECF)。异常控制流发生在计算机系统的各个层次。比如，在硬件层，硬件检测到的事件会触发控制突然转移到异常处理程序。在操作系统层，内核通过上下文切换将控制从一个用户进程转移到另一个用户进程。在应用层，一个进程可以发送信号到另一个进程，而接收者会将控制突然转移到它的一个信号处理程序。一个程序可以通过回避通常的栈规则，并执行到其他函数中任意位置的非本地跳转来对错误做出反应。

作为程序员，理解 ECF 很重要，这有很多原因：

- 理解 ECF 将帮助你理解重要的系统概念。ECF 是操作系统用来实现 I/O、进程和虚拟内存的基本机制。在能够真正理解这些重要概念之前，你必须理解 ECF。
- 理解 ECF 将帮助你理解应用程序是如何与操作系统交互的。应用程序通过使用一个叫做陷阱(trap)或者系统调用(system call)的 ECF 形式，向操作系统请求服务。比如，向磁盘写数据、从网络读取数据、创建一个新进程，以及终止当前进程，都是通过应用程序调用系统调用来实现的。理解基本的系统调用机制将帮助你理解这些服务是如何提供给应用的。
- 理解 ECF 将帮助你编写有趣的新应用程序。操作系统为应用程序提供了强大的 ECF 机制，用来创建新进程、等待进程终止、通知其他进程系统中的异常事件，以及检测和响应这些事件。如果理解了这些 ECF 机制，那么你就能用它们来编写诸如 Unix shell 和 Web 服务器之类的有趣程序了。
- 理解 ECF 将帮助你理解并发。ECF 是计算机系统中实现并发的基本机制。在运行中的并发的例子有：中断应用程序执行的异常处理程序，在时间上重叠执行的进程和线程，以及中断应用程序执行的信号处理程序。理解 ECF 是理解并发的第一步。我们会在第 12 章中更详细地研究并发。