操作系统课程设计 Project 1
Introduction to Linux Kernel Modules

姓名：郭倩昀
班级：F1903303
学号：519021910095
Email：guoqianyun@sjtu.edu.cn

2021 年 10 月 16 日

目录

1 Introduction to Linux Kernel Modules

1.1 实验内容与目标

- 熟悉内核模块
- 内核模块的创建、加载与卸载
- 设计内核模块，给/proc 文件系统创建新条目

1.2 实验过程及步骤

- 配置环境
  本次 project 需要在 Linux 系统的虚拟机环境下完成，因此首先安装虚拟机 virtualbox，并安装 Ubuntu 系统，鉴于版本兼容原因我选择了安装 ubuntu-16.04.7-desktop-amd64 版本，进行了部分必要的软件更新操作。

- 熟悉内核模块
  在 ubuntu 系统中输入 lsmod 可以查看目前系统加载的所有内核模块，包括 module name, size, used by 等信息。
  分析源代码 simple.c 文件熟悉基本内核模块的组成部分，熟悉模块入口点和模块出口点函数。
  simple.c 文件需要经过 Makefile 编译，命令行语句输入 make 即可完成。

- 修改内核模块
  根据 project 要求，为了在 simple_init() 中输出 GOLDEN_RATIO_PRIME、jiffies 和 HZ，在函数中添加语句：
  printk(KERN_INFO " GOLDEN_RATIO_PRIME = %lu\n", GOLDEN_RATIO_PRIME);
  printk(KERN_INFO " jiffies = %lu\n",jiffies);
  printk(KERN_INFO " HZ = %lu\n",HZ);
  为了在 simple_exit() 函数中输出 3300 和 24 的最大公约数以及 jiffiles 数值，在函数中添加语句：
  printk(KERN_INFO " gcd = %lu\n",gcd(3300,24));
  printk(KERN_INFO " jiffies = %lu\n",jiffies);
  修改后的代码见实验代码部分。

- 加载和卸载内核模块
  首先可以通过命令 sudo dmesg -C 来清空内核日志缓冲区，其中 sudo 作为指令前缀可以获取管理员权限，但此时需要输入密码，或者可以登录终端的时候输入 sudo su root 来切换到管理员权限。
  输入 sudo insmod simple.ko 可以加载内核模块 simple，输入 dmesg 可以查看到内核日志信息“Loading Module simple”以及 project 要求在 simple_init() 输出的任务，表示内核模块加载成功；输入 sudo rmmod simple 可以卸载内核模块，再次输入 dmesg 可以查看到内核日志信息“Removing Module simple”以及 project 要求在 simple_exit() 输出的任务，表示内核模块卸载成功。
  实验测试结果见实验测试部分。

- 熟悉/proc 文件系统内核模块
  以教材中源代码 hello.c 为例熟悉/proc 文件系统内核模块的基本组成部分和对应功能。hello 模块加载后输入命令 cat /proc/hello 会输出 Hello World。

- 设计/proc 文件系统内核模块 jiffies 和 seconds
  模仿 hello 模块设计 jiffies 模块和 seconds 模块，增加相应的头文件，在 proc_read 函数中修改输出为相应的 jiffies 和 seconds(根据 proc_init 和 proc_read 两次读取的 jiffies 计算得出) 。测试时首先用 Makefile 编译模块的时候注意需要将文件名称修改为编译对象名称。
  测试的时候内核加载和卸载同之前的 simple 模块一样，加载了模块后输入命令 cat /proc/jiffies 以及 cat /proc/seconds 会输出相应的任务结果。具体代码和测试结果见实验代码和实验测试部分。

1.3 实验代码

simple.c

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

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

/* This function is called when the module is removed. */
void simple_exit(void) {
    printk(KERN_INFO "Removing Module simple\n");
    printk(KERN_INFO "  gcd = %lu\n",gcd(3300,24));
        printk(KERN_INFO "  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("Simple Module");
MODULE_AUTHOR("GQY");
```

jiffies.c

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

```c
#include <linux/uaccess.h>

#define BUFFER_SIZE 128
#define PROC_NAME "jiffies"

ssize_t proc_read(struct file *file,char __user *usr_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)
{
    proc_create(PROC_NAME,0666,NULL,&proc_ops);
    printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
    return 0;
}

void proc_exit(void)
{
    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;
    if (completed){
      completed=0;
      return 0;
    }
    completed=1;

    rv=sprintf(buffer, "Now the jiffies value is %lu\n", jiffies);
    copy_to_user(usr_buf,buffer,rv);
    return rv;
}
module_init(proc_init);
module_exit(proc_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Display the jiffies value");
MODULE_AUTHOR("GQY");
```

seconds.c

```c
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/proc_fs.h>
#include <linux/sched.h>
```

```c
#include <linux/uaccess.h>
#include <linux/jiffies.h>/* jiffies*/
#include <asm/param.h> /* HZ*/

#define BUFFER_SIZE 128
#define PROC_NAME "seconds"


unsigned long int volatile jiffies1,jiffies2;
const int hz=HZ;

ssize_t proc_read(struct file *file,char __user *usr_buf,size_t count,loff_t *pos);


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

/* when the module loaded*/
int proc_init(void)
{
    proc_create(PROC_NAME,0666,NULL,&proc_ops);
    jiffies1 = jiffies;
    printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
    return 0;
}

void proc_exit(void)
{
    remove_proc_entry(PROC_NAME,NULL);
    printk( KERN_INFO "/proc/%s removed\n", PROC_NAME);
}

/*Implemention of proc_read*/
ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count, loff_t *pos)
{
    jiffies2 = jiffies;
    int rv=0;
    char buffer[BUFFER_SIZE];
    static int completed=0;
    if (completed){
      completed=0;
      return 0;
    }
    completed=1;
    rv=sprintf(buffer, "The running time is %d s\n", ((jiffies2-jiffies1)/hz));

    copy_to_user(usr_buf,buffer,rv);
    return rv;
}
module_init(proc_init);
module_exit(proc_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Show the program execution time");
MODULE_AUTHOR("GQY");
```

1.4 实验测试

- 测试 simple 模块 (图 1)

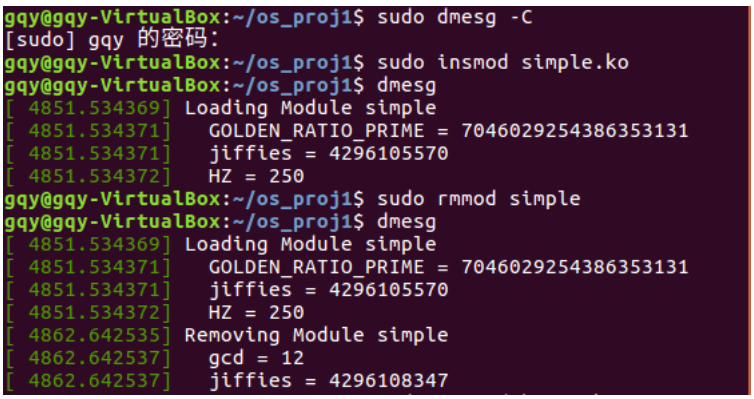

图 1: 测试 simple 模块

- 测试 hello 模块 (图 2)

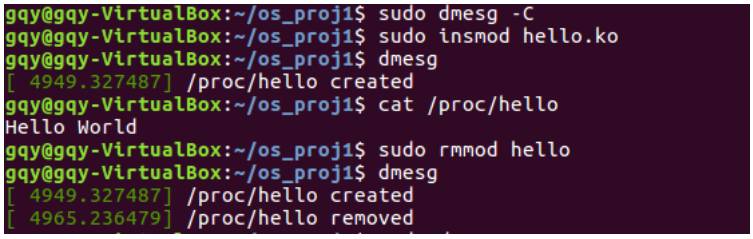

图 2: 测试 hello 模块

- 测试 jiffies 模块 (图 3)

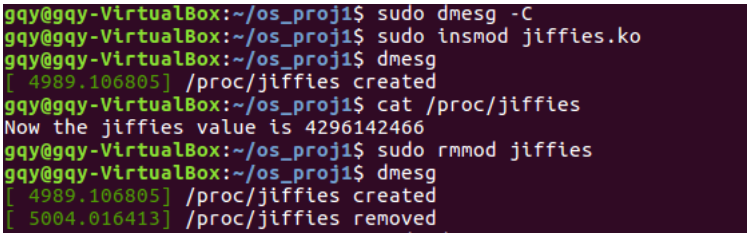

图 3: 测试 jiffies 模块

- 测试 seconds 模块 (图 4)

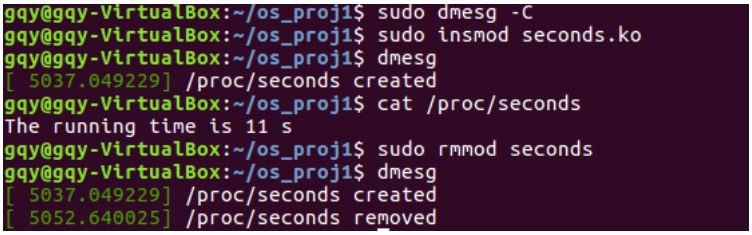

图 4: 测试 seconds 模块

2 Conclusion

2.1 问题与解决方案

本次 project 的内容基本上比较简单，一步步按照书上的指示就可以顺利完成。不过实验过程中的还是出现了一些小问题，首先就是在编译内核模块的时候，因为课本给予的示例是给予旧版本的内核，但由于 ubuntu 系统版本以及内核版本的变化，部分函数（如 proc_create）的定义也有了变化，同时有一些头文件的已经更新导致无法再正常使用，所以我通过报错信息上网查阅了相关的资料，替换了头文件之后成功编译通过。

2.2 实验心得

本次 project 是我第一次使用虚拟机，第一次使用 linux 系统并成功进行了有关内核模块的一系列操作。实验整体过程比较顺利，中间出现的一点小问题也都顺利解决了。这些小问题的出现也告诉我们，在面对技术的不断发展，要以发展的视角看待问题，在遇到问题的时候耐心思考，积极寻求帮助，积极探索问题的解决方案。总体来说本次实验中，我对 linux 系统内核模块有了更深入的了解，在实践中应用理论知识，在不断试错的过程中一步步完成任务并对本课程的学习更加有信心。同时非常感谢吴老师和助教们对课程的悉心指导！