 今天在做一个驱动的时候要用到另一个驱动（I2C）提供的API，在内核初始化时碰到了一个依赖问题。 我的驱动在I2C初始化之前就运行起来了，而这时I2C提供的API还处于不可用状态。查了很多资料，网上有人说所有使用module_init这个宏的驱动程序的起动顺序都是不确定的（我没有查到权威的资料）。 所有的__init函数在区段.initcall.init中还保存了一份函数指针，在初始化时内核会通过这些函数指针调用这些__init函数指针，并在整个初始化完成后，释放整个init区段（包括.init.text，.initcall.init等）。 注意，这些函数在内核初始化过程中的调用顺序只和这里的函数指针的顺序有关，和1）中所述的这些函数本身在.init.text区段中的顺序无关。在2.4内核中，这些函数指针的顺序也是和链接的顺序有关的，是不确定的。在2.6内核中，initcall.init区段又分成7个子区段，分别是 .initcall1.init .initcall2.init .initcall3.init .initcall4.init .initcall5.init .initcall6.init .initcall7.init当需要把函数fn放到.initcall1.init区段时，只要声明core_initcall(fn).即可。 其他的各个区段的定义方法分别是：core_initcall(fn) -.initcall1.init postcore_initcall(fn) -.initcall2.init arch_initcall(fn) -.initcall3.init subsys_initcall(fn) -.initcall4.init fs_initcall(fn) -.initcall5.init device_initcall(fn) -.initcall6.init late_initcall(fn) -.initcall7.init而与2.4兼容的initcall(fn)则等价于device_initcall(fn)。各个子区段之间的顺序是确定的，即先调用.initcall1.init中的函数指针，再调用.initcall2.init中的函数指

针，等等。而在每个子区段中的函数指针的顺序是和链接顺序相关的，是不确定的。 在内核中，不同的init函数被放在不同的子区段中，因此也就决定了它们的调用顺序。这样也就解决了一些init函数之间必须保证一定的调用顺序的问题。按照include/linux/init.h文件所写的，我在驱动里偿试了这样两种方式： __define_initcall("7", fn).late_initcall(fn).都可以把我的驱动调整到最后调用。实际上上面两个是一回事： #define late_initcall(fn) __define_initcall("7", fn) 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