Linux设备驱动程序:与硬件通信Linux认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/645/2021_2022_Linux_E8_AE_BE_E5_A4_c103_645024.htm I/O 端口和 I/O 内存 每种外设都是通过读写寄存器来进行控制。 在硬件层，内存区和 I/O 区域没有概念上的区别: 它们都是通过向在地址总线和控制总线发出电平信号来进行访问，再通过数据总线读写数据。 因为外设要与I\O总线匹配，而大部分流行的 I/O 总线是基于个人计算机模型（主要是 x86 家族：它为读和写 I/O 端口提供了独立的线路和特殊的 CPU 指令），所以即便那些没有单独I/O端口地址空间的处理器，在访问外设时也要模拟成读写I\O端口。这一功能通常由外围芯片组（PC 中的南北桥）或 CPU 中的附加电路实现（嵌入式中的方法）。 Linux 在所有的计算机平台上实现了 I/O 端口。但不是所有的设备都将寄存器映射到 I/O 端口。虽然ISA设备普遍使用 I/O 端口，但大部分 PCI 设备则把寄存器映射到某个内存地址区，这种 I/O 内存方法通常是首选的。因为它无需使用特殊的处理器指令，CPU 核访问内存更有效率，且编译器在访问内存时在寄存器分配和寻址模式的选择上有更多自由。 I/O 寄存器和常规内存 在进入这部分学习的时候，首先要理解一个概念：side effect，书中译为边际效应，第二版译为副作用。我觉得不管它是怎么被翻译的，都不可能精准表达原作者的意思，所以我个人认为记住side effect就好。下面来讲讲side effect的含义。我先贴出两个网上已有的两种说法（在这里谢谢两位高人的分享）：第一种说法： 3. side effect（译为边际效应或副作用）：是指读取某个地址时可能导致该地址内容发生变化，比如，有

些设备的中断状态寄存器只要一读取，便自动清零。I/O寄存器的操作具有side effect，因此，不能对其操作不能使用cpu缓存。 原文网址： http://qinbh.blog.sohu.com/62733495.html 第二种说法： 说一下我的理解：I/O端口与实际外部设备相关联，通过访问I/O端口控制外部设备，“ 边际效应 ”是指控制设备（读取或写入）生效，访问I/O口的 主要目的就是边际效应，不像访问普通的内存，只是在一个位置存储或读取一个数值，没有别的含义了。我是基于arm平台理解的，在《linux设备驱动程 序》第二版中的说法是“ 副作用 ”，不是“ 边际效应 ”。 原文网址： linux.chinaunix.net/bbs/viewthread.php?tid=890636gt.http://linux.chinaunix.net/bbs/viewthread.php?tid=890636gt. void barrier(void) /*告知编译器插入一个内存屏障但是对硬件没有影响。编译后的代码会将当前CPU 寄存器中所有修改过的数值保存到内存中, 并当需要时重新读取它们。可阻止在屏障前后的编译器优化，但硬件能完成自己的重新排序。其实linux/kernel.hgt. # define barrier() __memory_barrier() #include asm/system.hgt.registers.addr, io_destination_address). writel(dev-gt.registers.operation, DEV_READ). wmb()./*类似一条分界线，上面的写操作必然会在下面的写操作前完成，但是上面的三个写操作的排序无法保证*/ writel(dev-gt. struct resource *request_region(unsigned long first, unsigned long n, const char *name)./*告诉内核：要使用从 first 开始的 n 个端口,name 参数为设备名。若分配成功返回非 NULL，否则将无法使用需要的端口。*/ /*所有的的端口分配显示在 /proc/ioports 中。若不能分配到需要的端口，则可以到这里看看谁先用了。*/

/*当用完 I/O 端口集(可能在模块卸载时), 应当将它们返回给系统*/ void release_region(unsigned long start, unsigned long n). int check_region(unsigned long first, unsigned long n). /*检查一个给定的 I/O 端口集是否可用,若不可用, 返回值是一个负错误码。不推荐使用*/ 操作 I/O 端口 在驱动程序注册I/O 端口后，就可以读/写这些端口。大部分硬件会把8、16和32位端口区分开，不能像访问系统内存那样混淆使用。驱动必须调用不同的函数来存取不同大小的端口。 只支持内存映射的 I/O 寄存器的计算机体系通过重新映射I/O端口到内存地址来伪装端口I/O。为了提高移植性，内核向驱动隐藏了这些细节。Linux 内核头文件(体系依赖的头文件 ) 定义了下列内联函数(有的体系是宏，有的不存在)来访问 I/O 端口: unsigned inb(unsigned port). void outb(unsigned char byte, unsigned port). /*读/写字节端口( 8 位宽 )。port 参数某些平台定义为 unsigned long ，有些为 unsigned short 。 inb 的返回类型也体系而不同。*/ unsigned inw(unsigned port). void outw(unsigned short word, unsigned port). /*访问 16位 端口( 一个字宽 )*/ unsigned inl(unsigned port). void outl(unsigned longword, unsigned port). 