

深入浅出Linux设备驱动中断处理介绍 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/221/2021_2022__E6_B7_B1_E5_85_A5_E6_B5_85_E5_c103_221622.htm 与Linux设备驱动中

中断处理相关的首先是申请与释放IRQ的API：request_irq()

和free_irq()。request_irq()的原型为：int request_irq(unsigned int irq,void (*handler)(int irq, void *dev_id, struct pt_regs

*regs),unsigned long irqflags,const char * devname, void

*dev_id).irq是要申请的硬件中断号；handler是向系统登记的中断处理函数，是一个回调函数，中断发生时，系统调用这个函数，dev_id参数将被传递；irqflags是中断处理的属性，若

设置SA_INTERRUPT，标明中断处理程序是快速处理程序，快速处理程序被调用时屏蔽所有中断，慢速处理程序不屏蔽

；若设置SA_SHIRQ，则多个设备共享中断，dev_id在中断共享时会用到，一般设置为这个设备的device结构本身或

者NULL。free_irq()的原型为：void free_irq(unsigned int irq,void *dev_id). 另外，与Linux中断息息相关的一个重要概念

是Linux中断分为两个半部：上半部（tophalf）和下半部(bottom half)。上半部的功能是"登记中断"，当一个中断发生时，它进行相应地硬件读写后就把中断例程的下半部挂到

该设备的下半部执行队列中去。因此，上半部执行的速度就会很快，可以服务更多的中断请求。但是，仅有"登记中断"

是远远不够的，因为中断的事件可能很复杂。因此，Linux引入了一个下半部，来完成中断事件的绝大多数使命。下半部和上半部最大的不同是下半部是可中断的，而上半部是不可

中断的，下半部几乎做了中断处理程序所有的事情，而且可

中断的，下半部几乎做了中断处理程序所有的事情，而且可

以被新的中断打断！下半部则相对来说并不是非常紧急的，通常还是比较耗时的，因此由系统自行安排运行时机，不在中断服务上下文中执行。Linux实现下半部的机制主要有tasklet和工作队列。tasklet基于Linux softirq，其使用相当简单，我们只需要定义tasklet及其处理函数并将二者关联：
void my_tasklet_func(unsigned long). //定义一个处理函数：
DECLARE_TASKLET(my_tasklet, my_tasklet_func, data). //定义一个tasklet结构my_tasklet，与my_tasklet_func(data)函数相关联
然后，在需要调度tasklet的时候引用一个简单的API就能使系统在适当的时候进行调度运行：
tasklet_schedule(&.global_var, buf, sizeof(int))) { return -EFAULT. } //调度tasklet执行 tasklet_schedule(&.test_tasklet).
return sizeof(int). } 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com