上海交通大學

SHANGHAI JIAOTONG UNIVERSITY

操作系统课程设计报告 - Project1

Introduction to Linux kernel

姓名：薛春宇

学号：518021910698

完成时间：2020/5/22

一、实验目的

1. 了解 Linux kernel 的编译方法和过程，加深自身对 Linux 原理的了解

2. 学习如何编写 Linux kernel module，以及如何将这些模块插入内核中，以扩展操作系统的功能

二、实验内容

1. 在虚拟机上编译新的 Linux 内核并安装、启动；

2. 分别编写两个 kernel modules：

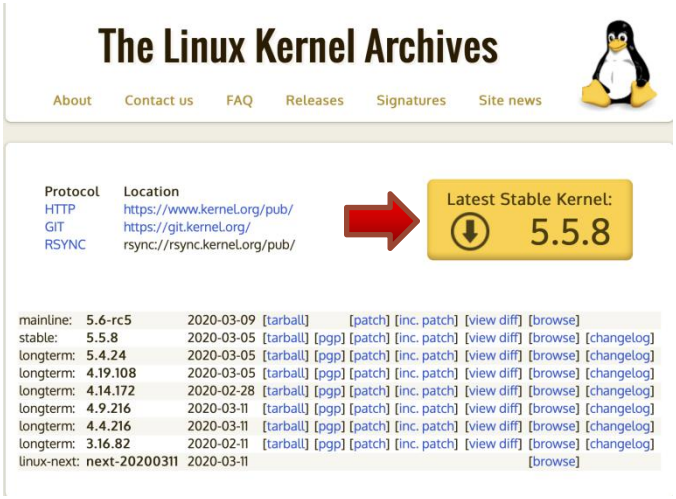
   （1）报告 jiffies 的当前值；

   （2）报告当前运行秒数；

三、内容一：编译安装 Linux 内核

1. 下载内核源码：

打开 Ubuntu，前往网站 https://www.kernel.org 下载内核源码（.tar.xz）到桌面上。



注意：本次试验 Ubuntu 版本为 ubuntu-19.10-desktop-amd64，Linux kernel 版本为 linux-5.5.8

2. 解压：

在桌面右键打开 Ubuntu 终端，执行命令将文件解压到桌面：

```
apple@ubuntu:~$ sudo tar -xavf linux-5.5.8.tar.xz
[sudo] apple 的密码：
```

3. 安装必要工具：

在终端内执行以下安装命令：

```
apple@ubuntu:~$ sudo apt-get install gcc make libncurses5-dev openssl libssl-dev
```

```
apple@ubuntu:~$ sudo apt-get install build-essential
```

```
apple@ubuntu:~$ sudo apt-get install pkg-config
```

```
apple@ubuntu:~$ sudo apt-get install libc6-dev
```

```
apple@ubuntu:~$ sudo apt-get install bison
```

```
apple@ubuntu:~$ sudo apt-get install flex
```

```
apple@ubuntu:~$ sudo apt-get install libelf-dev
```

注意：若出现如下情况，说明该组件已经安装到最新版本。

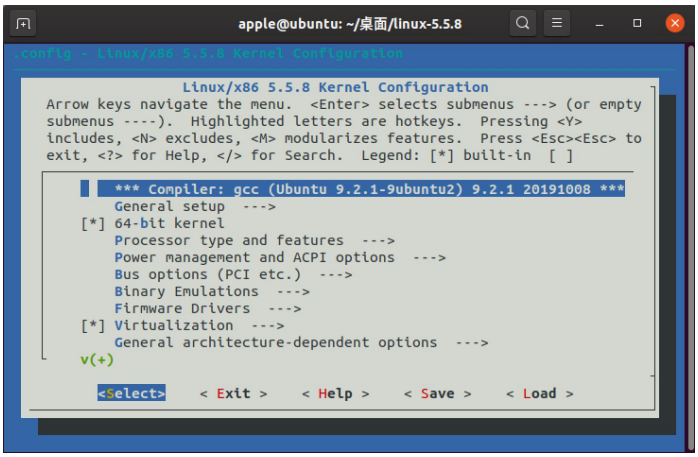
```
apple@ubuntu:~$ sudo apt-get install libelf-dev
[sudo] apple 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
libelf-dev 已经是最新版 (0.176-1.1)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 50 个软件包未被升级。
```

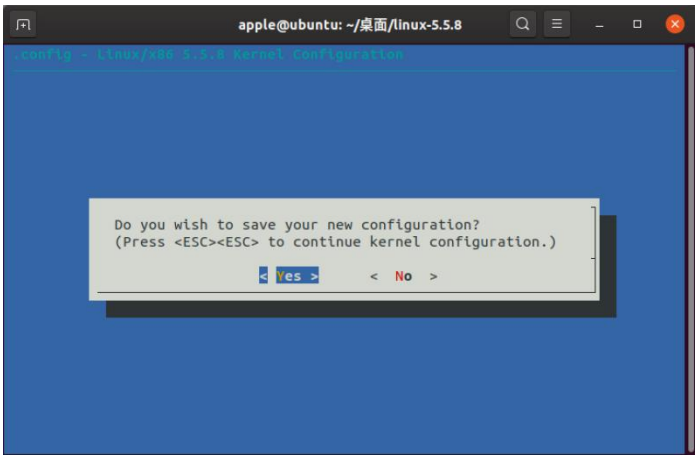
4. 开始准备编译：

在终端中打开已经解压在桌面的 linux-5.5.8 文件，并执行以下命令：

```
apple@ubuntu:~/桌面/linux-5.5.8$ sudo make menuconfig
```

输入密码后会出现下图：





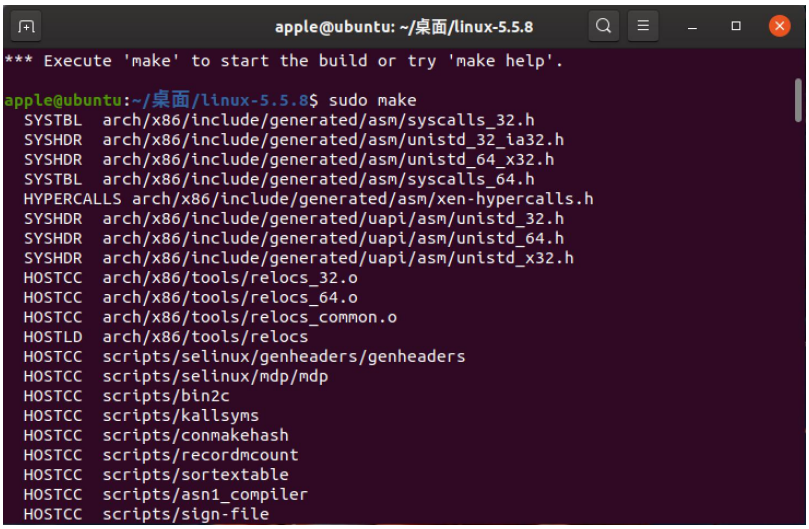
直接设置所有为默认选项，选择 Exit 后继续选择 Yes 跳转至命令行窗口。

5. 编译

执行以下指令开始编译：

```
apple@ubuntu:~/桌面/linux-5.5.8$ sudo make
```

编译过程如下图：



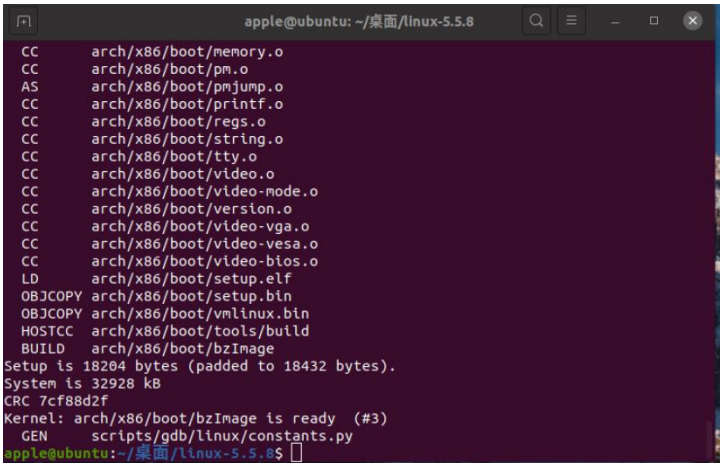
注意：编译即将完成时遇到如下错误：

```
make: *** [Makefile:1078: vmlinux] 错误 137
apple@ubuntu:~/桌面/linux-5.5.8$ ^C
apple@ubuntu:~/桌面/linux-5.5.8$ free -m
               总计         已用         空闲         共享       缓冲/缓存       可用
内存:           3908          507         3186            0          214         3176
交换:            947          559          387
```

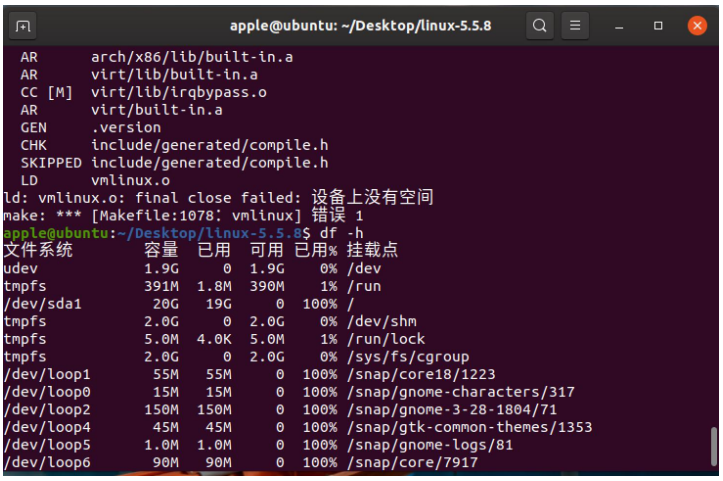
查阅相关资料后发现不是缺少 swap 交换分区（内存不够）的问题，于是尝试重新步骤 4，这次将所有可勾选的选项全部取消，问题解决。

6. 编译完成：

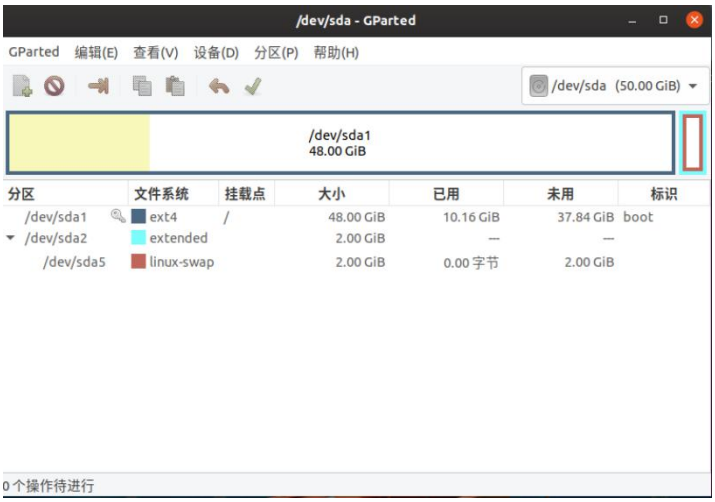
第一次完成结果如下图：



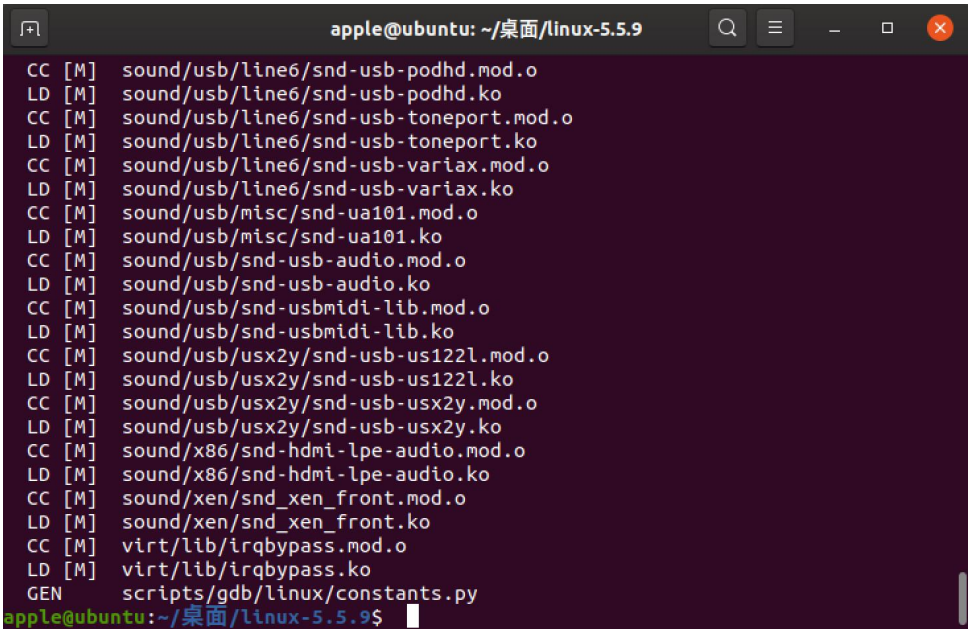
然而，在安装完 kernel 并重启 Ubuntu 并切换内核版本的时候 GRUB 崩溃，无法打开任意一版的内核。随后发现，在编译过程中有警告窗口“当前磁盘容量剩余 0KB”。



在将虚拟机磁盘分别扩展至 30GB、40GB、50GB 后仍然是这个问题。查阅资料后发现未在 Ubuntu 系统内部利用 sudo gparted 指令重新分配磁盘空间（有 30GB 未分配的磁盘空间）。



重新分配空间（48GB 虚拟机，2GBswap 空间），在 menuconfig 中将所有选项勾选并编译完成后结果如下：



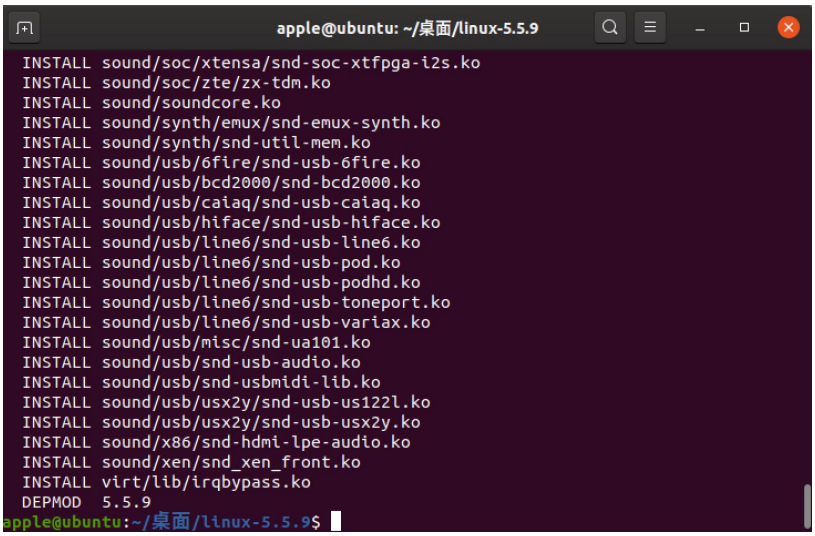
共 2 小时 20 分钟左右。(过程中最新版 kernel 已经从 5.5.8 升级到 5.5.9)。

7. 安装：

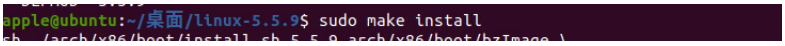
执行如下指令：

```
apple@ubuntu:~/桌面/linux-5.5.9$  sudo make modules_install
```

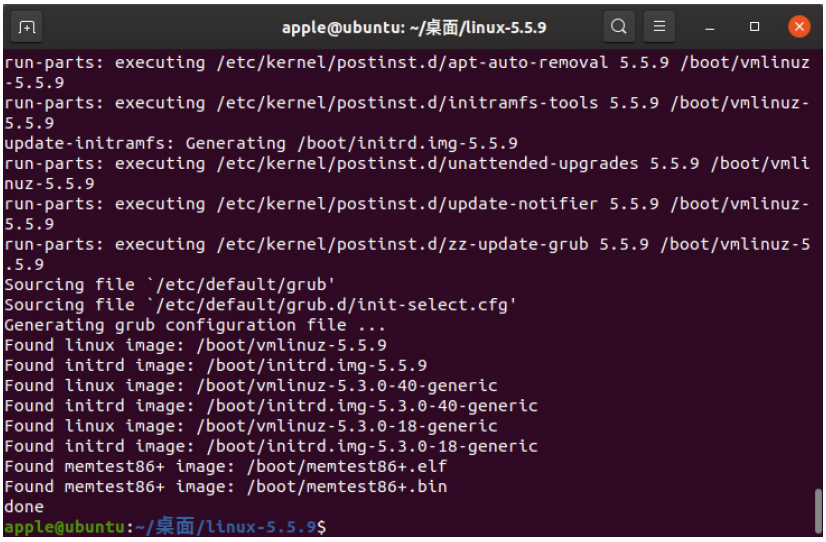
运行结果如下：



再执行命令：



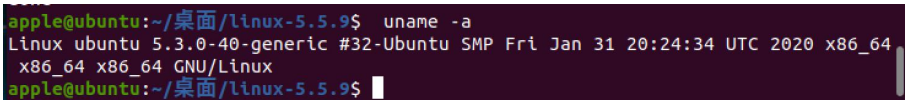
结果如下：



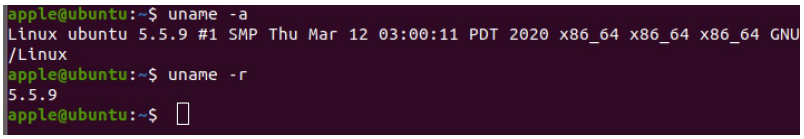
安装完成。

8. 重启验证：

在未安装最新版本 Linux Kernel 之前，Ubuntu19.10 自带内核版本为 Linux-5.3.0-40-generic，如下图：



安装完成并重启后，可以看到系统已更新到最新版本的内核 Linux-5.5.9。



四、内容二：编写 Kernel Modules

（部分过程待实机展示）

1. simple.c:

（1）代码实现：

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

static int startTime, endTime;

/* This function is called when the module is loaded. */
int simple_init(void)
{
    printk(KERN_INFO "Loading Kernel Module\n");
    // Print values of GOLDEN_RATIO_PRIME
    printk(KERN_INFO "The value GOLDEN_RATIO_PRIME  is: %lu\n", GOLDEN_RATIO_PRIME);
    // Record the start time
    startTime = jiffies;
    // Print values of jiffies and HZ
    printk(KERN_INFO "The value of jiffies when init is: %lu\n", jiffies);
    printk(KERN_INFO "The value of HZ is: %lu\n", HZ);
    return 0;
}

/* This function is called when the module is removed. */
void simple_exit(void)
{
    printk(KERN_INFO "Removing Kernel Module\n");
    // Print the result of gcd
    printk("The result of gcd is: %lu\n", gcd(3300, 24));
    // Print values of jiffies when exit
    printk("The value of jiffies when exit is: %lu\n", jiffies);
    // Record the end time
    endTime = jiffies;
    // Print the total number of seconds that have elapsed
    printk("The total elapsing time is: %lu\n", (endTime - startTime) / HZ);
}

/* 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("Chunyu Xue");
```

(2) 运行结果：

```
apple@ubuntu:~/桌面/OSProject1$ sudo dmesg -c
[sudo] apple 的密码：
apple@ubuntu:~/桌面/OSProject1$ sudo insmod simple.ko
apple@ubuntu:~/桌面/OSProject1$ dmesg
[ 6506.619578] Loading Kernel Module...
[ 6506.619579] The value GOLDEN_RATIO_PRIME  is: 7046029254386353131
[ 6506.619580] The value of jiffies when init is: 4296518706
[ 6506.619580] The value of HZ is: 250
apple@ubuntu:~/桌面/OSProject1$ sudo rmmod simple
apple@ubuntu:~/桌面/OSProject1$ dmesg
[ 6506.619578] Loading Kernel Module...
[ 6506.619579] The value GOLDEN_RATIO_PRIME  is: 7046029254386353131
[ 6506.619580] The value of jiffies when init is: 4296518706
[ 6506.619580] The value of HZ is: 250
[ 6518.121888] Removing Kernel Module...
[ 6518.121890] The result of gcd is: 12
[ 6518.121891] The value of jiffies when exit is: 4296521581
[ 6518.121891] The total elapsing time is: 11
apple@ubuntu:~/桌面/OSProject1$
```

2. jiffies、seconds：

（1）代码实现：

jiffies：

```c
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/proc_fs.h>
#include <asm/uaccess.h>
#include <linux/jiffies.h>
#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)
{
        /* Creadte the /proc/jiffies entry */
        proc_create(PROC_NAME, 0666, NULL, &proc_ops);
        printk(KERN_INFO "Successfully create /proc file...\n", PROC_NAME);

        return 0;
}

/* This function is called when the module is removed. */
void proc_exit(void)
{
        /* Remove the /proc/hello entry */
        remove_proc_entry(PROC_NAME, NULL);
        printk(KERN_INFO "Successfully remove /proc file...\n", PROC_NAME);
}

/* This function is called each time /proc/hello is read */
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;
```

```
46                  return 0;
47          }
48
49          completed = 1;
50
51          rv = sprintf(buffer, "The value of jiffies is: %lu\n", jiffies);
52
53          /* Copies kernel space buffer to user space usr_buf */
54          copy_to_user(usr_buf, buffer, rv);
55
56          return rv;
57  }
58
59  module_init(proc_init);
60  module_exit(proc_exit);
61
62  MODULE_LICENSE("GPL");
63  MODULE_DESCRIPTION("Hello Module");
64  MODULE_AUTHOR("Chunyu Xue");
65
66
```


seconds：

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

#define BUFFER_SIZE 128
#define PROC_NAME "seconds"

static long unsigned int startTime;

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)
{
        // Record the start time
        startTime = jiffies;

        /* Creadte the /proc/jiffies entry */
        proc_create(PROC_NAME, 0666, NULL, &proc_ops);
        printk(KERN_INFO "Successfully create /proc file...\n", PROC_NAME);

        return 0;
}

/* This function is called when the module is removed. */
void proc_exit(void)
{
        /* Remove the /proc/hello entry */
        remove_proc_entry(PROC_NAME, NULL);
        printk(KERN_INFO "Successfully remove /proc file...\n", PROC_NAME);
}

/* This function is called each time /proc/hello is read */
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, "The running time is: %lu\n", (jiffies - startTime) / HZ);

        /* Copies kernel space buffer to user space usr_buf */
        copy_to_user(usr_buf, buffer, rv);

        return rv;
}

module_init(proc_init);
module_exit(proc_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Hello Module");
MODULE_AUTHOR("Chunyu Xue");

```

（2）运行结果：

```
apple@ubuntu:~/桌面/OSProject1$ sudo insmod jiffies.ko
apple@ubuntu:~/桌面/OSProject1$ sudo insmod seconds.ko
apple@ubuntu:~/桌面/OSProject1$ dmesg
[ 6586.054207] Successfully create /proc file...
[ 6594.655264] Successfully create /proc file...
apple@ubuntu:~/桌面/OSProject1$ cat /proc/jiffies
The value of jiffies is: 4296544101
apple@ubuntu:~/桌面/OSProject1$ cat /proc/seconds
The running time is: 20
apple@ubuntu:~/桌面/OSProject1$ sudo rmmod jiffies
apple@ubuntu:~/桌面/OSProject1$ sudo rmmod seconds
apple@ubuntu:~/桌面/OSProject1$ dmesg
[ 6586.054207] Successfully create /proc file...
[ 6594.655264] Successfully create /proc file...
[ 6627.898718] Successfully remove /proc file...
[ 6634.718404] Successfully remove /proc file...
apple@ubuntu:~/桌面/OSProject1$
```

3. Makefile：

```
1  obj-m += simple.o jiffies.o seconds.o
2
3  all :
4          make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
5
6  clean:
7          make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
8
```

五、实验结果

在解决遇到的一系列问题，经过两天的不断尝试之后，终于顺利在 Ubuntu19.10 上编译并安装了最新版的 Linux-5.5.9 内核。

按照指示编写 simple.c、jiffies.c、seconds.c 等三个 c 文件，以及一个 Makefile，运行后能够成功完成相应的任务。

六、实验反思

1. 遇到问题时多利用网络资源（如 CSDN、博客园等网站）；
2. 就算无法直接查询到结果，也可借鉴别人类似的方法多加尝试；
3. 在解决问题时要有耐心。