# 中断

## 背景

如何让处理器和外设协同工作，且不会降低机器的整体性能呢？轮询（polling）可能会是一种解决方法。它可以让内核定期对设备的状态进行查询，然后做出相应的处理。不过这种方法很可能会让内核做不少无用功，因为无论硬件设备是正在忙碌着完成任务还是已经大功告成，轮询总会周期性地重复执行。更好的方法是由我们来提供一种机制，让硬件在需要的时候再向内核发出信号。这就是中断机制。

## 中断

中断使得硬件得以发出通知给处理器。例如，在你敲击键盘的时候，键盘控制器（控制键盘的硬件设备）会发送一个中断，通知操作系统有键按下。中断本质上是一种特殊的电信号，由硬件设备发向处理器。处理器接收到中断后，会马上向操作性同反映此信号的到来，然后就由操作系统负责处理这些新到来的数据。硬件设备生成中断的时候并不考虑与处理器的时钟同步——换句话说就是中断随时可以产生。因此，内核随机可能会因为新到来的中断而被打断。

不同的设备对应的中断不同，而每个中断都通过一个唯一的数字标志。因此，来自键盘的中断就有别于来自硬盘的中断，从而使得操作系统能够对中断进行区分，并知道哪个硬件设备产生了哪个中断。这样，操作系统才能给不同的中断提供对应的中断处理程序。

这些中断值通常被称为中断请求（IRQ）线。每个IRQ线都会被关联一个数值亮——例如，在经典的PC机上，IRQ0是时钟中断，而IRQ1是键盘中断。

## 异常

异常与中断不同，它在产生时必须考虑与处理器的时钟同步。实际上，异常也常常称为同步中断。在处理器执行到由于编程失误而导致的错误指令（如被0除）的时候，或者是在执行期间出现特殊情况（如缺页），必须靠内核来处理的时候，处理器就会产生一个异常。因为许多处理器体系结构处理异常与处理中断的方法类似，因此，内核对他们的处理也很类似。

# 中断处理程序

## 概述

## 注册中断处理程序

## 编写中断处理程序

# 中断上下文

又想中断处理程序运行得快，又想中断处理程序完成的工作量多，这两个目的显然有所抵触。鉴于两个目的之间存在此消彼长的矛盾关系，所以我们一般把中断处理切为两个部分或两半。中短程狐狸程序是上半部（top half）——接收到一个终端，它就立马开始执行，但只做有严格时限的工作，例如对接收的中断进行应答或复位硬件。这些工作都是在所有中断被禁止的情况下完成的。能够被允许稍后的完成的工作会推迟到下半部（botoom halkf）去。此后，在合适的时机，下半部分会被中断执行。

# 中断处理机制实现

# /proc/interrupts

# 中断控制

## 禁止和激活中断

## 禁止指定中断

## 中断系统的状态