中断和异常

# 一、什么是中断？

中断通常被定义为一个事件，该事件改变处理器执行的指令顺序。这样的事件与CPU芯片内外部硬件电路产生的电信号相对应。

中断通常分为同步中断和异步中断：

* 同步中断是当指令执行时由CPU控制单元产生的，之所以称为同步，是因为只有在一条指令终止执行后CPU才会发出中断。
* 异步中断是由其他硬件设备依照CPU时钟信号随机产生的。

在Intel微处理器手册中：

* 把同步中断称为***异常（exception）***
* 把异步中断称为***中断（interrupt）***

这两类中断的共同特点是什么？如果CPU当前不处于核心态，则发起从用户态到核心态的切换。接下来，在内核中执行一个专门的例程，称为中断服务例程（interrupt service routine）。或中断处理程序（interrupt handler）

另一方面，异常是由程序的错误产生的，或者是由内核必须处理的异常条件产生的。第一种情况下，内核通过发送一个每个Unix/Linux程序员都熟悉的信号来处理异常。第二种情况下，内核执行恢复异常需要的所有步骤，例如缺页异常等。

# 二、中断信号的作用

中断信号提供了一种特殊的方式，使处理器转而去运行正常控制流之外的代码。当一个中断信号达到时，CPU必须停止它当前正在做的事情，并且切换到一个新的活动。为了这做到这一点，就要在内核态堆栈保存程序计数器的当前值（即EIP和CS寄存器的内容），并把与中断类型相关的一个地址放进程序计数器。

这可能会让我们想起系统调度的进程切换，发生在内核用一个进程替换另一个进程时。但是中断处理与进程切换有一个明显的差异：*由中断或异常处理程序执行的代码不是一个进程。*更准确的说，它是一个内核控制路径，代表中断发生时正在运行的进程执行。作为一个内核控制路径，中断处理程序比一个进程要“轻”（中断的上下文很少，建立或终止中断处理需要的时间也很少）

中断处理是由内核执行的最敏感的任务之一，因为它必须满足下列约束：

* 当内核正打算去完成一些别的事情时，中断随时会到来。因此，内核的目标就是让中断尽可能快地处理完，尽其所能把更多的处理向后推迟。因此，内核响应中断后需要进行的操作分为两部分：关键而紧急的部分，内核立即执行；其余推迟的部分，内核随后执行。
* 因为中断随时会到来，所以内核可能正在处理其中的一个中断时，另一个不同类型的中断又发生了。内核应该尽可能地允许这种情况发生，因为这能维持更多的I/O设备得到处理的机会。因此，中断处理程序必须编写成使相应的内核控制路径能以嵌套的方式执行。当最后一个内核控制路径终止时，内核必须能恢复被中断进程的执行，或者，如果中断信号已导致了重新调度，内核也应能切换到另外的进程。
* 尽管内核在处理前一个中断时可以接受一个新的中断，但在内核代码中还是存在一些临界区，在临界区中，中断必须被禁止。必须尽可能地限制这样的临界区，因为根据以前的要求，内核，尤其是中断处理程序，应该在大部分时间内以开中断的方式运行。

# 三、IRQ和中断

中断这个名词使用得并不是很谨慎，为什么？由于中断是用来表示由CPU和外部硬件发出的信号所产生的。但是中断不能由处理器外部的外设直接产生，而必须借助于一个称为可编程中断控制器（programmable interrupt controller）的标准组件来请求，该组件存在于每个系统中。

外部设备，会有电路连接到用于向中断控制器发送中断请求的组件。控制器在执行了各种电工任务之后，将中断请求转发到CPU的中断输入中。因为外部设备不能直接发出中断，而必须通过中断控制器的标准组件来请求中断，所以这种请求更正确的叫法是IRQ，或中断请求（Interrupt Request）。

每个能够发出中断请求的硬件设备控制器都有这么一条名为IRQ的输出线。所有现有的IRQ线都会与这个中断控制器（PIC）的硬件电路的输入引脚相连。下面来看看这种中断控制器执行下列动作：

* 1. 监视IRQ线，检查产生的信号 。如果有一条或两条以上的IRQ线上产生信号，就选择引脚编号较小的IRQ线。
  2. 如果一个引发信号出现在IRQ线上：

1. 把接收到的引发信号转换成对应的向量（索引）。
2. 把这个向量存放在中断控器的一个I/O端口，从而允许CPU通过数据总线读取此向量。
3. 把引发信号发送到处理器的INTR引脚，即产生一个中断。
4. 等待，直到CPU通过把这个中断信号写进可编程中断控制器的一个I/O端口来确认它；当这种情况发生时，清INTR线。
   1. 返回到第一步。

IRQ线是从0开始顺序编号的，因此，第一条IRQ线通常表示成IRQ0。与IRQn关联的Intel缺省向量是n+32。如前所述，通过向中断控制器端口发布合适的指令，就可以修改IRQ和向量之间的映射。

可以有选择地禁止每条IRQ线。因此，可以对PIC编程从而禁止IRQ，也就是说，可以告诉PIC停止对给定的IRQ线发布中断，或者激活它们，***禁止的中断是丢失不了的，它们一旦激活，PIC就又把它们发送到CPU。***这个特点被大多数中断处理程序使用，因为这允许中断处理程序逐次地处理同一类型的IRQ。

## （一）处理中断

在CPU得知发生中断后，它将进一步的处理委托给一个软件例程，该例程可能会修复故障、提供专门的处理或将外部事件通知用户进程。由于每个中断和异常都有唯一的编号，内核使用一个数组项是指向处理程序函数的指针。相关的中断号根据数组项在数组中位置判断。如下图所示：



### 1、进入和退出任务

如下图，中断处理划分为3部分。首先，必须建立一个适当的环境，使得处理程序函数能够在其中执行，接下来调用处理程序自身，最后将系统复原到中断之前的状态。调用中断处理程序前后的两部分，分别称为进入路径和退出路径。



进入和退出任务还负责确保处理器从用户态切换到核心态。进入路径的一个关键任务是，从用户态栈切换到核心态栈。但是这一点还不够。因为内核还要使用CPU资源执行其代码，进入路径必须保存用户应用程序当前的寄存器，以便 在中断活动结束后恢复。这与进程调度间用上下文切换的机制是相同的。在进入核心态时，只保存整个寄存器集合的一部分。内核并不使用全部寄存器。（如内核代码中不使用浮点操作，因而不保存浮点寄存器）。平台相关的数据结构pt\_regs列出了核心态可能修改的所有寄存器，它的定义考虑到了不同的CPU之间的差别。

在退出路径中，内核会检查下列事项。

* + 调度器是否应该选择一个新进程代替旧的进程。
  + 是否有信号必须投递到原进程

从中断返回之后，只有确认了这两个问题，内核才能完成其常规任务，即还原寄存器集合、切换到用户态栈、切换到适用于用户应用程序的适当的处理器状态，或切换到一个不同的保护环。

术语中断处理程序的使用可能引起岐义。因为它是用于指代CPU对ISR（中断服务程序）的调用，包括了进入/退出路径和ISR本身。当然，如果只指代在进入路径和退出路径之间进行由C语言实现的例程，将更为准确