CS5250 ASSIGNMENT 3

YANG MO – A0091836X

Task 1

1a – When will module_init and module_exit be loading/called?

`module_init` will be called when the module is loaded/installed into the kernel.

`module_exit` will be called when the module is removed from the kernel.

1b - Command of building, installing and removing the module

- To build the module, we need a "`Makefile`" which has the following commands(if the module file is named `HelloWorld.c`):

```
obj-m += HelloWorld.o

all:

        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:

        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

- To install the module, we need the following command:

```
sudo insmod HelloWorld.ko
```

- To remove the module, we need the following command:

```
sudo rmmod HelloWorld
```

1c - screenshots and results

- Build with Makefile

`Makefile:`

```
yangmo@yangmo-ubuntu:~/mymodules/task1$ ls
HelloWorld.c  Makefile
yangmo@yangmo-ubuntu:~/mymodules/task1$ cat Makefile
obj-m += HelloWorld.o

all:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
        make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Run "`make`":

```
yangmo@yangmo-ubuntu:~/mymodules/task1$ make
make -C /lib/modules/4.6.3/build M=/home/yangmo/mymodules/task1 modules
make[1]: Entering directory '/home/yangmo/linux-4.6.3'
  CC [M]  /home/yangmo/mymodules/task1/HelloWorld.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/yangmo/mymodules/task1/HelloWorld.mod.o
  LD [M]  /home/yangmo/mymodules/task1/HelloWorld.ko
make[1]: Leaving directory '/home/yangmo/linux-4.6.3'
```

- Install the module

```
yangmo@yangmo-ubuntu:~/mymodules/task1$ sudo insmod HelloWorld.ko
[sudo] password for yangmo:
yangmo@yangmo-ubuntu:~/mymodules/task1$
```

Check the message

```
yangmo@yangmo-ubuntu:~/mymodules/task1$ dmesg | tail
[   23.945435] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   23.947007] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   23.947413] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[   23.947668] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   48.407108] ISO 9660 Extensions: Microsoft Joliet Level 3
[   48.702664] ISO 9660 Extensions: RRIP_1991A
[  109.428492] do_trap: 33 callbacks suppressed
[  109.428495] traps: pool[2409] trap int3 ip:7f086096f9eb sp:7f083da396f0 error
:0 in libglib-2.0.so.0.4800.0[7f086091f000+10e000]
[  176.155294] Hello, world
```

- Remove the module

```
yangmo@yangmo-ubuntu:~/mymodules/task1$ sudo rmmod HelloWorld
yangmo@yangmo-ubuntu:~/mymodules/task1$
```

Check the message

```
yangmo@yangmo-ubuntu:~/mymodules/task1$ dmesg | tail
[   23.945435] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   23.947007] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[   23.947413] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[   23.947668] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   48.407108] ISO 9660 Extensions: Microsoft Joliet Level 3
[   48.702664] ISO 9660 Extensions: RRIP_1991A
[  109.428492] do_trap: 33 callbacks suppressed
[  109.428495] traps: pool[2409] trap int3 ip:7f086096f9eb sp:7f083da396f0 error
:0 in libglib-2.0.so.0.4800.0[7f086091f000+10e000]
[  176.155294] Hello, world
[  183.996237] Goodbye, cruel world
```

1d – Add <who> parameter

Added/Changed code:

HelloWorld.c (~/mymodules/task1) - gedit

Open

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

MODULE_LICENSE("GPL");

static char *who = "unknown";

module_param(who, charp, 0000);
MODULE_PARM_DESC(who, "char string of parameter who");

static int hello_init(void)
{
        printk(KERN_ALERT "Hello, %s\n", who);
        return 0;
}

static void hello_exit(void)
{
        printk(KERN_ALERT "Goodbye, cruel world\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

Make again:

```
yangmo@yangmo-ubuntu:~/mymodules/task1$ make
make -C /lib/modules/4.6.3/build M=/home/yangmo/mymodules/task1 modules
make[1]: Entering directory '/home/yangmo/linux-4.6.3'
  CC [M]  /home/yangmo/mymodules/task1/HelloWorld.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/yangmo/mymodules/task1/HelloWorld.mod.o
  LD [M]  /home/yangmo/mymodules/task1/HelloWorld.ko
make[1]: Leaving directory '/home/yangmo/linux-4.6.3'
```

Load module with the new parameter <who> and give it value → `'"YANG MO"'` (The reason why to single quote than double quote the name `YANG MO` is that shell will consume the quotes and if only quote once, the string `YANG MO` will be treated separately and giving error). After loading, remove the module and check the message:

```
yangmo@yangmo-ubuntu:~/mymodules/task1$ sudo insmod HelloWorld.ko who='"YANG MO"
'
yangmo@yangmo-ubuntu:~/mymodules/task1$ sudo rmmod HelloWorld
yangmo@yangmo-ubuntu:~/mymodules/task1$ dmesg | tail
[  176.155294] Hello, world
[  183.996237] Goodbye, cruel world
[  740.053385] HelloWorld: unknown parameter 'MO' ignored
[  740.053441] Hello, YANG
[  759.890364] Goodbye, cruel world
[  772.900923] HelloWorld: unknown parameter 'YANG MO' ignored
[  772.900966] Hello, unknown
[  785.782042] Goodbye, cruel world
[  898.766892] Hello, YANG MO
[  913.365200] Goodbye, cruel world
yangmo@yangmo-ubuntu:~/mymodules/task1$
```

Task 2

2a – mknod commad

```
yangmo@yangmo-ubuntu: ~
yangmo@yangmo-ubuntu:~$ sudo mknod /dev/one c 61 1
[sudo] password for yangmo:
yangmo@yangmo-ubuntu:~$
```

This command will create a new char device (there is a "c" in the command) named "one" and with major number 61 (for driver) and minor number 1(for device).

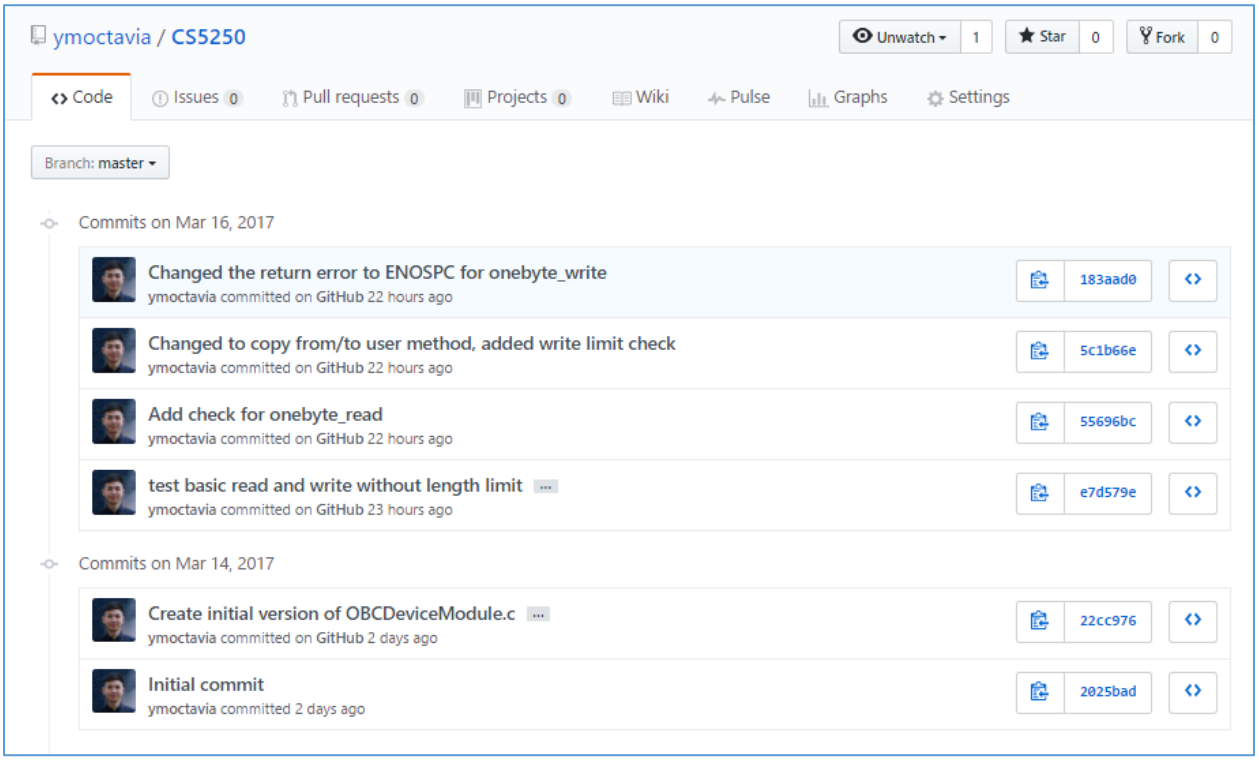
2b – Screenshot of device

```
brw-rw----  1 root     disk      7,   0 Mar 16 23:41 loop0
brw-rw----  1 root     disk      7,   1 Mar 16 23:41 loop1
brw-rw----  1 root     disk      7,   2 Mar 16 23:41 loop2
brw-rw----  1 root     disk      7,   3 Mar 16 23:41 loop3
brw-rw----  1 root     disk      7,   4 Mar 16 23:41 loop4
brw-rw----  1 root     disk      7,   5 Mar 16 23:41 loop5
brw-rw----  1 root     disk      7,   6 Mar 16 23:41 loop6
brw-rw----  1 root     disk      7,   7 Mar 16 23:41 loop7
crw-rw----  1 root     disk     10, 237 Mar 16 23:41 loop-control
drwxr-xr-x  2 root     root          60 Mar 16 23:41 mapper
crw-------  1 root     root     10, 227 Mar 16 23:41 mcelog
crw-r-----  1 root     kmem      1,   1 Mar 16 23:41 mem
crw-------  1 root     root     10,  56 Mar 16 23:41 memory_bandwidth
drwxrwxrwt  2 root     root          40 Mar 16 23:41 mqueue
drwxr-xr-x  2 root     root          60 Mar 16 23:41 net
crw-------  1 root     root     10,  58 Mar 16 23:41 network_latency
crw-------  1 root     root     10,  57 Mar 16 23:41 network_throughput
crw-rw-rw-  1 root     root      1,   3 Mar 16 23:41 null
crw-r--r--  1 root     root     61,   1 Mar 16 23:44 one
crw-r-----  1 root     kmem      1,   4 Mar 16 23:41 port
crw-------  1 root     root    108,   0 Mar 16 23:41 ppp
crw-------  1 root     root     10,   1 Mar 16 23:41 psaux
crw-rw-rw-  1 root     tty       5,   2 Mar 16 23:47 ptmx
drwxr-xr-x  2 root     root           0 Mar 16 23:41 pts
brw-rw----  1 root     disk      1,   0 Mar 16 23:41 ram0
brw-rw----  1 root     disk      1,   1 Mar 16 23:41 ram1
```

As can be seen from above screenshot, the device one is created.

2c - Screenshot of Github, implemented functions and four testing cases

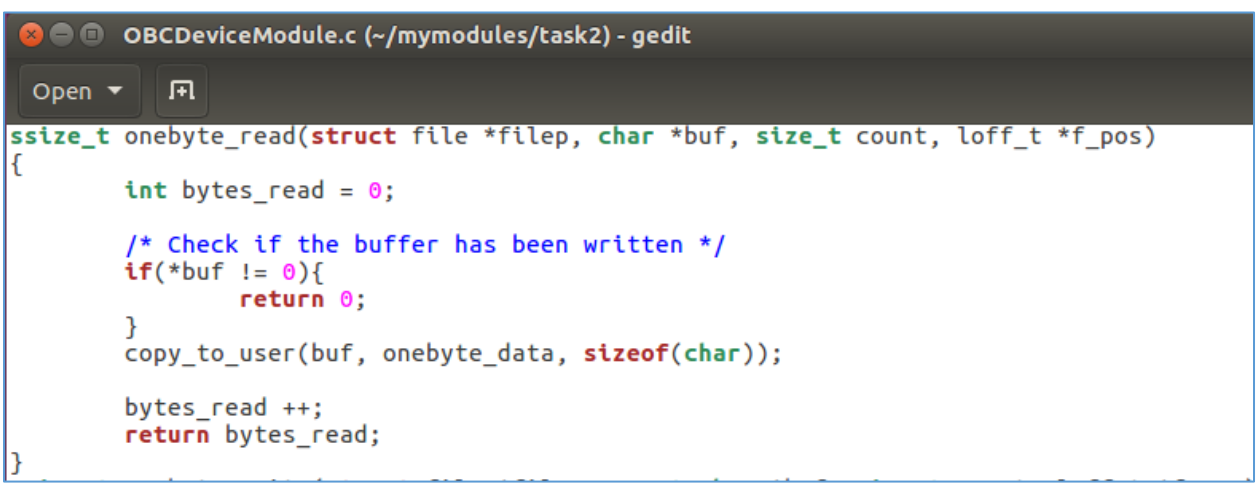
- Github Screenshot of commits:



For reference, the link to my CS5250 github repo is:

https://github.com/ymoctavia/CS5250/commits/master

- Code of read function:




```
ssize_t onebyte_read(struct file *filep, char *buf, size_t count, loff_t *f_pos)
{
        int bytes_read = 0;

        /* Check if the buffer has been written */
        if(*buf != 0){
                return 0;
        }
        copy_to_user(buf, onebyte_data, sizeof(char));

        bytes_read ++;
        return bytes_read;
}
```

When use command `cat` on the device, the read function will be kept calling until it returns a 0 to signify the end. Since there is only one byte of data, we only need to read once and return 0 if

the data has been read. The function used in this method is `copy_to_user` which will only read one byte from `onebyte_data` to `buf`.

Reference used for copy_to_user function:
https://www.fsl.cs.sunysb.edu/kernel-api/re248.html

- Code of write function:

OBCDeviceModule.c (~/mymodules/task2) - gedit

```
ssize_t onebyte_write(struct file *filep, const char *buf, size_t count, loff_t *f_pos)
{
        int bytes_write = 0;
        copy_from_user(onebyte_data, buf, sizeof(char));

        /* Check the length of the bytes that have been written*/
        if(count > sizeof(char))
        {
                printk(KERN_ALERT "No space left on device\n");

                /*Return Linux System Error<28>: No space left on device */
                return -ENOSPC;
        }

        bytes_write ++;
        return bytes_write;
}
```

  - Since only one byte of data should be written, I am using `copy_from_user` function to copy one byte of data from `buf` to `onebyte_data`.

    Reference used for copy_from_user function:
    https://www.fsl.cs.sunysb.edu/kernel-api/re249.html

  - Since when there are more than one byte of data written only the first byte will be written and error should be given to console, I am checking where count is larger than `sizeof(char)`. If it is true, then we first print a message in dmesg and then return "`-ENOSPC`" to tell that the linux system error "`No space left on device`" and print it on the console

    Referenced used for –ENOSPC :
    http://www-numi.fnal.gov/offline_software/srt_public_context/WebDocs/Errors/unix_system_errors.html

- Screenshot of four testing cases:
  Change to root first

root@yangmo-ubuntu: ~

```
yangmo@yangmo-ubuntu:~$ sudo -s
[sudo] password for yangmo:
root@yangmo-ubuntu:~#
```

Conduct the test as instructed in the task document

```
root@yangmo-ubuntu: ~
yangmo@yangmo-ubuntu:~$ sudo -s
[sudo] password for yangmo:
root@yangmo-ubuntu:~# cat /dev/one
Xroot@yangmo-ubuntu:~# printf a>/dev/one
root@yangmo-ubuntu:~# cat /dev/one
aroot@yangmo-ubuntu:~# printf b>/dev/one
root@yangmo-ubuntu:~# cat /dev/one
broot@yangmo-ubuntu:~# printf abc>/dev/one
bash: printf: write error: No space left on device
root@yangmo-ubuntu:~# cat /dev/one
aroot@yangmo-ubuntu:~#
```

As can be seen

- Test 1 gives default value ‘X’
- Test 2 writes one byte of data “a” , then read and give new value “a”
- Test 3 writes another byte of data “b”, then read and give new value “b”
- Test 4 tries to write more than one byte of data “abc”, an error “No Space left on device” is printed in the console as expected. In addition, since only first byte of data is written, when read from the device it only gives “a”.

The above test results are proved correct according to the screenshot in the task document.