# linux内核模块开发

## 什么是内核模块？

linux内核的整体结构非常庞大，其包含的组件也非常多，如何使用需要的组件呢？

方法一：把所有的组件编译进内核文件，但这样会导致生成的内核文件过大和调整组件不方便。

因此需要我们的内核模块机制，其具有以下特点：

1. 模块本身并不编译进内核文件
2. 可以根据需求，在内核运行期间动态的安装或者卸载

代码示例：

#include<linux/init.h>

#include<linux/module.h>

staticinthello\_init(void)

{

printk(KERN\_WARNING"Hello,world!\n");

return0;

}

staticvoidhello\_exit(void)

{

printk(KERN\_INFO"Goodbye,world\n");

}

module\_init(hello\_init);

module\_exit(hello\_exit);

程序结构

1. 模块加载函数（必需）

通过module\_init宏来制定

1. 模块卸载函数

通过module\_exit宏来制定。

安装与卸载

加载insmod(insmodhello.ko)

卸载rmmod(rmmodhello)

查看lsmod

加载modprobe（modprobehello）

modprobe如同insmod,也是加载一个模块到内核。它的不同之处在于它会根据文件

/lib/modules/<$version>/modules.dep

来查看要加载的模块,看它是否还依赖于其他模块，如果是，modprobe会首先找到这些模块,把它们先加载到内核。

## 对比

对比应用程序，内核模块具有以下不同:应用程序是从头(main)到尾执行任务，执行结束后从内存中消失。内核模块则是先在内核中注册自己以便服务于将来的某个请求，然后它的初始化函数结束，此时模块仍然存在于内核中，直到卸载函数被调用，模块才从内核中消失。

## 常见问题：版本不匹配

内核模块的版本由其所依赖的内核代码版本所决定,在加载内核模块时，insmod程序会程序会将内核模块版本与当前正在运行的内核版本比较，如果不一致时，就会出现类似下面的错误：

insmodhello.ko

disagreesaboutversionofsymbolstruct\_module

insmod:errorinserting'hello.ko':-1Invalidmoduleformat

解决方法：

1、使用modprobe--force-modversion强行插入

2、确保编译内核模块时，所依赖的内核代码版本等同于当前正在运行的内核。

\*\*可通过过uname–r察察看当前运行的内核版本的内核版本\*\*

## 内核打印

Printk是内核中出现最频繁的函数之一，通过将Printk与Printf对对比，将有助于大家理解。

相同点：

打印信息

不同点：

Printk在内核中使用，Printf在应用程序中使用

Printk允许根允许根据严重程度，通过附加不同的“优先级”来对消息分类。