### 实验八 编写内核模块的Helloworld

## 1实验目的

学通过编写自己的内核模块，让学生对编译、插入、卸载内核模块有个基本的了解，通过与应用编程相比，使学生对linux的内核运行过程有个基本全面的了解。

## 2实验内容

* 什么是内核；
* 什么是linux内核模块；
* 认识linux内核模块的作用和应用场景；
* 认识linux内核模块的编写和启用，卸载。

## 3实验原理

3.1内核理论知识

1.(Linux)内核是(Linux)操作系统的核心，一般包含五大部分：进程管理、存储管理、文件管理、设备管理和网络管理，是一组程序模块，具有访问硬件设备和所有主存空间的权限，是仅有的能够执行特权指令的程序。主要功能是：资源抽象、资源分配、资源共享。(资源是指CPU、内存等。)在内核基础上挂载第三方软件便构成操作系统，Ubuntu、RedHat、Fedora、Debian等都是基于Linux内核(版本号可能不同)的不同操作系统。

2.内核模块是Linux内核向外部提供的一个插口，其全称为动态可加载内核模块（Loadable Kernel Module，LKM），称为模块。Linux内核之所以提供模块机制，是因为它本身是一个单内核（monolithic kernel）。单内核的最大优点是效率高，因为所有的内容都集成在一起，但其缺点是可扩展性和可维护性相对较差，模块机制就是为了弥补这一缺陷。模块是具有独立功能的程序，它可以被单独编译，但不能独立运行。它在运行时被链接到内核作为内核的一部分在内核空间运行，这与运行在用户空间的进程是不同的。模块通常由一组函数和数据结构组成，用来实现一种文件系统、一个驱动程序或其他内核上层的功能。

3.2内核模块目录结构

下面是linux内核模块目录存放模块含义（模块目录在实验步骤中可知）：

/arch：存放安装linux的硬件，平台信息的模块如x86，powerpc等，取决于linux运行环境。

/crypto：存放加密、压缩、CRC校验算法的内核模块

/drivers：存放设备驱动程序模块

/fs：存放linux文件系统操作的内核模块

/lib：存放核心的库代码模块

/mm：该目录包含了体系结构无关的内存管理模块。

/net：存放linux系统中的如无线网络，蓝牙，ip协议等网络模块

/sound：存放了声音系统架构的相关模块和具体声卡的设备驱动程序。

3.3内核模块程序与普通c语言程序区别



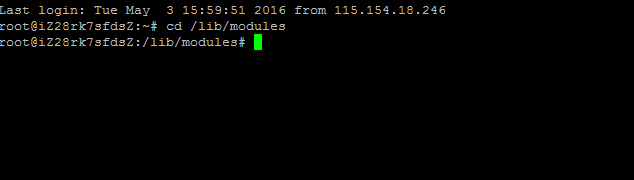
内核模块程序是c语言程序，并且不能调用libc库中的函数，它运行在内核空间，且只有超级用户可以对其运行。另外，模块程序必须通过module\_init()和module\_exit()函数来告诉内核“我来了”和“我走了”。

## 4实验步骤

4.1查看内核模块

步骤1.使用cd命令进入/lib/modules/内核模块目录

命令：[root@localhost Desktop]$ cd /lib/modules/

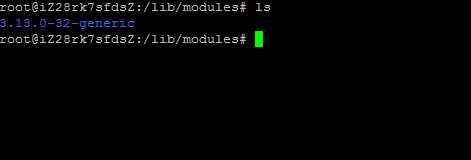


**图表 1执行cd /lib/modules结果**

步骤2.使用ls指令查看目录下内容

命令：[root@localhost Desktop]$ ls

该目录指linux内核版本，实际环境可能不一致

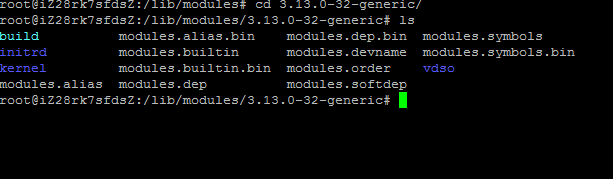


**图表 2执行ls指令结果**

步骤3.使用cd指令进入该目录,并用ls指令查看目录内容

命令：[root@localhost Desktop]$ cd 3.13.0-32-generic/

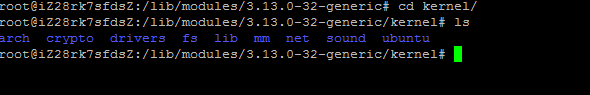
[root@localhost Desktop]$ ls



**图表 3执行cd，ls指令结果**

步骤4.使用cd指令进入kernel目录

命令：[root@localhost Desktop]$ cd kernel/

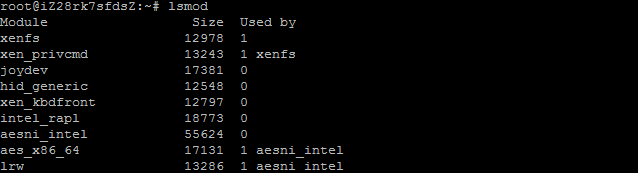


**图表 4执行cd指令结果**

4.2查看运行中的模块

步骤1.使用lsmod指令查看已装载的模块

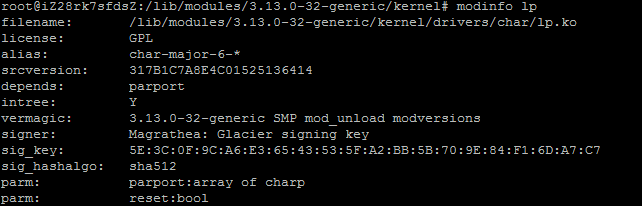
命令：[root@localhost Desktop]$ lsmod



**图表 5使用lsmod指令结果**

步骤2.使用modinfo指令查看一模块信息

命令：[root@localhost Desktop]$ modinfo lp



**图表 6执行modinfo指令结果**

其中每个信息含义如下：

filename：该模块的模块名及所在内核模块目录

license：该模块采用的版权协议

alias：该模块别名，通用识别符

srcversion：该模块版本值，用于内核加载模块时的版本校验。

depends：该模块所依赖的其它模块

vermagic：内核版本信息

signer：模块签名工具，用于模块签名检查机制

sig\_key：签名密钥

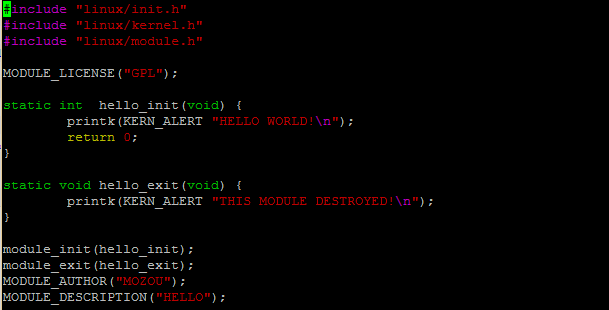
sig\_hashalgo：签名密钥算法

parm：模块加载运行参数

4.3编写自己的helloworld模块

步骤1.编写helloworld模块程序

命令：[root@localhost Desktop]$ vi hello.c



**图7 hello.c文件代码**

1.hello.c文件可以放在任意文件夹之下。所有模块都要使用头文件module.h，此文件必须包含进来，头文件kernel.h包含了常用的内核函数，头文件init.h包含了宏\_init和\_exit，它们允许释放内核占用的内存。

2.尽管不是严格要求，但模块应该指定代码所使用的许可证。内核能够识别的许可证有：“GPL”、“GPL v2”、“GPL and additional rights"、“Dual BSD/GPL"、“GPL MPL/GPL”、“Proprietary”。在我们的模块里我们用“GPL”。

3.hello\_init是模块的初始化函数，它必需包含诸如要编译的代码、初始化数据结构等内容。由于内核程序不能使用c库，所以使用了printk()函数，该函数是由内核定义的，功能与C库中的printf()类似，它把要打印的信息输出到终端或系统日志。

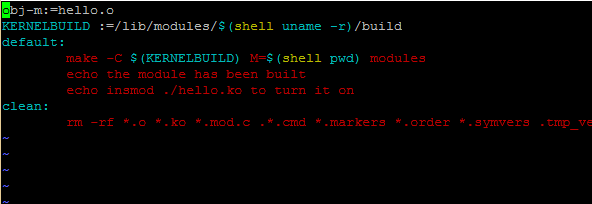
4.hello\_exit是模块的退出和清理函数。此处可以做所有终止该程序时相关的清理工作。

5.module\_init()和module\_exit()是模块编程中最基本也是必须的两个函数，module\_init()是驱动程序初始化的入口点。而module\_exit()注销由模块提供的所有功能。

6.MODULE\_AUTHOR等宏含义在理论知识处有说明。

步骤2.编写makefile文件做程序的编译

命令：[root@localhost Desktop]$ vi Makefile



**图表 8 makefile文件代码**

1.注意makefile里面要求的tab

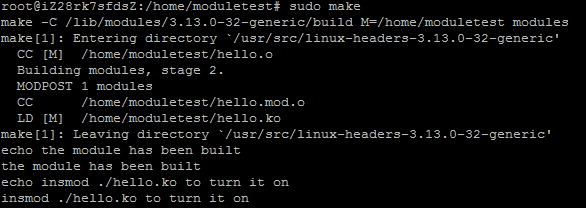
2.Makefile文件与hello.c文件放在同一个目录。

3.KERNELBUILD :=/lib/modules/$(shell uname -r)/build是编译内核模块需要的Makefile的路径，通过$(shell uname -r)取得该机器的内核版本得到正确目录。

4.make -C $(KERNELBUILD) M=$(shell pwd) modules 编译内核模块。-C 将工作目录转到KERNELBUILD，调用该目录下的Makefile，并向这个Makefile传递参数M的值是$(shell pwd) modules。

步骤3.编译模块

命令：[root@localhost Desktop]$ sudo make



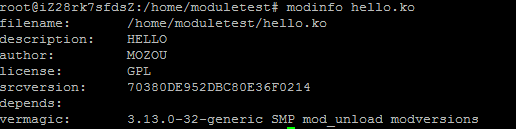
**图表 9 make指令执行结果**

得到hello.ko即为我们需要模块。并输出了代码中的提示内容。并得到一些中间文件，可用Makefile文件中的clean参数清除。命令如下：

[root@localhost Desktop]$ sudo make clean

步骤4.查看编写的helloworld模块信息

命令：[root@localhost Desktop]$ modinfo hello.ko



**图表 10执行modinfo指令结果**

步骤5.加载helloworld模块

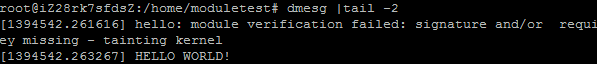
命令：[root@localhost Desktop]$ ismod hello.ko



**图表 11执行ismod指令结果**

步骤6.查看模块运行结果

命令：[root@localhost Desktop]$ dmesg |tail -2



**图表 12执行dmesg指令结果**

或[root@localhost Desktop]$ tail /var/log/kern.log



**图表 13执行tail指令结果**

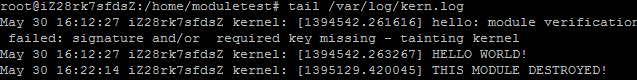
由于模块工作在内核空间，所以不会在shell中直接显示结果。需要到内核工作日志中查看。

步骤7.卸载helloworld模块

命令：[root@localhost Desktop]$ rmmod hello.ko

步骤8.查看卸载模块结果

命令：[root@localhost Desktop]$ tail /var/log/kern.log



**图表 14执行tail指令结果**

4.4参考源码：

hello.c代码：

#include “linux/init.h”

#include “linux/kernel.h”

#include “linux/module.h”

MODULE\_LICENSE(“GPL”);

static int hello\_init(void) {

printk(KERN\_ALERT “HELLO WORLD!\n”);

Return 0;

}

static void hello\_exit(void) {

printk(KERN\_ALERT “THIS MODULE DESTROYED!\n”);

}

module\_init(hello\_init);

module\_exit(hello\_exit);

MODULE\_AUTHOR(“MOZOU”);

MODULE\_DESCRIPTION(“HELLO”);

makefile代码：

obj-m:=hello.o

KERNELBUILD :=/lib/modules/$(shell uname -r)/build

default:

Make -C $(KERNELBUILD) M=$(shell pwd) modules

clean:

rm -rf \*.o \*.ko \*.mod.c .\*.cmd \*.markers

## 5实验总结

经过本次实验，对linux操作系统的内核有了新的理解与认识。认识了内核模块工作的动态装载性，作用在于使得内核尽可能小并具有最大的灵活性，用模块化的思想编写OS。但是这样的设计有可能会对系统性能和内存利用有负面影响，装入的内核模块和其他内核部分一样，具有相同的访问权限，因此，差的内核模块会导致系统崩溃。并为之后编码自己的内核模块写下基础。

在内核模块编码中，对内核的知识有了更直观和深入的认识，了解了部分linux驱动程序的工作原理。对程序在用户空间和内核空间的不同有了一定了解。

## 6参考文献

深入理解UNIX系统内核 [美] Uresh Vahalia著 第2章进程与内核

Linux设备驱动程序第3版 第2章建立和运行模块

## 7实验拓展

编写自己的内核模块，并实现较复杂功能，让模块带参数加载或使用，实现模块导出函数或变量功能。