**实验2.3 内核模块 实验报告**

1. **实验目的**

模块是Linux系统的一种特有机制，可用以动态扩展操作系统内核功能。编写实现某些特定功能的模块，将其作为内核的一部分在管态下运行。本实验通过内核模块编程在/porc文件系统中实现系统时钟的读操作接口。

1. **实验内容**

设计并构建一个在/proc文件系统中的内核模块clock，支持read()操作，read()返回值为一字符串，其中包块一个空各分开的两个子串，为别代表xtime.tv\_sec和xtime.tv\_usec。

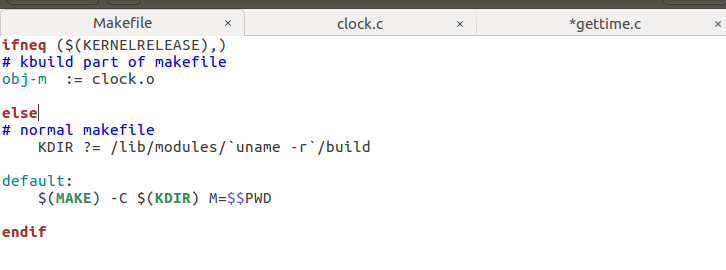
1. **实验设计原理**

Linux模块是一些可以作为独立程序来编译的函数和数据类型的集合。在装载这些模块式，将它的代码链接到内核中。Linux模块可以在内核启动时装载，也可以在内核运行的过程中装载。如果在模块装载之前就调用了动态模块的一个函数，那么这次调用将会失败。如果这个模块已被加载，那么内核就可以使用系统调用，并将其传递到模块中的相应函数。

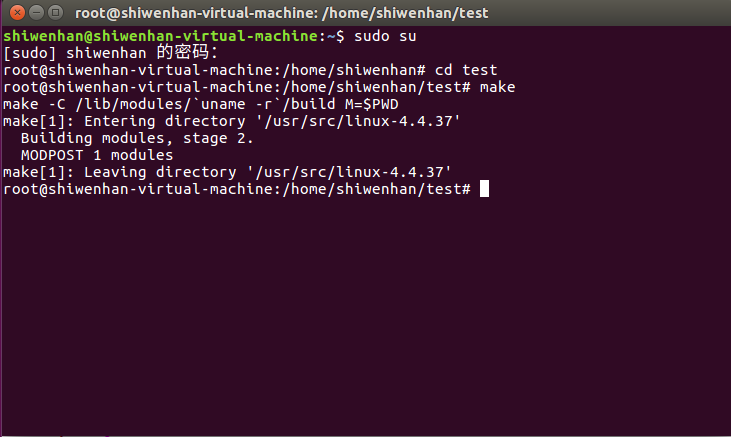
1. **实验步骤**

* **编写内核模块**

文件中主要包含init\_module()，cleanup\_module()，read\_clock()三个函数。其中init\_module()，cleanup\_module()负责将模块从系统中加载或卸载，以及增加或删除模块在/proc中的入口。read\_clock()负责产生/proc/clock被读时的动作。

* **编译内核模块Makefile文件**

编译完成之后生成clock.ko模块文件。



**注意：由于内核版本的不同，在makefile和clock.c文件中都做了相应的修改，以适应该主机上的Linux内核。如实验指导书上给出的create\_proc\_entry函数已经被替换为proc\_create。**

**注意：实验指导书给出的makefile和clock程序均有错误，且一些指令不再适用于本实验所用的新Linux内核版本（4.0），包括makefile中的命令缩进，-符号的使用等，和clock文件中因word格式而造成的字符错误等，在接下来的程序中尽量予以修改。**

* **内核模块源代码clock.c**

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/proc\_fs.h>

#include <linux/string.h>

#include <linux/vmalloc.h>

#include <asm/uaccess.h>

#define MODULE

#define MODULE\_VERSION "1.0"

#define MODULE\_NAME "clock"

int read\_clock(struct file \* flip, char \_\_user \* buffer, size\_t size, loff\_t \* p)

{

// 读接口，使用新版的形参列表，从设备中获取数据

int len;

struct timeval xtime;

do\_gettimeofday(&xtime);

len = sprintf(buffer, "%d %d\n", xtime.tv\_sec, xtime.tv\_usec);

printk("Get time.\n");

return len;

}

static struct proc\_dir\_entry \*proc\_dir = NULL;

static struct proc\_dir\_entry \*proc\_file = NULL;

// 建立读接口和模块的关联

struct file\_operations proc\_fops =

{

.read=read\_clock

};

// 建立这个模块

int init\_clock(void)

{

proc\_file = proc\_create("clock", 0600, proc\_dir, &proc\_fops);

return 0;

}

void exit\_clock(void)

{

remove\_proc\_entry("clock",proc\_file);

}

module\_init(init\_clock)

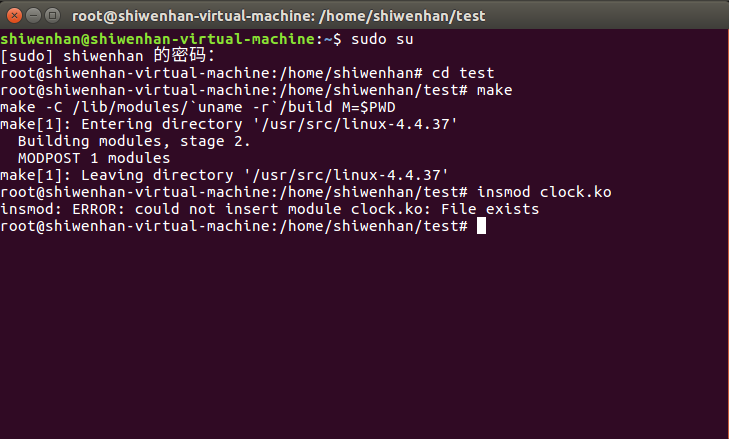
module\_exit(exit\_clock)

MODULE\_LICENSE("GPL");

**请注意，为了适应版本，我们改变了proc\_create接口，并重新根据规范重新设计了read\_clock这个读接口。在file\_operations里面将二者关联在一起。**

* **加载内核模块**

在系统root用户下运行用户态模块命令装载内核模块

 #insmod clock.ko

可以看到clock.ko已经被加载过，因此显示File exists。

* **测试**

测试源代码gettime.c

#include <stdio.h>

#include <sys/time.h>

#include <fcntl.h>

int

main(void)

{

struct timeval getSystemTime;

char procClockTime[256];

int infile,len;

gettimeofday(&getSystemTime,NULL);

infile = open("/proc/clock",O\_RDONLY);

len = read(infile,procClockTime,256);

close(infile);

procClockTime[len] = '\0';

printf("SystemTime is %d %d\nProcClockTime is %s\n",

getSystemTime.tv\_sec ,

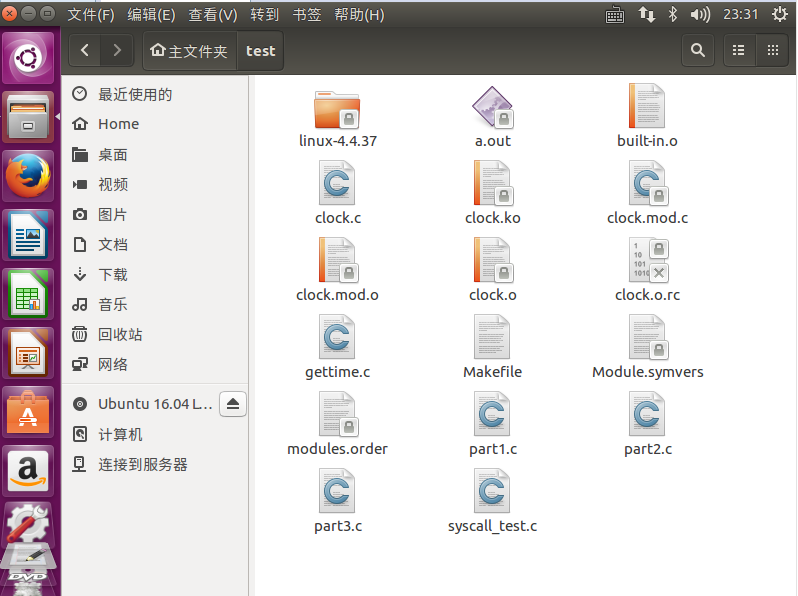
getSystemTime.tv\_usec,

procClockTime

);

sleep(1);

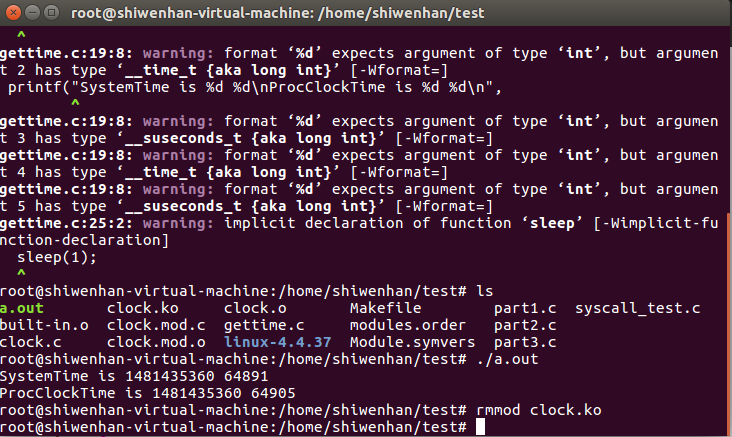
}

目前为止，test目录下的文件为：

具体测试结果请见实验结果分析。

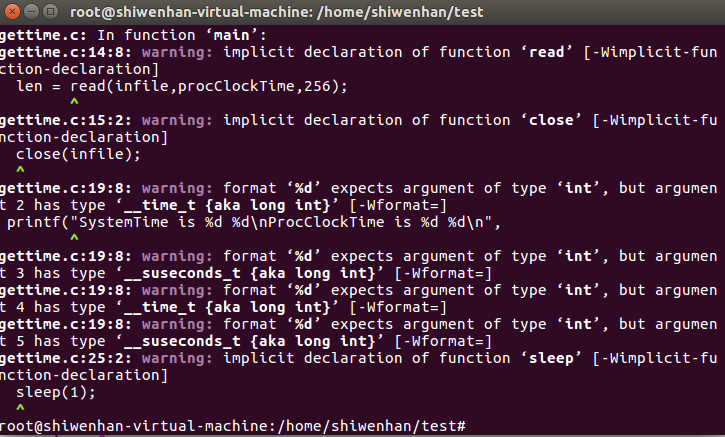
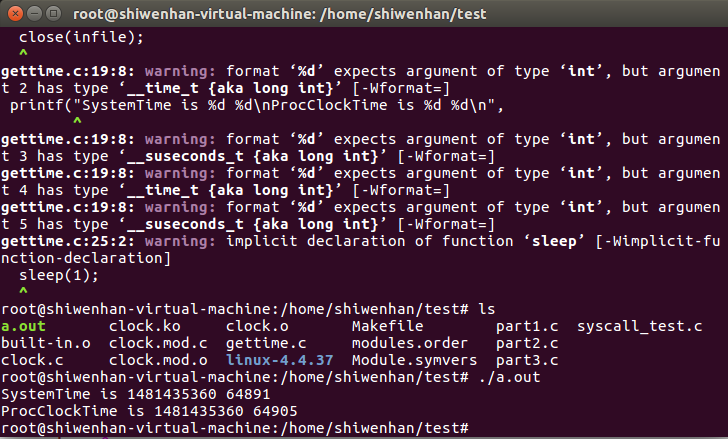
* **卸载内核模块**

在系统root用户下运行用户态模块命令卸载内核模块

#rmmod clock.ko

至此，clock.ko模块被卸载。

1. **实验结果及分析**

运行由gettime.c编译而成的a.out可得到如下结果：

可以看出两个时间并没有太大差异，也就是说我们确实完成了系统时间的读操作（即相同的功能，但是调用接口有差别）。

1. **人员任务分配**

本次实验由小组成员

史文翰

No.2014211218

Cla.2014211304

独自完成。