8.7　操作进程的工具

Linux 系统提供了大量的监控和操作进程的有用工具。

STRACE：打印一个正在运行的程序和它的子进程调用的每个系统调用的轨迹。对于好奇的学生而言，这是一个令人着迷的工具。用-static 编译你的程序，能得到一个更干净的、不带有大量与共享库相关的输出的轨迹。

PS：列出当前系统中的进程(包括僵死进程)。

TOP：打印出关于当前进程资源使用的信息。

PMAP：显示进程的内存映射。

/proc：一个虚拟文件系统，以 ASCII 文本格式输出大量内核数据结构的内容，用户程序可以读取这些内容。比如，输入“cat/proc/loadavg”，可以看到你的 Linux 系统上当前的平均负载。

8.8　小结

异常控制流(ECF)发生在计算机系统的各个层次，是计算机系统中提供并发的基本机制。

在硬件层，异常是由处理器中的事件触发的控制流中的突变。控制流传递给一个软件处理程序，该处理程序进行一些处理，然后返回控制给被中断的控制流。

有四种不同类型的异常：中断、故障、终止和陷阱。当一个外部 I/O 设备(例如定时器芯片或者磁盘控制器)设置了处理器芯片上的中断管脚时，(对于任意指令)中断会异步地发生。控制返回到故障指令后面的那条指令。一条指令的执行可能导致故障和终止同步发生。故障处理程序会重新启动故障指令，而终止处理程序从不将控制返回给被中断的流。最后，陷阱就像是用来实现向应用提供到操作系统代码的受控的入口点的系统调用的函数调用。

在操作系统层，内核用 ECF 提供进程的基本概念。进程提供给应用两个重要的抽象：1)逻辑控制流，它提供给每个程序一个假象，好像它是在独占地使用处理器，2)私有地址空间，它提供给每个程序一个假象，好像它是在独占地使用主存。

在操作系统和应用程序之间的接口处，应用程序可以创建子进程，等待它们的子进程停止或者终止，运行新的程序，以及捕获来自其他进程的信号。信号处理的语义是微妙的，并且随系统不同而不同。然而，在与 Posix 兼容的系统上存在着一些机制，允许程序清楚地指定期望的信号处理语义。

最后，在应用层，C 程序可以使用非本地跳转来规避正常的调用/返回栈规则，并且直接从一个函数分支到另一个函数。

参考文献说明

Kerrisk 是 Linux 环境编程的完全参考手册[62]。Intel ISA 规范包含对 Intel 处理器上的异常和中断的详细讨论[50]。操作系统教科书[102，106，113]包括关于异常、进程和信号的其他信息。W. Richard Stevens 的[111]是一本有价值的和可读性很高的经典著作，是关于如何在应用程序中处理进程和信号的。Bovet 和 Cesati[11]给出了一个关于 Linux 内核的非常清晰的描述，包括进程和信号实现的细节。

家庭作业

* 8.9　考虑四个具有如下开始和结束时间的进程：

进程	开始时间	结束时间
A	5	7
B	2	4
C	3	6
D	1	8