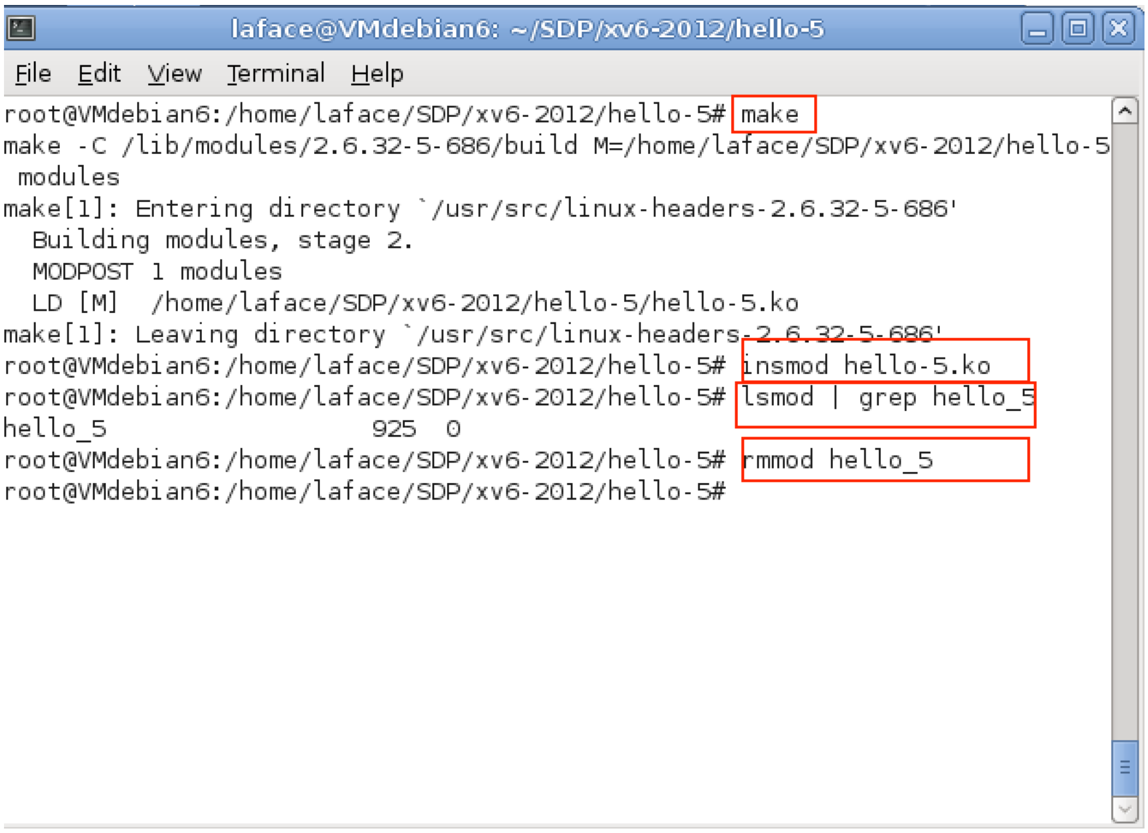
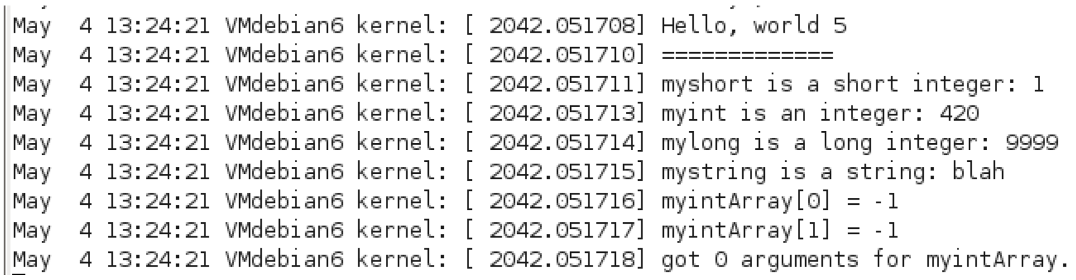
ES1

First of all i computed the command make to compile the module. This command generates a file “:ko which is the module that will go to install by using the command “insmod ModuleName.ko”and once do that by using the command “lsmod I grep ModuleName” to be sure that the module has been correctly installed.
The modules, once there have been installed, they are in this directory : “/sys/module”.
By using “tail -f /var/log/messages” ( in anther shell) i’m able to see what the module just installed prints.
When i want to remove a module i have to use this command “ rm ModuleName “.

The sequence of used command are reported below.

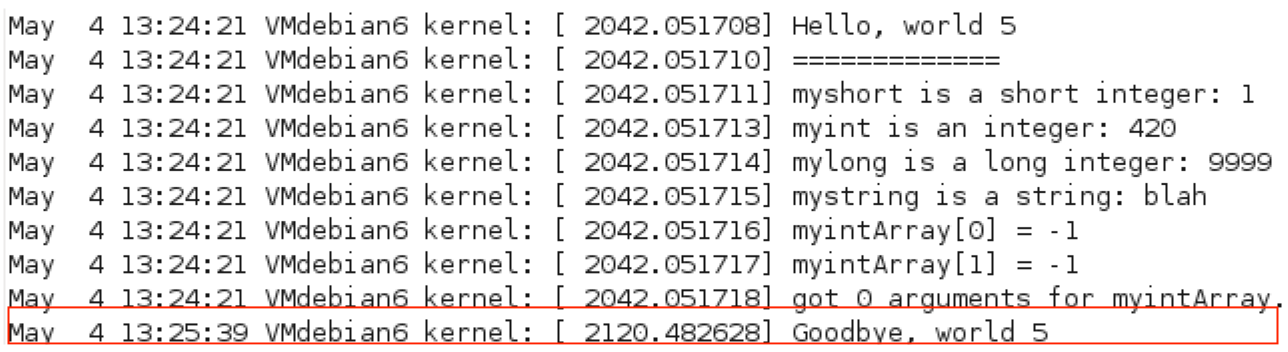

laface@VMdebian6: ~/SDP/xv6-2012/hello-5
File Edit View Terminal Help

```
root@VMdebian6:/home/laface/SDP/xv6-2012/hello-5# make
make -C /lib/modules/2.6.32-5-686/build M=/home/laface/SDP/xv6-2012/hello-5
 modules
make[1]: Entering directory `/usr/src/linux-headers-2.6.32-5-686'
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /home/laface/SDP/xv6-2012/hello-5/hello-5.ko
make[1]: Leaving directory `/usr/src/linux-headers-2.6.32-5-686'
root@VMdebian6:/home/laface/SDP/xv6-2012/hello-5# insmod hello-5.ko
root@VMdebian6:/home/laface/SDP/xv6-2012/hello-5# lsmod | grep hello_5
hello_5                  925  0
root@VMdebian6:/home/laface/SDP/xv6-2012/hello-5# rmmod hello_5
root@VMdebian6:/home/laface/SDP/xv6-2012/hello-5#
```



```
May  4 13:24:21 VMdebian6 kernel: [ 2042.051708] Hello, world 5
May  4 13:24:21 VMdebian6 kernel: [ 2042.051710] =============
May  4 13:24:21 VMdebian6 kernel: [ 2042.051711] myshort is a short integer: 1
May  4 13:24:21 VMdebian6 kernel: [ 2042.051713] myint is an integer: 420
May  4 13:24:21 VMdebian6 kernel: [ 2042.051714] mylong is a long integer: 9999
May  4 13:24:21 VMdebian6 kernel: [ 2042.051715] mystring is a string: blah
May  4 13:24:21 VMdebian6 kernel: [ 2042.051716] myintArray[0] = -1
May  4 13:24:21 VMdebian6 kernel: [ 2042.051717] myintArray[1] = -1
May  4 13:24:21 VMdebian6 kernel: [ 2042.051718] got 0 arguments for myintArray.
```

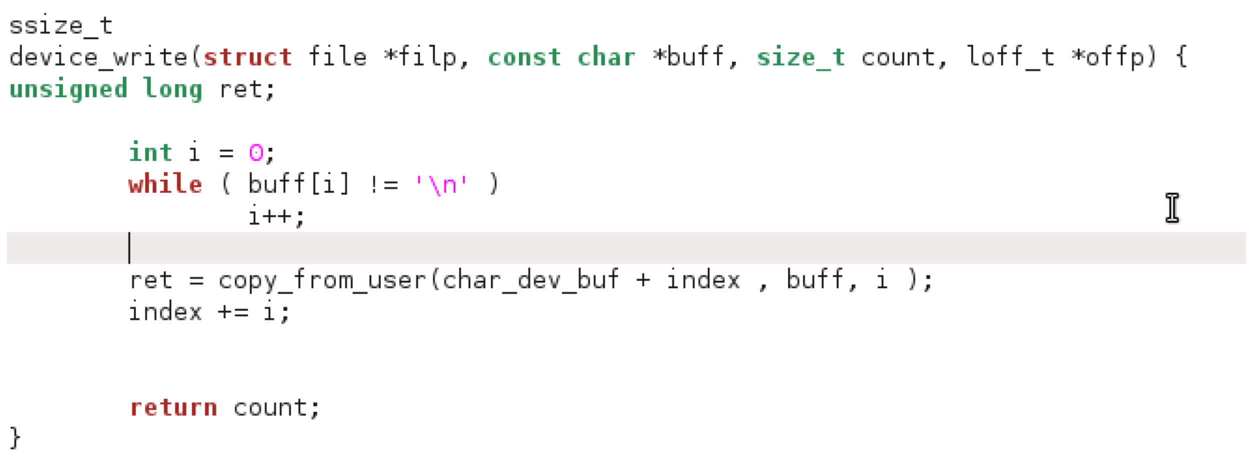
These 2 image show the 2 used command line to test the installed module.
When i computed the “remove command” ( rm hello_5) it appears this line



```
May  4 13:24:21 VMdebian6 kernel: [ 2042.051708] Hello, world 5
May  4 13:24:21 VMdebian6 kernel: [ 2042.051710] =============
May  4 13:24:21 VMdebian6 kernel: [ 2042.051711] myshort is a short integer: 1
May  4 13:24:21 VMdebian6 kernel: [ 2042.051713] myint is an integer: 420
May  4 13:24:21 VMdebian6 kernel: [ 2042.051714] mylong is a long integer: 9999
May  4 13:24:21 VMdebian6 kernel: [ 2042.051715] mystring is a string: blah
May  4 13:24:21 VMdebian6 kernel: [ 2042.051716] myintArray[0] = -1
May  4 13:24:21 VMdebian6 kernel: [ 2042.051717] myintArray[1] = -1
May  4 13:24:21 VMdebian6 kernel: [ 2042.051718] got 0 arguments for myintArray.
May  4 13:25:39 VMdebian6 kernel: [ 2120.482628] Goodbye, world 5
```

EX 2:

In the second exercise we wanted to install a module called chardev_SDP_lab.
by using the CAT command to read and the ECHO commando to write, program went in an infinite cycle while by using a test program given by the professor, the module worked.
The goal of the laboratory was to modify the drive in way that was compatible with cat and echo command.
I modified two methods: device_write and device_read.



```
ssize_t
device_write(struct file *filp, const char *buff, size_t count, loff_t *offp) {
unsigned long ret;

        int i = 0;
        while ( buff[i] != '\n' )
                i++;

        ret = copy_from_user(char_dev_buf + index , buff, i );
        index += i;


        return count;
}
```

The problem in the device_write was that the program did’t recognise the End Of File in fact didn’t copy the exact string but a bigger string.
I made a while cycle to recognise the “\n” character and once to do that i copied only the character before the “\n”.
Morevore, by adding an index equal to to the number of character of but varibale, to char_dev_buf variable i was able to save in the correct way in way that if a user compute fir 2 consecutive time an echo the content of the second echo didn't overwrite the content of the previous echo.

Instead regarding the read problem i modified the device_read function by emptying the char_dev_buf variable each time the device_read function is called and by returning to the user the number of the buffer.

```
ssize_t
device_read(struct file *filp, char *buff, size_t count, loff_t *offp) {
unsigned long ret;


        int strl = strlen ( char_dev_buf );

        count = ( count < strl )? count : strl;

        ret = copy_to_user(buff, char_dev_buf, count );
        strl = count;
        memset ( char_dev_buf , 0 , 128 );


        return count;



}
```