Proj1 Introduction to Linux Kernel Modules

by 郑航 520021911347

Proj1 Introduction to Linux Kernel Modules
- 1 Introduction
  - 1.1 Objectives
  - 1.2 Environment
  - 1.3 Abstract
- 2 Loading and Removing Kernel Modules
  - 2.1 Basic command
    - 2.1.1 lsmod命令
    - 2.1.2 dmesg命令
    - 2.1.3 insmod命令和rmmod命令
  - 2.2 simple
    - 2.2.1 simple.c
    - 2.2.2 make
    - 2.2.3 load the simple module
    - 2.2.4 remove the simple module
    - 2.2.5 additional functions
  - 2.3 The /proc File System
  - 2.4 Assignment
    - 2.4.1 jiffies module
    - 2.4.2 seconds module
- 3 Review
  - 3.1 Difficulties&Problems
  - 3.2 Summary

1 Introduction

1.1 Objectives

- 初步了解Linux内核模块的运作，学会创建/删除内核模块，以及自己实现一些简单的内核模块

1.2 Environment

- win10下的VMware Workstation Pro中运行的Ubuntu18.04

1.3 Abstract

- 创建并修改simple模块，并打印对应信息
- 创建并修改jiffies模块，并打印jiffies值

  创建并修改seconds模块，并打印seconds值

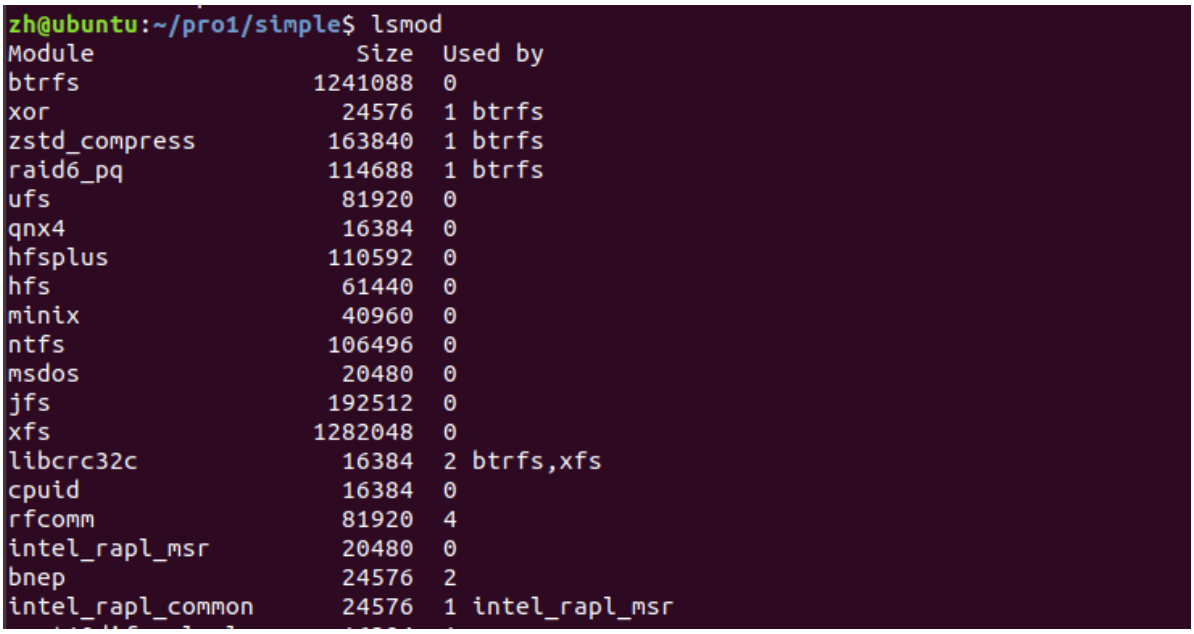
2 Loading and Removing Kernel Modules

2.1 Basic command

2.1.1 lsmod命令

```
lsmod
```

在命令行界面输入该命令，可以列出当前模块的信息，展示模块的名字，大小和使用处



2.1.2 dmesg命令

```
dmesg
```

在命令行界面输入该命令，可以查看硬件信息，在本project中即是展示出我们在内核模块中打印的内容

2.1.3 insmod命令和rmmod命令

```
sudo insmod simple.ko
sudo rmmod simple
```

在命令行界面输入insmod和对应的.ko文件，可以将内核模块加载到内核中；输入rmmod和对应的内核模块，可以将内核模块从内核中移除。

2.2 simple

2.2.1 simple.c

```c
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
/* This function is called when the module is loaded. */
int simple_init(void)
{
printk(KERN_INFO "Loading Kernel Module\n");
return 0;
}
/* This function is called when the module is removed. */
void simple_exit(void)
{
printk(KERN_INFO "Removing Kernel Module\n");
}
/* Macros for registering module entry and exit points. */
module_init(simple_init);
module_exit(simple_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("os proj one");
MODULE_AUTHOR("Hang Zheng");
```

每个模块都需要有模块出入口函数，在simple模块中：

- 函数simple_init()作为 module entry point，当模块被载入内核时该函数被调用。Module entry point函数需要返回一个int，其中0表示加载成功，其他情况则表示加载失败
- 函数simple_exit()作为 module exit point，当模块被移出内核时该函数被调用。Module entry point函数返回void
- 这两个函数都不需要传递任何参数，并且用以下的两个宏进行声明：

```c
module_init(simple_init)
module_exit(simple_exit)
```

需要注意在以下两行中调用的printk()函数，它相当于内核中的printf()，其打印的内容可通过dmesg命令查看

```c
printk(KERN_INFO "Loading Kernel Module\n");
printk(KERN_INFO "Removing Kernel Module\n");
```

其中，KERN_INFO是内核信息的优先级标志

```c
MODULE LICENSE("GPL");
MODULE DESCRIPTION("Simple Module");
MODULE AUTHOR("SGG");
```

这部分是对该模块的一些补充信息

2.2.2 make

在simple文件夹中编写如下的Makefile文件

```
obj-m += simple.o
all:
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

并在命令行中执行

```
make
```

出现以下结果，即说明编译成功

```
zh@ubuntu:~/pro1/simple$ make
make -C /lib/modules/5.4.0-99-generic/build M=/home/zh/pro1/simple modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-99-generic'
  CC [M]  /home/zh/pro1/simple/simple.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/zh/pro1/simple/simple.mod.o
  LD [M]  /home/zh/pro1/simple/simple.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-99-generic'
```

输入ls命令，发现已经出现了一系列文件

```
zh@ubuntu:~/pro1/simple$ ls
Makefile  modules.order  Module.symvers  simple.c  simple.ko  simple.mod  simple.mod.c  simple.mod.o  simple.o
```

2.2.3 load the simple module

执行insmod命令，将simple模块载入内核

```
sudo insmod simple.ko
```

通过dmesg命令查看是否加载成功

```
zh@ubuntu:~/pro1/simple$ sudo insmod simple.ko
zh@ubuntu:~/pro1/simple$ dmesg
[68161.424979] Loading Kernel Module
```

2.2.4 remove the simple module

执行rmmod命令，将simple模块从内核中移除

```
sudo rmmod simple
```

通过dmesg命令查看是否移除成功

```
zh@ubuntu:~/pro1/simple$ sudo rmmod simple
zh@ubuntu:~/pro1/simple$ dmesg
[68102.332917] Removing Kernel Module
```

2.2.5 additional functions

project中，要求对simple.c做一定修改，以完成以下四个功能

- Print out the value of GOLDEN_RATIO_PRIME in the simple_init() function.
- Print out the greatest common divisor of 3,300 and 24 in the simple_exit() function.
- Print out the values of jiffies and HZ in the simple_init() function.
- Print out the value of jiffies in the simple_exit() function.

修改后的simple.c 源代码如下：

```c
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/hash.h>
#include <linux/gcd.h>
#include <linux/jiffies.h>

/* This function is called when the module is loaded. */
int simple_init(void)
{
    printk(KERN_INFO "Loading Kernel Module\n");
    printk("The GOLDEN_RATIO_PRIME: %llu\n", GOLDEN_RATIO_PRIME);
    printk("jiffies: %lu\n", jiffies);
    printk("HZ: %u\n", HZ);
    return 0;
}

/* This function is called when the module is removed. */
void simple_exit(void)
{
    printk(KERN_INFO "Removing Kernel Module\n");
    printk("gcd of 3300 and 24: %lu\n", gcd(3300,24));
    printk("jiffies: %lu\n", jiffies);
}

/* Macros for registering module entry and exit points. */
module_init(simple_init);
module_exit(simple_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("os proj one");
MODULE_AUTHOR("Hang Zheng");
```

结果如下：

```
zh@ubuntu:~/pro1/simple$ sudo insmod simple.ko
zh@ubuntu:~/pro1/simple$ dmesg
[69020.170750] Loading Kernel Module
[69020.170800] The GOLDEN_RATIO_PRIME: 7046029254386353131
[69020.170801] jiffies: 4312146548
[69020.170802] HZ: 250
zh@ubuntu:~/pro1/simple$ sudo dmesg -c
[69020.170750] Loading Kernel Module
[69020.170800] The GOLDEN_RATIO_PRIME: 7046029254386353131
[69020.170801] jiffies: 4312146548
[69020.170802] HZ: 250
zh@ubuntu:~/pro1/simple$ sudo rmmod simple
zh@ubuntu:~/pro1/simple$ dmesg
[69087.945671] Removing Kernel Module
[69087.945675] gcd of 3300 and 24: 12
[69087.945676] jiffies: 4312163491
```

2.3 The /proc File System

需要完成的功能：

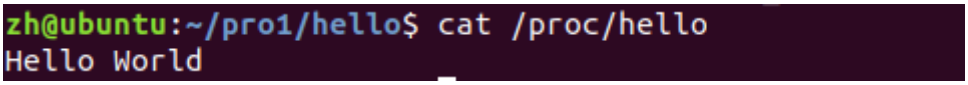
- create a new entry in the /proc file system named /proc/hello
- return the Hello World message when the /proc/hello file is read

hello.c 完整源码如下：

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <asm/uaccess.h>
#include <linux/uaccess.h>
#define BUFFER_SIZE 128

#define PROC_NAME "hello"
#define MESSAGE "Hello World\n"

/**
 * Function prototypes
 */
ssize_t proc_read(struct file *file, char *buf, size_t count, loff_t *pos);

static struct file_operations proc_ops = {
        .owner = THIS_MODULE,
        .read = proc_read,
};


/* This function is called when the module is loaded. */
int proc_init(void)
{

        // creates the /proc/hello entry
        // the following function call is a wrapper for
        // proc_create_data() passing NULL as the last argument
        proc_create(PROC_NAME, 0, NULL, &proc_ops);
```

```
31
32          printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
33
34      return 0;
35  }
36
37  /* This function is called when the module is removed. */
38  void proc_exit(void) {
39
40          // removes the /proc/hello entry
41          remove_proc_entry(PROC_NAME, NULL);
42
43          printk( KERN_INFO "/proc/%s removed\n", PROC_NAME);
44  }
45
46  ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count,
    loff_t *pos)
47  {
48          int rv = 0;
49          char buffer[BUFFER_SIZE];
50          static int completed = 0;
51
52          if (completed) {
53                  completed = 0;
54                  return 0;
55          }
56
57          completed = 1;
58
59          rv = sprintf(buffer, "Hello World\n");
60
61          // copies the contents of buffer to userspace usr_buf
62          copy_to_user(usr_buf, buffer, rv);
63
64          return rv;
65  }
66
67
68  /* Macros for registering module entry and exit points. */
69  module_init( proc_init );
70  module_exit( proc_exit );
71
72  MODULE_LICENSE("GPL");
73  MODULE_DESCRIPTION("Hello Module");
74  MODULE_AUTHOR("Hang Zheng");
```

编译后加载到内核模块中，输入cat /proc/hello，成功输出了“Hello World”，结果如下：



2.4 Assignment

2.4.1 jiffies module

需要完成功能：

- Design a kernel module that creates a /proc file named /proc/jiffies
- reports the current value of jiffies when the /proc/jiffies file is read

该模块的功能其实就是simple模块和hello模块的组合，主要思路是修改hello.c，引入一些头文件用来读取jiffies值，同时修改输出，使得每次访问/proc/jiffies就输出当时的jiffies值。

修改后的jiffies.c源码如下：

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <asm/uaccess.h>
#include <linux/uaccess.h>
#include <linux/jiffies.h>

#define BUFFER_SIZE 128
#define PROC_NAME "jiffies"

/**
 * Function prototypes
 */
ssize_t proc_read(struct file *file, char *buf, size_t count, loff_t *pos);

static struct file_operations proc_ops = {
        .owner = THIS_MODULE,
        .read = proc_read,
};


/* This function is called when the module is loaded. */
int proc_init(void)
{

        // creates the /proc/hello entry
        // the following function call is a wrapper for
        // proc_create_data() passing NULL as the last argument
        proc_create(PROC_NAME, 0, NULL, &proc_ops);

        printk(KERN_INFO "/proc/%s created\n", PROC_NAME);

    return 0;
}

/* This function is called when the module is removed. */
void proc_exit(void) {

        // removes the /proc/hello entry
        remove_proc_entry(PROC_NAME, NULL);

```

```
43          printk( KERN_INFO "/proc/%s removed\n", PROC_NAME);
44  }
45
46  ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count,
    loff_t *pos)
47  {
48          int rv = 0;
49          char buffer[BUFFER_SIZE];
50          static int completed = 0;
51
52          if (completed) {
53                  completed = 0;
54                  return 0;
55          }
56
57          completed = 1;
58      char message[BUFFER_SIZE];
59      sprintf(message, "%lu\n", jiffies);
60          rv = sprintf(buffer, message);
61
62          // copies the contents of buffer to userspace usr_buf
63          copy_to_user(usr_buf, buffer, rv);
64
65          return rv;
66  }
67
68
69  /* Macros for registering module entry and exit points. */
70  module_init( proc_init );
71  module_exit( proc_exit );
72
73  MODULE_LICENSE("GPL");
74  MODULE_DESCRIPTION("Jiffies Module");
75  MODULE_AUTHOR("Hang Zheng");
```

输出结果如下：

```
zh@ubuntu:~/pro1/jiffies$ sudo insmod jiffies.ko
zh@ubuntu:~/pro1/jiffies$ sudo dmesg -c
[75352.047977] /proc/jiffies created
zh@ubuntu:~/pro1/jiffies$ cat /proc/jiffies
4313733495
zh@ubuntu:~/pro1/jiffies$ sudo rmmod jiffies
zh@ubuntu:~/pro1/jiffies$ sudo dmesg -c
[75373.566619] /proc/jiffies removed
```

2.4.2 seconds module

需要完成功能：

- Design a kernel module that creates a /proc file named /proc/seconds
- reports the number of elapsed seconds since the kernel module was loaded

该模块的功能其实是在jiffies模块的功能基础上再进行一个简单计算即可，主要思路是保存加载如内核时的jiffies值，每次打开文件时读取当前jiffies值，两个jiffies值的差除以HZ即得seconds值

修改后的seconds.c源码如下：

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <asm/uaccess.h>
#include <linux/uaccess.h>
#include <linux/jiffies.h>

#define BUFFER_SIZE 128
#define PROC_NAME "seconds"

/**
 * Function prototypes
 */
ssize_t proc_read(struct file *file, char *buf, size_t count, loff_t *pos);
unsigned long Init_jiffies;

static struct file_operations proc_ops = {
        .owner = THIS_MODULE,
        .read = proc_read,
};


/* This function is called when the module is loaded. */
int proc_init(void)
{

        // creates the /proc/hello entry
        // the following function call is a wrapper for
        // proc_create_data() passing NULL as the last argument
        proc_create(PROC_NAME, 0, NULL, &proc_ops);
    Init_jiffies=jiffies;
        printk(KERN_INFO "/proc/%s created\n", PROC_NAME);

    return 0;
}

/* This function is called when the module is removed. */
void proc_exit(void) {

        // removes the /proc/hello entry
        remove_proc_entry(PROC_NAME, NULL);

        printk( KERN_INFO "/proc/%s removed\n", PROC_NAME);
}

ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count,
loff_t *pos)
{
        int rv = 0;
        char buffer[BUFFER_SIZE];
        static int completed = 0;
```

```
52
53          if (completed) {
54                  completed = 0;
55                  return 0;
56          }
57
58          completed = 1;
59      unsigned long seconds=0;
60      seconds=(jiffies-Init_jiffies)/HZ;
61      char message[BUFFER_SIZE];
62      sprintf(message, "%lu\n", seconds);
63          rv = sprintf(buffer, message);
64
65          // copies the contents of buffer to userspace usr_buf
66          copy_to_user(usr_buf, buffer, rv);
67
68          return rv;
69  }
70
71  /* Macros for registering module entry and exit points. */
72  module_init( proc_init );
73  module_exit( proc_exit );
74
75  MODULE_LICENSE("GPL");
76  MODULE_DESCRIPTION("Jiffies Module");
77  MODULE_AUTHOR("Hang Zheng");
```

输出结果如下：

```
zh@ubuntu:~/pro1/seconds$ sudo insmod seconds.ko
zh@ubuntu:~/pro1/seconds$ dmesg
[76344.331122] /proc/seconds created
zh@ubuntu:~/pro1/seconds$ cat /proc/seconds
11
zh@ubuntu:~/pro1/seconds$
zh@ubuntu:~/pro1/seconds$ cat /proc/seconds
21
zh@ubuntu:~/pro1/seconds$ cat /proc/seconds
34
zh@ubuntu:~/pro1/seconds$ sudo rmmod seconds
zh@ubuntu:~/pro1/seconds$ dmesg
[76344.331122] /proc/seconds created
[76386.489310] /proc/seconds removed
```

3 Review

3.1 Difficulties&Problems

- Makefile文件的编写：第一次写Makefile文件，通过查找资料学习了一下基础的语法，最终是通过课本附带的代码资源完成了这个Makefile文件。一开始make时出现错误“Makefile:3: *** missing separator. Stop.”，通过查找资料，排查后发现是Makefile文件中命令前缺少Tab键导致，修改后即可正确编译。

- simple module：第一次加载simple module时失败，提示错误“module verification failed: signature and/or required key missing - tainting kernel”，查找资料后提示这是签名错误的问题，将simple module移除后再次加载进内核即顺利解决问题。
- hello module中，hello.c中的proc_read()函数，调用了copy_to_user()函数，而在make过程中，报错提示”error: implicit declaration of function ‘copy_to_user’; did you mean ‘raw_copy_to_user’?“。根据提示，修改函数名为raw_copy_to_user后即可正确编译并实现功能。后查询资料得，这是隐式声明函数的问题，一般是缺少相应头文件。在hello.c中添加头文件 #include <linux/uaccess.h>后，改回copy_to_user也可正确编译。

3.2 Summary

Project one从最基础的内核模块概念开始，介绍了内核模块的组成结构及其编写要求，介绍了内核模块加载与移除的方法，并引导实现了一些简单的功能模块。总体来说，难度比较低，但非常的有启发性，让我对内核模块的运作方式有了更具体的了解，也借机学习了vim和make的相关知识，对Linux下编程有了更大的兴趣。