Project 1

Name: 刘　涵　之
ID：519021910102

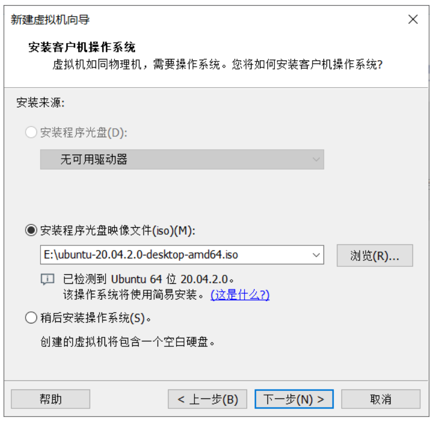
Project 1: Introduction to Linux Kernel Modules

1 Prepare Experiment Environment

In this section, I use VMware WorkStation to install the Ubuntu operating system. Then, I compile and upgrade the linux kernel to the latest version (5.11.3).

1.1 VMware WorkStation and Ubuntu OS

I install the **VMware Workstation 16 Pro** and download the image of Ubuntu OS (**ubuntu-20.04.2.0-desktop-amd64.iso**). I create a new virtual machine and choose this image to install.



新建虚拟机向导

安装客户机操作系统
虚拟机如同物理机，需要操作系统。您将如何安装客户机操作系统?

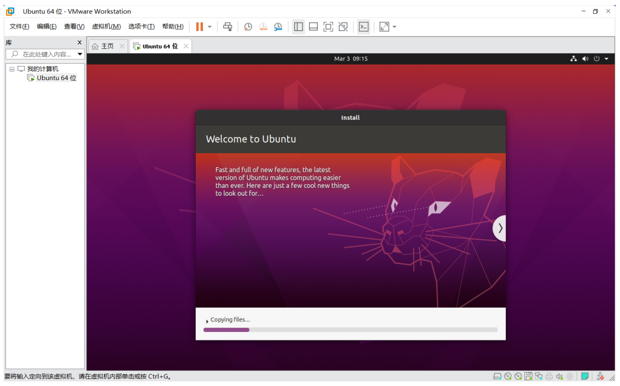
安装来源:

( ) 安装程序光盘(D):
无可用驱动器

[✓ 安装程序光盘映像文件(iso)(M):]
E:\ubuntu-20.04.2.0-desktop-amd64.iso
已检测到 Ubuntu 64 位 20.04.2.0。
该操作系统将使用简易安装。(这是什么?)

( ) 稍后安装操作系统(S)。
创建的虚拟机将包含一个空白硬盘。

图 1: VMware WorkStation



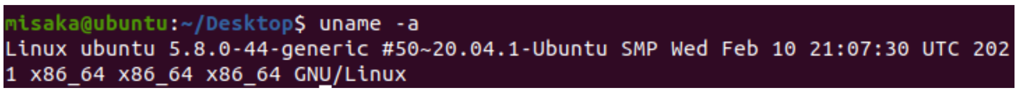
Fast and full of new features, the latest version of Ubuntu makes computing easier than ever. Here are just a few cool new things to look out for...

Copying files...

图 2: Install Ubuntu OS

Then I check the version of linux kernel, using the following instruction in bash.

```
uname -a
```



```
misaka@ubuntu:~/Desktop$ uname -a
Linux ubuntu 5.8.0-44-generic #50~20.04.1-Ubuntu SMP Wed Feb 10 21:07:30 UTC 202
1 x86_64 x86_64 x86_64 GNU/Linux
```

图 3: Check the Kernel Version

In Figure 3, it shows that the current kernel version is **5.8.0**.

1.2 Change the Package Source

To download packages more faster, i change the source of the system package manager to SJTUG source. I follow the document in https://mirror.sjtu.edu.cn/docs/ubuntu and using the following instructions.

```
sudo gedit /etc/apt/sources.list
```

```
deb https://mirrors.sjtug.sjtu.edu.cn/ubuntu focal main restricted
deb https://mirrors.sjtug.sjtu.edu.cn/ubuntu focal-updates main restricted
deb https://mirrors.sjtug.sjtu.edu.cn/ubuntu focal universe
deb https://mirrors.sjtug.sjtu.edu.cn/ubuntu focal-updates universe
deb https://mirrors.sjtug.sjtu.edu.cn/ubuntu focal multiverse
deb https://mirrors.sjtug.sjtu.edu.cn/ubuntu focal-updates multiverse
deb https://mirrors.sjtug.sjtu.edu.cn/ubuntu focal-backports main restricted universe
    multiverse
deb http://archive.canonical.com/ubuntu focal partner
deb https://mirrors.sjtug.sjtu.edu.cn/ubuntu focal-security main restricted universe
    multiverse
```

```
sudo apt update
sudo apt upgrade
```

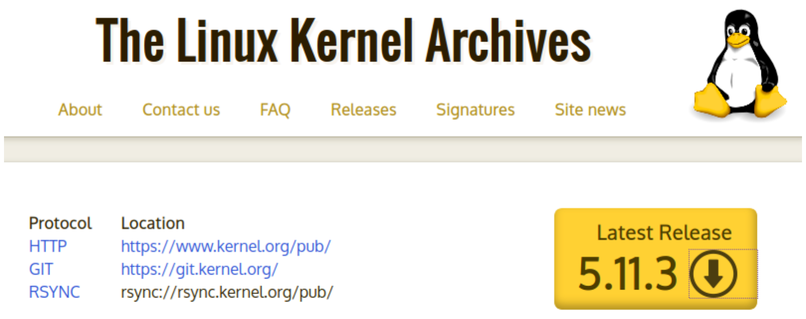
To prepare for compiling, i need to install some packages essential for building.

```
sudo apt-get install kernel-package git fakeroot build-essential
sudo apt-get install ncurses-dev xz-utils libssl-dev bc flex bison
```

1.3 Compile and Upgrade Linux Kernel (Recommended)

To understand how kernel is installed, I decide to upgrade my kernel.

I find the latest release of linux kernel on www.kernel.org



Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

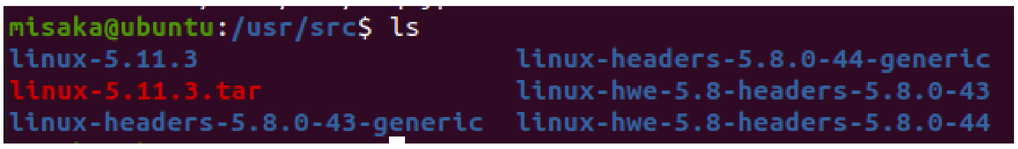
图 4: the Latest Kernel Version

I download the **linux-5.11.3.tar.xz** and then move it to **/usr/src**

```
cp linux-5.11.3.tar.xz /usr/src
```

Then i unzip the file using the following instructions.

```
sudo xz -d linux-5.11.3.tar.xz
sudo tar xvf linux-5.11.3.tar
```

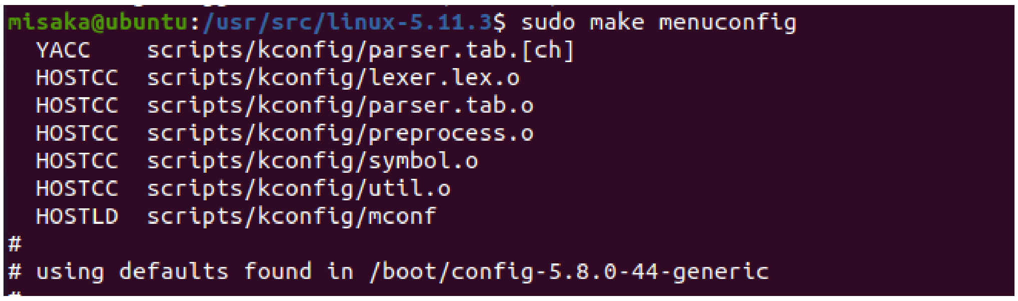


```
misaka@ubuntu:/usr/src$ ls
linux-5.11.3                      linux-headers-5.8.0-44-generic
linux-5.11.3.tar                  linux-hwe-5.8-headers-5.8.0-43
linux-headers-5.8.0-43-generic    linux-hwe-5.8-headers-5.8.0-44
```

图 5: File in /usr/src

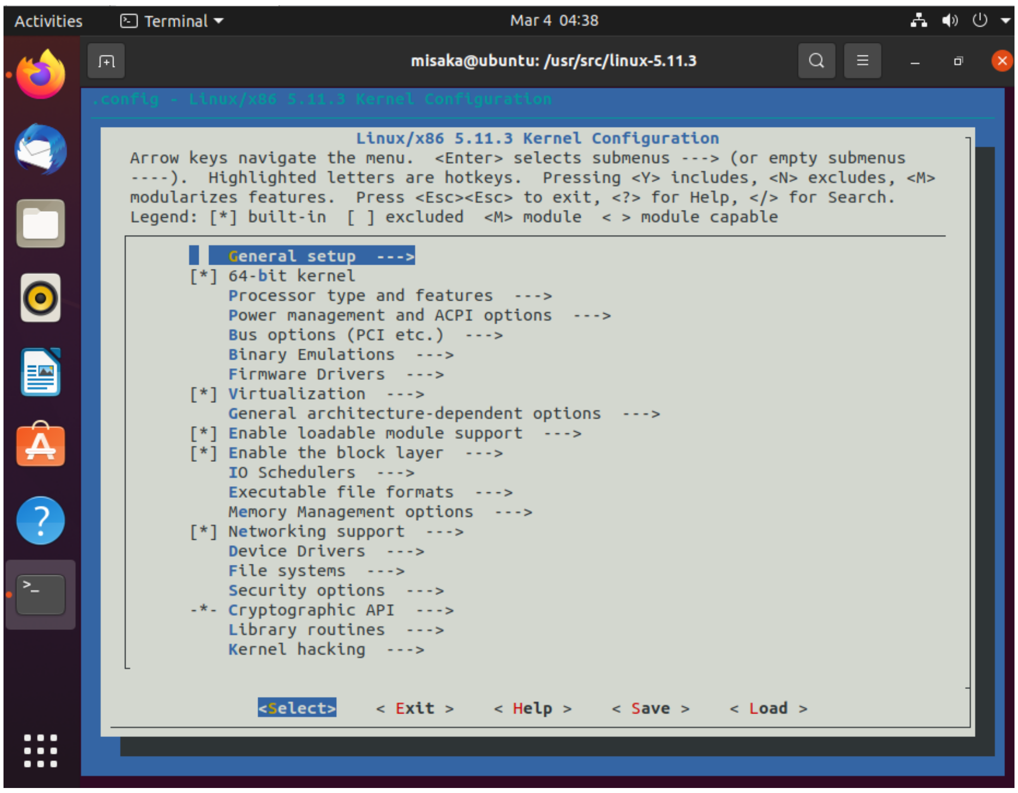
Then i use the following instruction to set the configurations, the menu is displayed as Figure 7. I use the default configurations.

```
sudo make menuconfig
```



```
misaka@ubuntu:/usr/src/linux-5.11.3$ sudo make menuconfig
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTCC  scripts/kconfig/util.o
  HOSTLD  scripts/kconfig/mconf
#
# using defaults found in /boot/config-5.8.0-44-generic
```

图 6: sudo make menuconfig



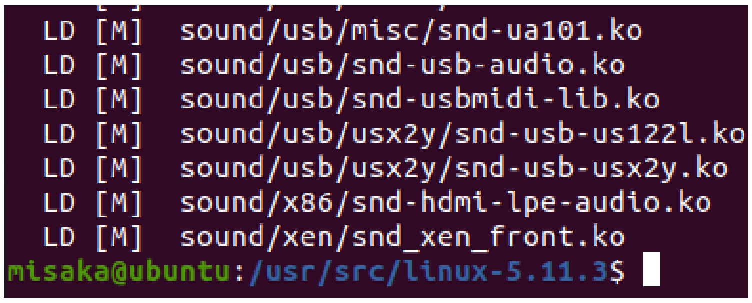
```
.config - Linux/x86 5.11.3 Kernel Configuration
                    Linux/x86 5.11.3 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        General setup  --->
    [*] 64-bit kernel
        Processor type and features  --->
        Power management and ACPI options  --->
        Bus options (PCI etc.)  --->
        Binary Emulations  --->
        Firmware Drivers  --->
    [*] Virtualization  --->
        General architecture-dependent options  --->
    [*] Enable loadable module support  --->
    [*] Enable the block layer  --->
        IO Schedulers  --->
        Executable file formats  --->
        Memory Management options  --->
    [*] Networking support  --->
        Device Drivers  --->
        File systems  --->
        Security options  --->
    -*- Cryptographic API  --->
        Library routines  --->
        Kernel hacking  --->

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

图 7: Menu of configurations

Finally, i run the following instruction to compile the kernel with 8 threads compile in parallel.

```
sudo make -j8
```

After 30 minutes, the compilation is complete.

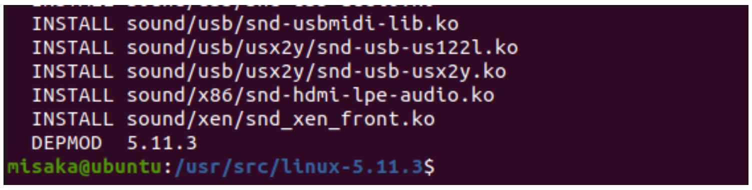


```
  LD [M]  sound/usb/misc/snd-ua101.ko
  LD [M]  sound/usb/snd-usb-audio.ko
  LD [M]  sound/usb/snd-usbmidi-lib.ko
  LD [M]  sound/usb/usx2y/snd-usb-us122l.ko
  LD [M]  sound/usb/usx2y/snd-usb-usx2y.ko
  LD [M]  sound/x86/snd-hdmi-lpe-audio.ko
  LD [M]  sound/xen/snd_xen_front.ko
misaka@ubuntu:/usr/src/linux-5.11.3$
```

图 8: sudo make -j8

I run the following instruction to install modules.

```
sudo make modules_install
```

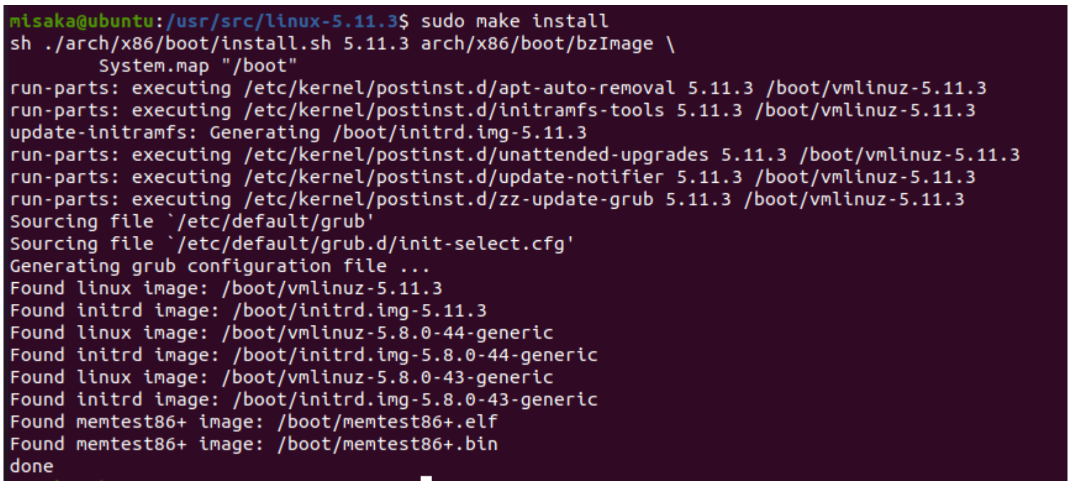


```
  INSTALL sound/usb/snd-usbmidi-lib.ko
  INSTALL sound/usb/usx2y/snd-usb-us122l.ko
  INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
  INSTALL sound/x86/snd-hdmi-lpe-audio.ko
  INSTALL sound/xen/snd_xen_front.ko
  DEPMOD  5.11.3
misaka@ubuntu:/usr/src/linux-5.11.3$
```

图 9: sudo make modules_install

Then i can install the new kernel!

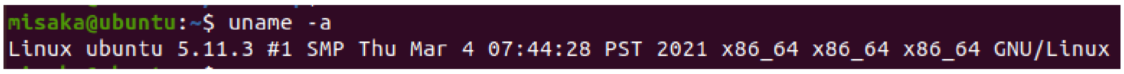
```
sudo make install
```



```
misaka@ubuntu:/usr/src/linux-5.11.3$ sudo make install
sh ./arch/x86/boot/install.sh 5.11.3 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 5.11.3 /boot/vmlinuz-5.11.3
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.11.3 /boot/vmlinuz-5.11.3
update-initramfs: Generating /boot/initrd.img-5.11.3
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.11.3 /boot/vmlinuz-5.11.3
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.11.3 /boot/vmlinuz-5.11.3
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.11.3 /boot/vmlinuz-5.11.3
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.11.3
Found initrd image: /boot/initrd.img-5.11.3
Found linux image: /boot/vmlinuz-5.8.0-44-generic
Found initrd image: /boot/initrd.img-5.8.0-44-generic
Found linux image: /boot/vmlinuz-5.8.0-43-generic
Found initrd image: /boot/initrd.img-5.8.0-43-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

图 10: Install the new kernel

Then i reboot the system and check the kernel version again. It shows that the new kernel (5.11.3) is installed successfully.



```
misaka@ubuntu:~$ uname -a
Linux ubuntu 5.11.3 #1 SMP Thu Mar 4 07:44:28 PST 2021 x86_64 x86_64 x86_64 GNU/Linux
```

图 11: Check the new kernel

2 Kernel Modules Overview

In this section i will talk about the programming project at the end of Chapter 2.

2.1 Task 1 - Simple Module

I use the following commands to make and install my module (for Task 2 3, all most the same).

```
sudo make
sudo dmesg -C
sudo insmod simple.ko
dmesg
sudo rmmod simple
dmesg
```

In the Simple module, we are required to print the **Golden Ratio Prime** when initializing (simple_init()) and print the **greatest common divisor** of 3300 and 24 when exiting (simple_exit()).

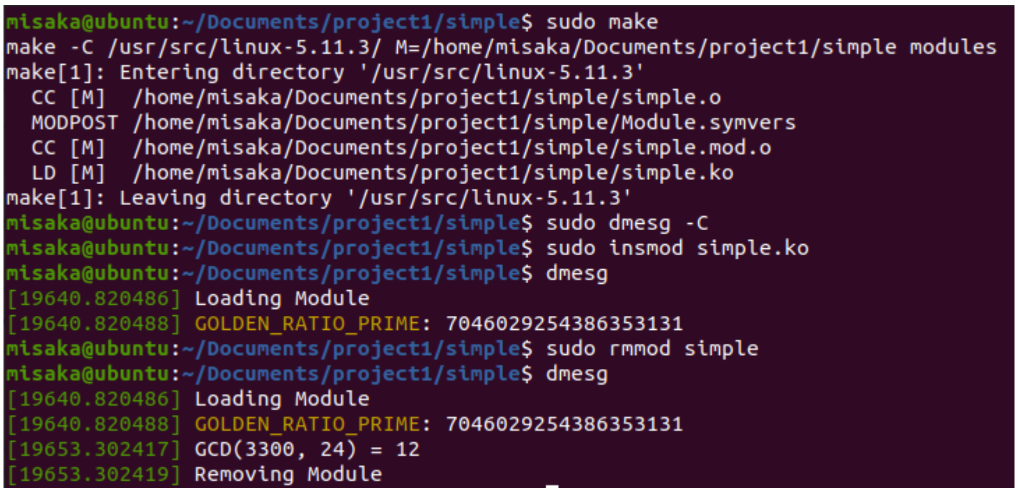
As written on the textbook, i find the constant GOLDEN_RATIO_PRIME in the <linux/hash.h> and gcd() in the <linux/hash.h>. The code of simple.c is displayed as follows.

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

int simple_init(void)
{
       printk(KERN_INFO "Loading Module\n");
       printk(KERN_INFO "GOLDEN_RATIO_PRIME: %llu\n", GOLDEN_RATIO_PRIME);
       return 0;
}

void simple_exit(void) {
        printk(KERN_INFO "GCD(3300, 24) = %lu\n", gcd(3300,24));
        printk(KERN_INFO "Removing Module\n");
}

module_init( simple_init );
module_exit( simple_exit );

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple Module");
MODULE_AUTHOR("MisakaCenter");
```

For this task, the **Makefile** is written as follows.

```
obj-m := simple.o
all:
        make -C /usr/src/linux-5.11.3/ M=$(shell pwd) modules
clean:
        make -C /usr/src/linux-5.11.3/ M=$(shell pwd) clean
```

The result is shown as follows.



```
misaka@ubuntu:~/Documents/project1/simple$ sudo make
make -C /usr/src/linux-5.11.3/ M=/home/misaka/Documents/project1/simple modules
make[1]: Entering directory '/usr/src/linux-5.11.3'
  CC [M]  /home/misaka/Documents/project1/simple/simple.o
  MODPOST /home/misaka/Documents/project1/simple/Module.symvers
  CC [M]  /home/misaka/Documents/project1/simple/simple.mod.o
  LD [M]  /home/misaka/Documents/project1/simple/simple.ko
make[1]: Leaving directory '/usr/src/linux-5.11.3'
misaka@ubuntu:~/Documents/project1/simple$ sudo dmesg -C
misaka@ubuntu:~/Documents/project1/simple$ sudo insmod simple.ko
misaka@ubuntu:~/Documents/project1/simple$ dmesg
[19640.820486] Loading Module
[19640.820488] GOLDEN_RATIO_PRIME: 7046029254386353131
misaka@ubuntu:~/Documents/project1/simple$ sudo rmmod simple
misaka@ubuntu:~/Documents/project1/simple$ dmesg
[19640.820486] Loading Module
[19640.820488] GOLDEN_RATIO_PRIME: 7046029254386353131
[19653.302417] GCD(3300, 24) = 12
[19653.302419] Removing Module
```

图 12: simple.ko

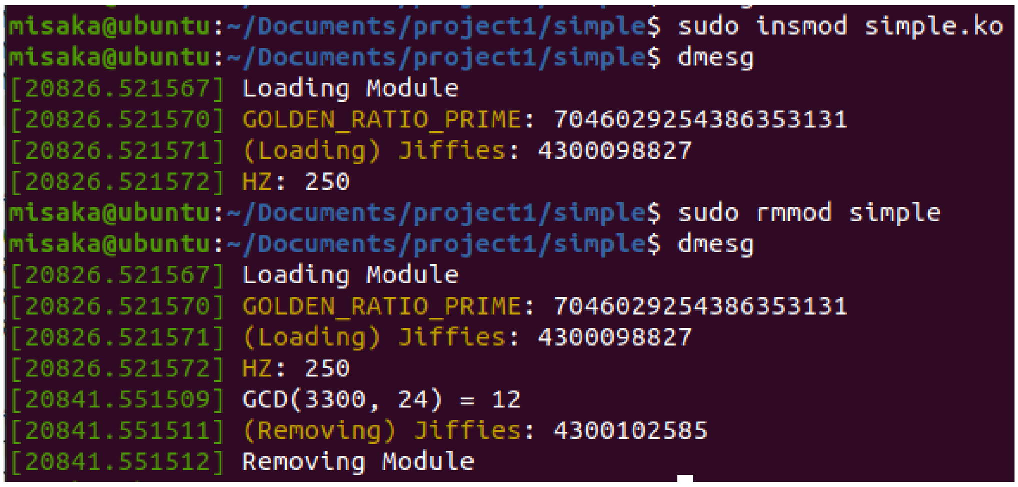
More over, for preparation of the Task 2, i add the following code to the simple_init.

```
printk(KERN_INFO "(Loading) Jiffies: %lu\n", jiffies);
printk(KERN_INFO "HZ: %d\n", HZ);
```

and i add the following code to the simple_exit.

```
printk(KERN_INFO "(Removing) Jiffies: %lu\n", jiffies);
```

The result is shown as follows.



```
misaka@ubuntu:~/Documents/project1/simple$ sudo insmod simple.ko
misaka@ubuntu:~/Documents/project1/simple$ dmesg
[20826.521567] Loading Module
[20826.521570] GOLDEN_RATIO_PRIME: 7046029254386353131
[20826.521571] (Loading) Jiffies: 4300098827
[20826.521572] HZ: 250
misaka@ubuntu:~/Documents/project1/simple$ sudo rmmod simple
misaka@ubuntu:~/Documents/project1/simple$ dmesg
[20826.521567] Loading Module
[20826.521570] GOLDEN_RATIO_PRIME: 7046029254386353131
[20826.521571] (Loading) Jiffies: 4300098827
[20826.521572] HZ: 250
[20841.551509] GCD(3300, 24) = 12
[20841.551511] (Removing) Jiffies: 4300102585
[20841.551512] Removing Module
```

图 13: Print jiffies and HZ

2.2 Task 2 - Jiffies Module

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

As same as Task 1, jiffies can be get in <linux/jiffies.h>, and its type is unsigned long volatile. We can use %lu to format its output. The code of jiffies.c is display as follows.

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

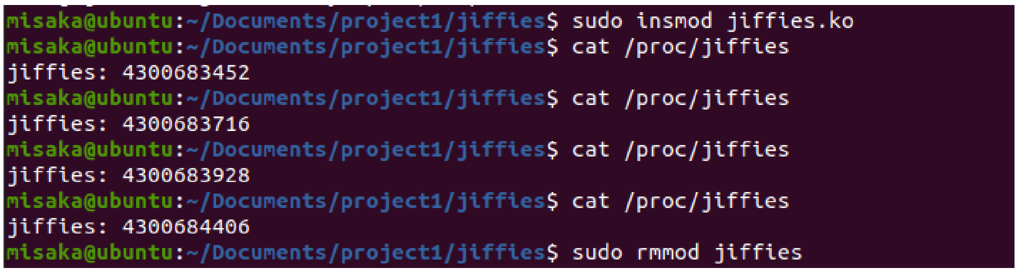
```c
#include <asm/uaccess.h>
#include <linux/jiffies.h>

#define BUFFER_SIZE 128

#define PROC_NAME "jiffies"

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

static struct proc_ops proc_ops = {
        .proc_read = proc_read
};

int proc_init(void)
{
        proc_create(PROC_NAME, 0, 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, "jiffies: %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("Jiffies Module");
MODULE_AUTHOR("MisakaCenter");
```

For this task, the **Makefile** is written as follows.

```makefile
obj-m := jiffies.o
all:
        make -C /usr/src/linux-5.11.3/ M=$(shell pwd) modules
```

```
clean:
	make -C /usr/src/linux-5.11.3/ M=$(shell pwd) clean
```

The result is shown as follows.



```
misaka@ubuntu:~/Documents/project1/jiffies$ sudo insmod jiffies.ko
misaka@ubuntu:~/Documents/project1/jiffies$ cat /proc/jiffies
jiffies: 4300683452
misaka@ubuntu:~/Documents/project1/jiffies$ cat /proc/jiffies
jiffies: 4300683716
misaka@ubuntu:~/Documents/project1/jiffies$ cat /proc/jiffies
jiffies: 4300683928
misaka@ubuntu:~/Documents/project1/jiffies$ cat /proc/jiffies
jiffies: 4300684406
misaka@ubuntu:~/Documents/project1/jiffies$ sudo rmmod jiffies
```

图 14: Jiffies Module

2.3 Task 3 - Seconds Module

Design a kernel module that creates a proc file named /proc/seconds that reports the number of elapsed seconds since the kernel module was loaded. This will involve using the value of jiffies as well as the HZ rate.

To solve this task:

- When initializing, record the current jiffies in the global variable **jiffies_load**
- jiffies / HZ turns out to be seconds
- when /proc/seconds is read, (jiffies−jiffies_load)/HZ is the seconds we need.

The code of seconds.c is display as follows.

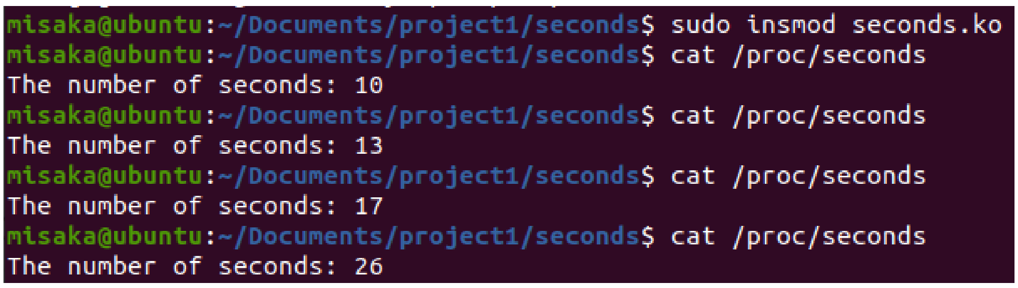
```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <asm/uaccess.h>
#include <linux/jiffies.h>
#include <asm/param.h>

#define BUFFER_SIZE 128

#define PROC_NAME "seconds"

unsigned long jiffies_load;

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

static struct proc_ops proc_ops = {
	.proc_read = proc_read
};

int proc_init(void)
{
	proc_create(PROC_NAME, 0, NULL, &proc_ops);
	printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
	jiffies_load = jiffies;
```

```c
        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;
        unsigned long seconds_since_load = (jiffies - jiffies_load )/ HZ;
        if (completed) {
                completed = 0;
                return 0;
        }
        completed = 1;
        rv = sprintf(buffer, "The number of seconds: %lu\n", seconds_since_load);
        copy_to_user(usr_buf, buffer, rv);
        return rv;
}

module_init( proc_init );
module_exit( proc_exit );

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Seconds Module");
MODULE_AUTHOR("MisakaCenter");
```

For this task, the **Makefile** is written as follows.

```makefile
obj-m := seconds.o
all:
        make -C /usr/src/linux-5.11.3/ M=$(shell pwd) modules
clean:
        make -C /usr/src/linux-5.11.3/ M=$(shell pwd) clean
```

The result is shown as follows.



```
misaka@ubuntu:~/Documents/project1/seconds$ sudo insmod seconds.ko
misaka@ubuntu:~/Documents/project1/seconds$ cat /proc/seconds
The number of seconds: 10
misaka@ubuntu:~/Documents/project1/seconds$ cat /proc/seconds
The number of seconds: 13
misaka@ubuntu:~/Documents/project1/seconds$ cat /proc/seconds
The number of seconds: 17
misaka@ubuntu:~/Documents/project1/seconds$ cat /proc/seconds
The number of seconds: 26
```

图 15: Seconds Module