Linux操作系统内核模块与用户程序对比 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_Linux_E6_93_8D_E4_BD_c103_145219.htm 内核模块是如何开始和结束的 用户程序通常从函数main()开始，执行一系列的指令并且当指令执行完成后结束程序。内核模块有一点不同。内核模块要么从函数init_module 或是你用宏module_init指定的函数调用开始。这就是内核模块 的入口函数。它告诉内核模块提供那些功能扩展并且让内核准备好在需要时调用它。当它完成这些后，该函数就执行结束了。模块在被内核调用前也什么都不做。 所有的模块或是调用cleanup_module或是你用宏module_exit指定的函数。这是模块的退出函数。它撤消入口函数所做的一切。 例如注销入口函数所注册的功能。 所有的模块都必须有入口函数和退出函数。既然我们有不只一种方法去定义这两个 函数，我将努力使用“ 入口函数 ”和“ 退出函数 ”来描述 它们。但是当我只用init_module 和cleanup_module时，我希望你明白我指的是什么。 模块可调用的函数 程序员并不总是自己写所有用到的函数。一个常见的基本的例子就是 printf()你使用这些C标准库，libc提供的库函数。这些函数(像printf()) 实际上在连接之前并不进入你的程序。 在连接时这些函数调用才会指向 你调用的库，从而使你的代码最终可以执行。 内核模块有所不同。在hello world模块中你也许已经注意到了我们使用的函数 printk() 却没有包含标准I/O库。这是因为模块是在insmod加 载时才连接的目标文件。那些要用到的函数的符号链接是内核自己提供的。 也就是说，你可以在内核模块中使用的函数只能来自内核本身

。如果你对内核提供了哪些函数符号 链接感兴趣，看一看文件/proc/kallsyms。 需要注意的一点是库函数和系统调用的区别。库函数是高层的，完全运行在用户空间， 为程序员提供调用真正的在幕后完成实际事务的系统调用的更方便的接口。系统调用在内核 态运行并且由内核自己提供。标准C库函数printf()可以被看做是一个通用的输出语句，但它实际做的是将数据转化为符合格式的字符串并且调用系统调用 write()输出这些字符串。 是否想看一看printf()究竟使用了哪些系统调用? 这很容易，编译下面的代码。 #include int main(void){ printf("hello"). return 0. }使用命令gcc -Wall -o hello hello.c编译。用命令 strace hello行该可执行文件。是否很惊讶？每一行都和一个系统调用相对应。 strace[3] 是一个非常有用的程序，它可以告诉你程序使用了哪些系统调用和这些系统调用的参数，返回值。这是一个极有价值的查看程序在干什么的工具。在输出的末尾，你应该看到这样类似的一行 write(1, "hello", 5hello)。这就是我们要找的。藏在面具printf() 的真实面目。既然绝大多数人使用库函数来对文件I/O进行操作(像 fopen, fputs, fclose)。 你可以查看man说明的第二部分使用命令man 2 write. 。man说明的第二部分专门介绍系统调用(像kill()和read())。 man说明的第三部分则专门介绍你可能更熟悉的库函数， (像cosh()和random())。 你甚至可以编写代码去覆盖系统调用，正如我们不久要做的。骇客常这样做来为系统安装后门或木马。 但你可以用它来完成一些更有益的事，像让内核在每次某人删除文件时输出“ Tee hee, that tickles! ”的信息。 用户空间和内核空间 内核全权负责对硬件资源的访问，不管被访问的是显示卡，硬盘，还是内存。 用户程序常为这

些资源竞争。就如同我在保存这份文档同时本地数据库正在更新。 我的编辑器vim进程和数据库更新进程同时要求访问硬盘。内核必须使这些请求有条不紊的进行，而不是随用户的意愿提供计算机资源。 为方便实现这种机制， CPU 可以在不同的状态运行。不同的状态赋予不同的你对系统操作的自由。Intel 80836 架构有四种状态。 Unix只使用了其中 的两种，最高级的状态(操作状态0,即“ 超级状态 ”，可以执行任何操作)和最低级的状态 (即“ 用户状态 ”)。 回忆以下我们对库函数和系统调用的讨论，一般库函数在用户态执行。 库函数调用一个或几个系统调用，而这些系统调用为库函数完成工作，但是在超级状态。 一旦系统调用完成工作后系统调用就返回同时程序也返回用户态。 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