**操作系统实验1**

1. **实验目的**

通过两个Linux系统下程序的编写、运行，学习如何在Linux系统下创建一个内核模块，并将模块加载到内核中。以及初步了解Linux系统下的基础操作。

1. **实验过程**
2. 项目一

在simple.c和其对应的Makefile文件所在目录下，在终端中执行”make”，完成对simple.c文件的编译，得到simple.ko文件，即编译完成的内核模块。

在终端中执行”sudo insmod simple.ko”，将simple.ko模块加载到内核中。

在终端中执行“dmesg”，查看内核日志缓冲区，来检查simple.ko是否被正确加载。

在内核日志缓冲区中看到“Loading Module.”，表明加载成功。

在终端中执行“sudo rmmod simple”，卸载模块。

在终端中执行“sudo dmesg -c”，清空内核日志缓冲区。

1. 项目二

编写程序，定义birthday结构体。

在模块入口点创建链表，包含5个birthday类型的元素，并分配内存。

对每个元素进行赋值，添加到链表中。

遍历链表，输出表中元素。

在模块退出点从链表中删除元素，并释放之前分配的内存。

程序代码如下：

// birthday.c

#include<linux/init.h>

#include<linux/module.h>

#include<linux/kernel.h>

#include<linux/printk.h>

#include<linux/list.h>

#include<linux/types.h>

#include<linux/slab.h>

/\* Define a struct called birthday. \*/

struct birthday

{

int day;

int month;

int year;

struct list\_head list;

};

/\* Declear a list\_head obeject, which we use as a reference to the head of the list by using the following macro. \*/

static LIST\_HEAD(birthday\_list);

/\* The module entry point. \*/

int birthday\_init(void)

{

printk(KERN\_INFO "Loading Module\n");

/\* Create and initialize instances of struct birthday. \*/

struct birthday \*person[5];

int i = 0;

/\* Insert 5 elements to the linked list. \*/

for(i=0; i<5; i++)

{

/\* Allocate kernel memory. \*/

person[i] = kmalloc(sizeof(\*person[i]), GFP\_KERNEL);

/\* Set values. \*/

person[i]->day = i;

person[i]->month = i;

person[i]->year = 2000+i;

/\* Initialize member list. \*/

INIT\_LIST\_HEAD(&person[i]->list);

/\* Add this instance to the end of the liked list. \*/

list\_add\_tail(&person[i]->list, &birthday\_list);

}

/\* Declear a pointer. \*/

struct birthday \*ptr;

/\* Traverse the linked list by using the following macro. \*/

list\_for\_each\_entry(ptr, &birthday\_list, list)

{

/\* Output to the kernel log buffer. \*/

printk(KERN\_INFO "%d/%d/%d\n", ptr->day, ptr->month, ptr->year);

}

return 0;

}

/\* The module exit point. \*/

void birthday\_exit(void)

{

printk(KERN\_INFO "Removing Module\n");

/\* Declear two pointers. \*/

struct birthday \*ptr, \*next;

/\* Traverse the linked list by using the following macro. \*/

list\_for\_each\_entry\_safe(ptr, next, &birthday\_list, list)

{

/\* Remove the elements from the linked list by using the following macro. \*/

list\_del(&ptr->list);

/\* Return the memory that was previously allocated back to the kernel. \*/

kfree(ptr);

}

}

/\* Macros for registering module entry and exit points. \*/

module\_init( birthday\_init );

module\_exit( birthday\_exit) ;

MODULE\_LICENSE("GPL");

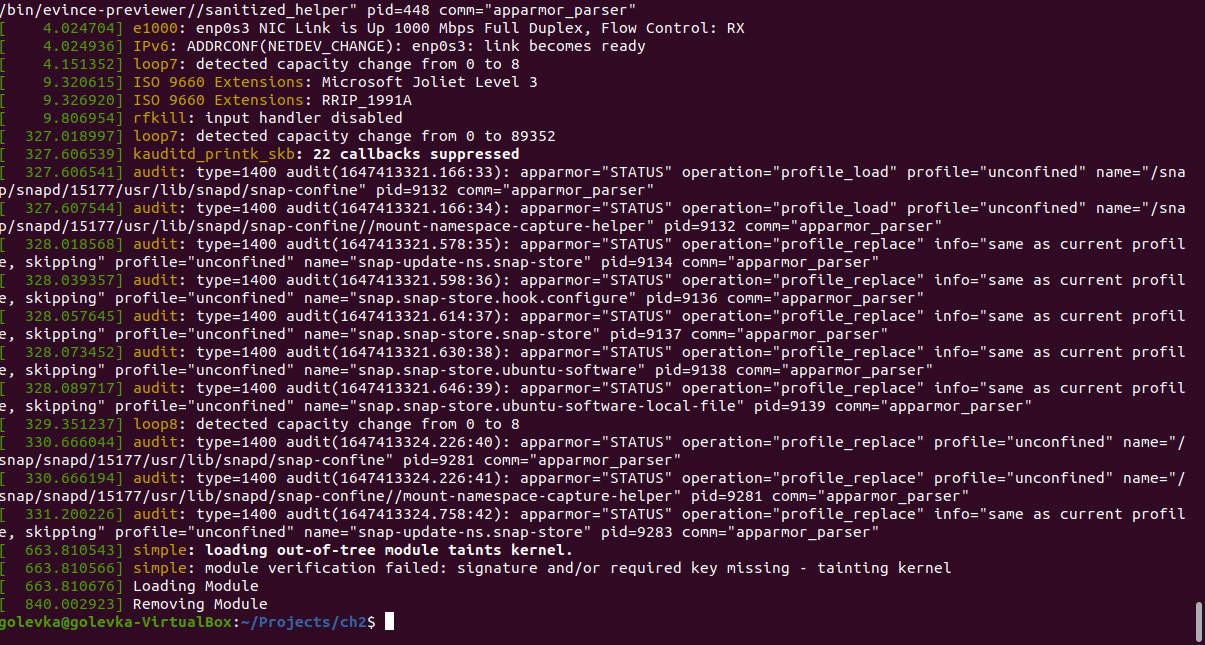
MODULE\_DESCRIPTION("Birthday Module");

MODULE\_AUTHOR("Guangwei Li");

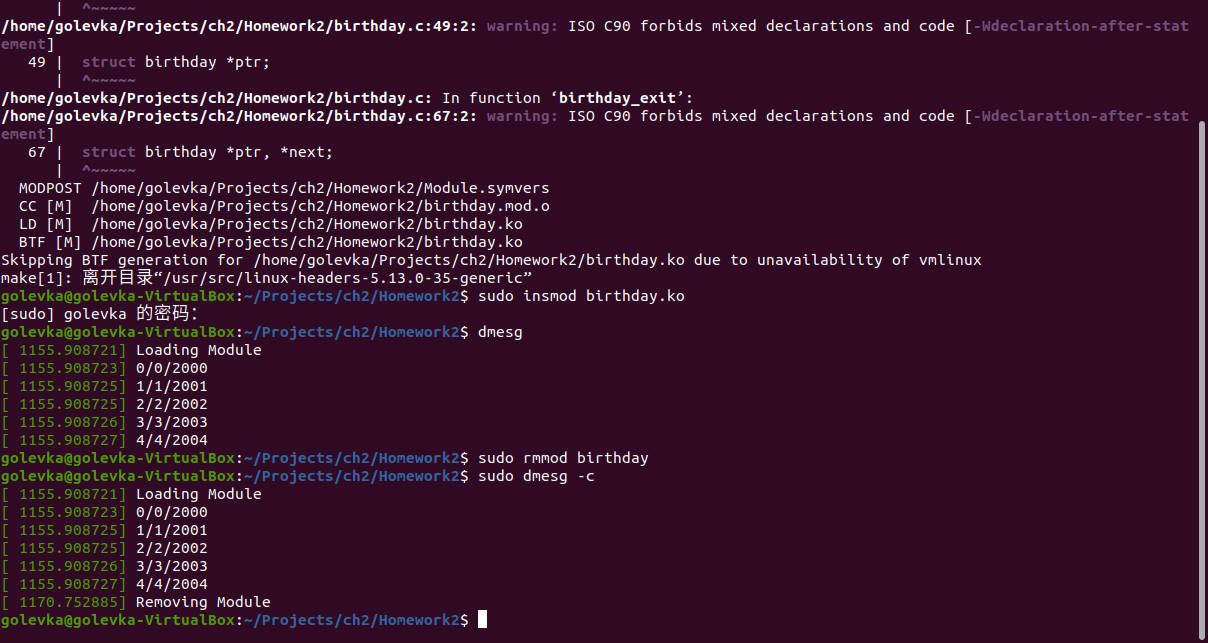
在birthday.c文件同一路径下，创建对应的Makefile文件。

在终端中执行与项目一类似的操作， 在内核日志缓冲区中看到相应消息，表明模块编写以及加载、卸载操作成功。

1. **实验结果**
   1. 项目一



* 1. 项目二



1. **实验小结**

项目一较为简单，了解了Makefile文件的作用以及编译的方法。

项目二需要自己编写或改写程序，遇到一些困难，主要在于由于没有图形界面，终端中文本编辑器的使用不熟练（后来还是改用了带图形界面的），以及在编写程序过程中的错误不会被及时报告，需要等到编译时才能看到报错，比较头疼。不过目前遇到的问题都已基本解决，同时也在实验中学习了很多Linux系统的使用方法。