第二部分　陷入、中断、系统调用和进程管理

第二部分与陷入、硬件中断和软件中断有关。

陷入和硬件中断使CPU的正常指令执行序列产生突然的切换。这提供了一种处理特殊条件的机制，这种特殊条件发生在CPU立即控制之外。

被称之为“系统调用”(system call)的机制使用了这种设施，由此，一道用户程序可以执行一条“trap”指令以故障地造成一次中断，借此获得操作系统的注意和帮助。

软件中断(software interrupt)或“信号”(signal)是进程之间通信的一种机制，特别是当有“坏新闻”时，更是如此。

第9章　硬件中断和陷入

与其他许多计算机一样，PDP11计算机有一个“中断”(interrupt)机制，它允许外部设备(它们是CPU以外的设备)控制器在适当时间中断CPU，以此要求操作系统的服务。

同样的机制也被应用于“陷入”(trap)，陷入是发生在CPU内部的事件，它们与硬件和软件错误有关，用户程序也使用陷入机制向系统提出服务请求。

9.1　硬件中断

中断的作用是使CPU从当前所执行的指令序列转移到另一道程序。

在硬件中断期间：

- CPU保存当前处理机状态字(PS)和当前程序计数(PC)到它的内部寄存器中。
- 从位于内存低地址区的两个连续字中重装PC和PS。这两个字中第1个字的地址被称之为该中断的“矢量单元”(vector location)。
- 最后将原先的PC和PS值保存至新的当前线。(取决于PS的新值，该栈可以是核心栈或用户栈。)

不同的外设通常具有不同的矢量单元。特定设备的实际矢量单元由硬接线决定，很难加以更改。还有对各种外设选择矢量单元也有几乎固定不变的惯例。

在中断发生后，由于重装了PC，所以CPU执行的指令源(指令序列)发生了变化。新的指令源应当是与造成中断的外设控制器相关的过程。因为PS已更改，所以处理机的运行状态也可能发生改变。在UNIX中，中断前的处理机状态可以是“用户”或“核心”态，但中断后处理机状态则一定是“核心态”。请回忆一下处理机状态的改变意味着：

1) 存储映照的更改(注意，为避免混淆，矢量单元总是被解释为核心态地址)。

2) 栈指针的改变(请回忆一下栈指针，SP或r6，是唯一具有双份的特殊寄存器，每种状态1个。这意味着在处理机状态更改后，栈指针值虽然没有重新装入，也被改变。

9.2 中断矢量

在我们的样本操作系统中，所选择的代表性外设列于表 9-1中，其中也包括了硬件定义的矢量单元和优先级。

表9-1 中断矢量单元和优先级

矢量单元	外部设备	中断优先级	处理机优先权
060	电传打字机输入	4	4
064	电传打字机输出	4	4
070	纸带机输入	4	4
074	纸带机输出	4	4
100	线频时钟	6	6
104	可编程时钟	6	6
200	行式打印机	4	4
220	RK磁盘驱动	5	5

9.3 中断处理程序

在我们的样本UNIX源代码中，有7个称之为“中断处理程序”的过程，它们是由中断引发而执行的：

```
pcrint      (8719)
pcpint      (8739)
lpint       (8976)
```

另外3个则在对相关设备的代码部分进行分析时一起讨论。

非当 CPU准备好接受它，否则不会立即发生。人们常常
允许中断具有较高优先级的活动。
几的优先级，它分 8级，分别标记为0-7级。每一个中断也
先级。只要处理机优先级大于或等于中断优先级，那么该

将由存放在中断矢量单元的 PS决定。这不一定与中断优
而操作系统则能够在任何时候更改矢量单元的内容。
实际上只支持4、5、6和7中断优先级，亦即中断优先级1、

9.5 中断优先级

在UNIX中，中断处理例程被初启时其优先级与相应中断的优先级相同。这意味着在处理中断期间， CPU不会接受来自同一优先级设备的第 2个中断。在处理机优先级降低后，这种对第 2个中断的延迟处理才可能结束。而降低处理机优先级的方式通常有两种：一种是执行为此目的专门编写的“ spl ”过程中的一种；另一种是从中断返回时重装处理机状态字。

在中断处理期间，可暂时提高处理机优先级以保证某些操作的整体性。例如，面向字符的设备(纸带输入机/穿孔器、行式打印机……)的中断处于 4级。它们的中断处理程序调用“ getc ”(0930)或“ putc ”(0967)，在对字符缓存队列进行操作时，它们将处理机优先级临时提升为5级。

控制台电传打字机的中断处理程序使用超时 (timeout)设施。这涉及一种时钟中断处理程序也对其进行操作的队列，时钟中断处理程序的运行优先级为 6级。为了防止时钟中断可能产生的干扰，“ timeout ”过程(3835)在7级运行(最高级别)。

一般而言，使中断处理程序在低于中断优先级的处理机优先级下运行是不符合常理的。采取这种处理的风险在于：在完成对第 1个中断的处理之前，可能产生同样类型、甚至来自同一设备的中断。在最好情况下这会引起一些不便，而在最坏的情况下则会产生灾难性的后果。但是，时钟中断处理程序执行得非常频繁，每秒钟至少有一次要做很多额外的工作，所以确实要采用这种不一般的方法。

9.6 中断处理程序的规则

正如上面所讨论的，中断处理程序对处理机优先级应当特别周密地进行考虑，以防其他中断发生得太快。与此同时，也要考虑到不使其他中断延迟过度，从而造成整个系统性能的下降。

应当着重提请注意的是：当一次中断发生时，当前活动进程多半并不是对此中断有兴趣的进程。考虑下列情景：

用户进程#m是活动进程并启动i/o操作。它执行一条trap指令，然后转移至核心态。核心态进程#m启动所要求的操作，然后调用“ sleep ”挂起它自身，以等待该操作的完成……

经若干时间后，当某个其他进程，例如用户进程 #n成为活动进程时，该I/O操作完成，发生一次中断。进程#n转入核心态，核心态进程#n处理该中断。即使它对该I/O操作并无任何关系也没有任何了解。

通常，核心态进程#n作为其活动的一部分会唤醒进程#m。当然也有一些例外，例如出错，然后重新执行有关I/O操作。

很清楚，因为当前的“ u ”结构不大可能是与本次中断有关的“ u ”结构，所以外设的中断处理程序不应当引用当前的“ u ”结构。(相关的u结构如果暂时已换出至磁盘上，那么它就是不可存取的。)

同样，中断处理程序不应当调用“ sleep ”，否则被挂起的进程非常可能是一个毫无关连的

无辜进程。

9.7 陷入

“陷入”与“中断”有些相似，它们都是由相同的硬件机构处理的事件，随之而来的是处理它们的软件机构也类似。

“陷入”与“中断”又不完全相同，它是由 CPU内部事件而非外部事件而引发的。 (在其他系统中，使用术语“内部中断” (internal interrupt)和“外部中断” (external interrupt)来突出两者之间的区别。)由于硬件和电源故障，陷入可能突然发生，若陷入是由执行一条非法指令或“ trap ”指令而引发的，那么这种陷入是可以预测和可重复产生的。

CPU会立即识别出“陷入”。“陷入”不会像低优先级中断那样选择被延迟处理。如果你愿意这样想，那么可以把“陷入”视为第 8优先级中断，也就是最高优先级中断。

在用户态程序中可以按需任意插入“ trap ”指令以向操作系统提出特定的服务请求，并立即得到操作系统的注意。这种机制是“系统调用”设施的重要组成部分。

与中断类似，陷入造成从矢量单元重新装入 PC和PS，并将PC、PS值保存至当前栈。表9-2列出了多种“陷入”类型的矢量单元。

表9-2 陷入矢量单元和优先级

矢量单元	陷入类型	处理机优先级
044	总线超时	7
010	非法指令	7
014	bpt-跟踪	7
020	iot	7
024	电源故障	7
030	仿真陷入指令	7
034	trap指令	7
114	11/70奇偶错	7
240	程序中断	7
244	浮点错	7
250	段违例	7

应将表9-1的内容与“ low.s ”文件进行比较。正如前面已指出的，该文件与 conf.c都是由公用程序“ mkconf ”按每个系统的实际外设配置而生成的。

9.8 汇编语言trap

从“ low.s ”中观察到：“陷入”和“中断”是由软件分开进行处理的。可是再仔细检查可以发现：“ call ”和“ trap ”是进入“ m40.s ”中同一段代码的不同入口点 (见0755和0776行)。下一章将对此段代码详细地进行分析。

在执行此段代码中，调用一“ C ”语言过程以执行进一步的特定处理。在某个中断情况下，

该“C”过程是专用于该特定设备控制器的中断处理程序。

在某个陷入情况下，该“C”过程是称之为“trap”的另一个过程。(是的，trap这个字确实是用得过多了!)在系统出错情况下，“trap”过程很可能将调用“panic”，而在“系统调用”情况下将经由“trap1”(2841)间接地调用相对应的系统调用过程。

9.9 返回

在完成了中断和陷入处理后，公共路径代码以“rtt”指令结束。该指令从当前栈(核心态栈)中取出PC和PS并重新装入相应寄存器，这样就恢复了中断和陷入前的处理机环境。