内容：内核管理、进程管理、文件系统、操作系统调用、网络协议栈、虚拟化等等。

**一、异常及非屏蔽中断**

异常就是CPU内核出现的中断，在CPU执行特定指令时出现非法情况。非屏蔽中断就是计算机内部硬件出错时引起的异常情况。

向量 异常名称 类型 描述

1 调度 故障/陷阱 当对一个程序进行逐步调度时

6 非法操作码 故障 当CPU检查到一个无效的操作码

4 溢出 陷阱 当into指令被执行

3 断点 陷阱 当int3(断点指令）指令引起

12 栈段异常 故障 超过栈段界限

14 页异常 故障 寻址的页不在内存，或违反一种分布保护机制

Intel把非屏蔽中断作为异常的一种来处理。IntelX86处理器发布大约20种异常。每种异常都会由专门的异常处理程序来处理操作，它们通常把一个UNIX信号发送到引起异常的进程。具体异常处理程序发送的信号如下：

向量 异常名称 出错码 异常处理程序 信号

2 非屏蔽中断 NO nmi() None

4 溢出 NO overflow() SIGSEGV

3 断点 NO int3() SIGTRAP

6 非法操作码 NO invalid\_op() SIGILL

12 栈段异常 YES stack\_segment() SIGBUS

14 页异常 YES page\_fault() SIGSEGV

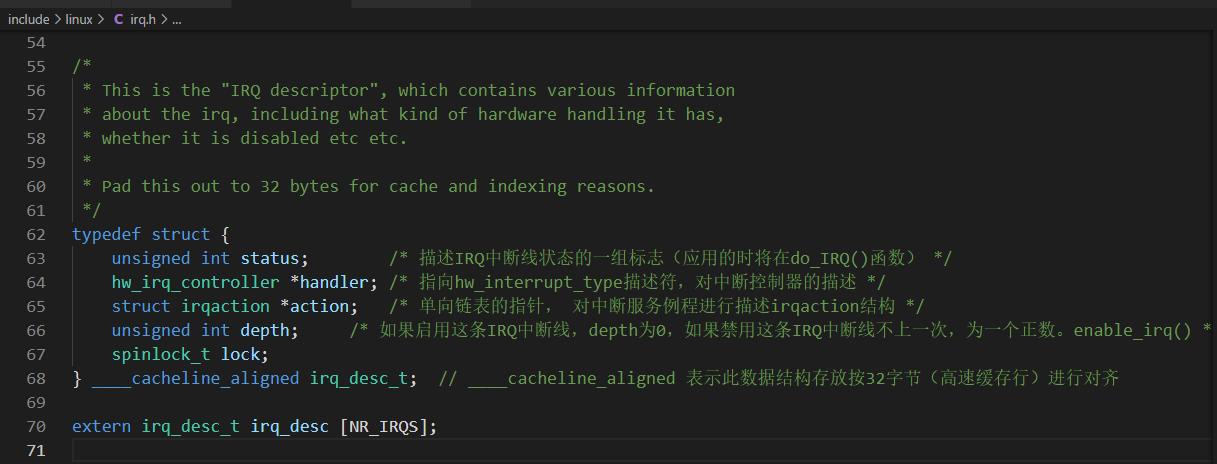
**二、中断请求初始化分析**

由于计算机硬件的限制，很多外部设备不共享中断线，例如，PC配置可以把同一条中断线分配给网卡或图形卡。在Linux系统设计中，专门为每个中断请求IRQ设置一个队列，称为中断请求队列。

中断线、中断请求(IRQ)号及中断向量它们之间关系：中断线是中断请求的一种物理描述，逻辑上对应一个中断请求号（中断号），第n个中断号(IRQn)的缺少中向量是n+32。

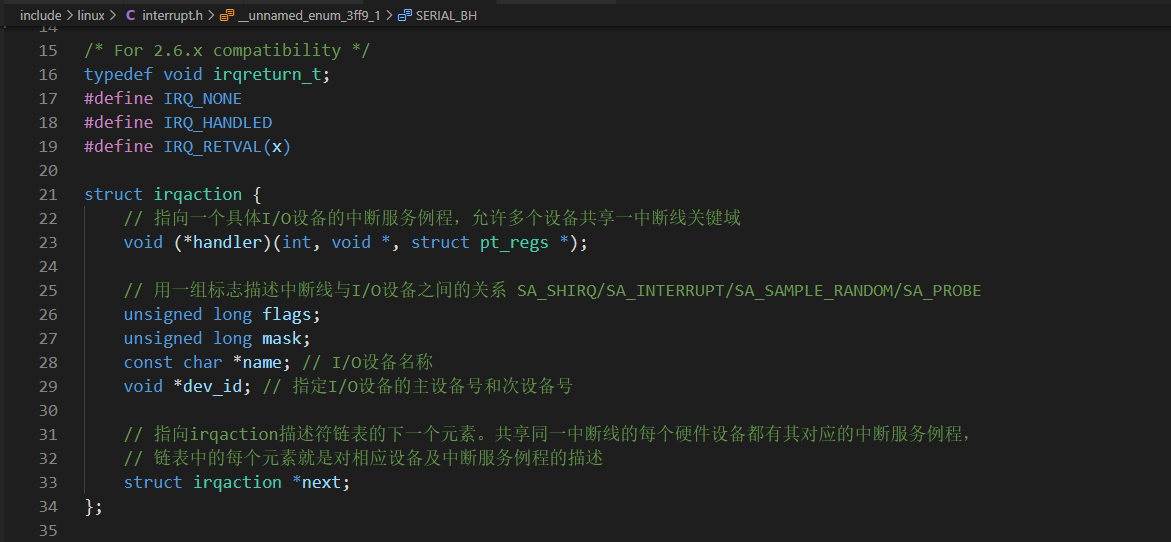
**三、 IRQ队列数据结构**

在256个中断向量中，除32个分配给异常以外，还有224个作为中断向量。对于第个IRQ，Linux都用一个irq\_desc\_t数据结构来描述，我们就把它叫IRQ描述符，224个IRQ形成一个数组irq\_desc[]，具体在内核源码分析如下：



1、中断服务例程描述符irqaction

在IRQ描述符中我们看到指针action的结构为irqaction，它是为多个设备能共享一条中断线而设置的一个数据结构。具体内核源码分析如下：



2、中断服务例程

具体来讲中断处理程序相当于某个中断向量的总处理程序，每个中断服务例程都有相同的参数：

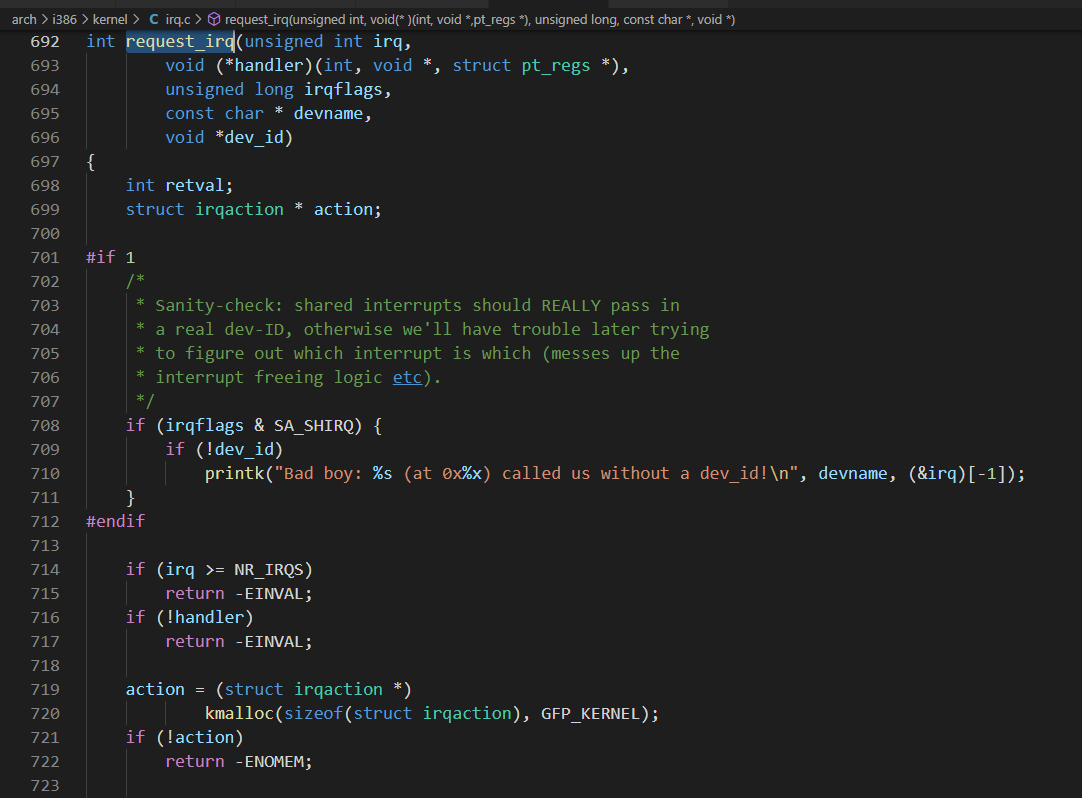
IRQ：中断号

dev\_id：设备标识符，其类型为void\*

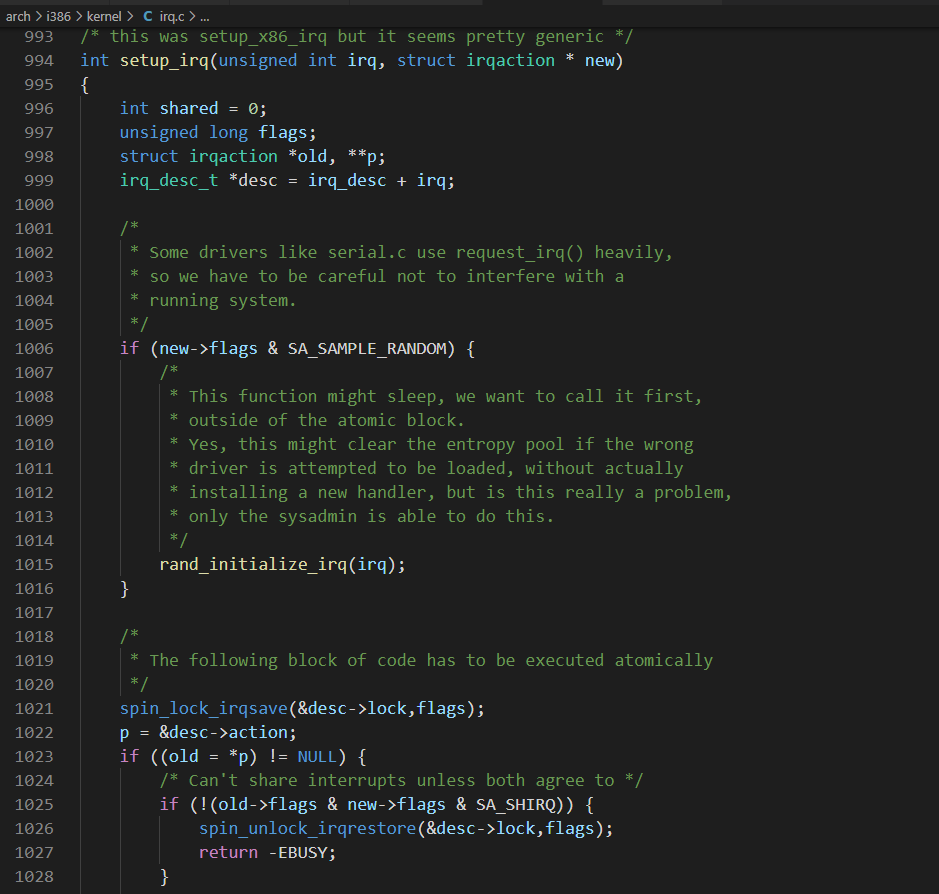
regs：指向内核堆栈区的指针。

3、中断请求队列的初始化

在IDT表初始化完成之初，每个中断服务队列还为空。即使打开中断且某个外设中断真的发生，也得不到实际服务，因为CPU虽然通过中断门进入某个中断向量的总处理程序，具体中断服务例程还没有挂入中断请求队列，因此在设备驱动程序的初始化阶段，必须通过request\_irq()函数将对应的中断服务例程挂入中断请求队列。具体内核源码如下：



真正对中断请求队列进行初始化的函数如下：



学习内核源码分析：具备一定开发能力、操作系统原理、Linux驱动与应用开发。