内核模块编程

# 内核模块

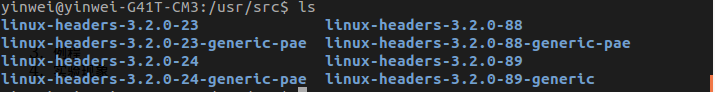
内核可以想象为一个大的进程，或者一个综合功能的模块，这个模块作为硬件和上层应用进行沟通的一个桥梁，为他们提供硬件管理、内存管理、网络管理、硬件驱动等等综合性的功能。 因而，内核是一个庞大的机制，如果我们要在内核上添加功能的话，那么势必要对内核进行编程。面对这个庞然大物，要对其进行手术，那必须对它整个构架了如指掌才能做到得心应手。而针对我们的需求，我们常常需要对内核进行修改或者添加一些功能，从而达到我们的需求。内核模块能够为这个过程提供一个方便快捷的途径。

内核模块的加载是运行在用户空间上，通过module\_init和module\_exit两个宏来完成，当我们使用insmod等来加载模块时，会通过系统调用进入内核空间，从而调用内核内的函数对内核进行修改。

下面基于ubuntu12.04 LST来进行内核模块的简单开发尝试。

# 源码包设置

Linux内核模块的编程是基于内核源码包来进行的，即，在对模块C源程序进行编译的时候，要用到内核的源码包及配置文件。我使用的是ubuntu12.04 LST,ubuntu默认情况下将源码包和配置、工具等分开来存放了，这样就导致当我们对内核源码进行编译的时候总是找不到配置文件：

从上面可以看出，源码包和头文件分开存放了。



如果直接采用源码进行编译的话，会报错，如上图。

因而，在进行内核模块编译前，需要将必要的配置文件和工具放到源码包中：

主要文件有Module.symvers，scripts目录，include/config/auto.config,include/generated目录，将上面的几个文件拷贝到源码包中。当然，首先得知道系统应用的是什么版本的内核，通过命令”uname -a”可打印系统信息：

这是我的配置，可以看出我的内核是3.2.0-88。源码包：linux-headers-3.2.0-88，头文件包：linux-headers-3.2.0-88-generic-pae，这写文件可以在头文件包中找到。A description...

# 例程

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

> File Name: test1.c

> Author:

> Mail:

> Created Time: 2015年09月18日 星期五 09时28分55秒

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <linux/module.h>

#include <linux/init.h>

#include <linux/kernel.h>

MODULE\_LICENSE("GPL");

MODULE\_AUTHOR("YINWEI");

static int \_\_init hello\_init(void)

{

printk("hello word!");

return 0;

}

static void \_\_exit hello\_exit(void)

{

printk("exit the hello!");

}

module\_init(hello\_init);

module\_exit(hello\_exit);

文件：Makefile

obj-m:=test1.o

KDIR:="/usr/src/linux-headers-3.2.0-88/"

CURP:=$(shell pwd)

all:

make -C ${KDIR} M=${CURP} modules

clean:

rm -rf \*.o \*.ko

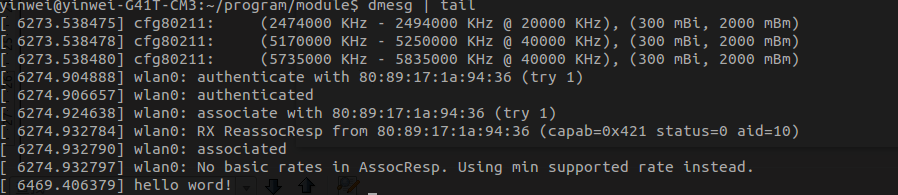
# 实验现象

在终端中，输入：

sudo insmod test1.ko

加载完成，输入：

dmesg | tail

可以看出，模块加载成功。

如果要卸载模块，则输入：

sudo rmmod test1

可以看出，模块卸载成功。