Operating Systems Project Report

Project Number (01 / 02 / 03):	03
Name:	蕭望緯
Student ID:	0811521
YouTube link (Format youtube.com/watch?v=[key]):	https://youtu.be/GY-x0657k4I
Date (YYYY-MM-DD):	2021-12-11
Names of the files uploaded to E3:	OS_Project03_0811521.pdf
Physical Machine Total RAM (Example: 8.0 GB):	16GB
Physical Machine CPU (Example: Intel i7-2600K):	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz

Checklist	
Yes/No	**Item**
Y	**The report name follows the format “OS_ProjectXX_StudentID.pdf”.**
Y	**The report was uploaded to E3 before the deadline.**
Y	**The YouTube video is public, and anyone with the link can watch it.**
Y	**The audio of the video has a good volume.**
Y	**The pictures in your report and video have a good quality.**
Y	**All the questions and exercises were answered inside the report.**
Y	**I understand that late submission is late submission, regardless of the time uploaded.**
Y	**I understand that any cheating in my report / video / code will not be tolerated.**

Individual questions

1. What is a static kernel module?

What is a dynamic kernel module?

What is the other name of a dynamic kernel module?

What are the differences between system calls and dynamic kernel modules (mention at least 3)?

Ans:
a. compiled as part of the base kernel and available anytime
b. compiled separately and dynamically loaded when needed
c. Loadable Kernel Modules (LKM)
d. using LKM does not require recompiling the entire kernel, while adding system calls is otherwise.
LKMs can be loaded and unloaded based on the demand, so the memory is efficiently utilized, whereas system calls occupy the memory once they are installed into the kernel.
LKMs run slower than system calls.

2. Why does adding a system call require kernel re-compilation, while adding a kernel module does not?

Ans: system calls are supported by the base kernel, so adding additional ones will lead to changes in the files of the base kernel, which will require recompilation to take effect. Kernel module is compiled separately and loaded when demand is on.

3. What are the commands **insmod**, **rmmod** and **modinfo** for?

How do you use them? (Write how would you use them with a module named **dummyModule.ko**).

Ans:

insmod: load a module into the kernel

rmmod: remove a module from the kernel

modinfo: display information(attributes) of a module

```
insmod dummyModule.ko
rmmod dummyModule
modinfo dummyModule.ko
```

4. Write the usage (parameters, what data type they are and what do they do) of the following commands:

a. module_init

b. module_exit

c. MODULE_LICENSE

d. module_param

e. MODULE_PARM_DESC

Ans:
a. **module_init(*x*)**
x(function): function to be called at module insertion time
b. **module_exit(*x*)**

x(function): function to be called at module insertion time
c. **MODULE_LICENSE(x)**
x(string): license name (e.g., “GPL”, “Dual BSD/GPL”)
d. **module_param(name, type, perm)**
name: variable name
type: its type (e.g., int, bool, etc.)
perm(int): permissions for the corresponding file in sysfs
e. **MODULE_PARM_DESC(name, desc)**
name: variable name
desc(string): description of the variable

5. What do the following terminal commands mean (explain what they do and what does the -x mean in each case):

a. cat

b. ls -l

c. dmesg -wH

d. lsmod

e. lsmod | grep

Ans:
a. read or write content to files
b. list file in a directory. **-l:** show file or directory, size, modified date and time, file or folder name and owner of the file, and its permission.
c. read messages stored in the ring buffer. **-H:** enable human-readable output. **-w**: Wait for new messages.
d. show what kernel modules are currently loaded.
e. search with keywords and show whether related kernel modules are currently loaded

6. There is a 0644 in the line

```
module_param(studentId, int, 0644);
```

inside **paramsModule.c** (Section 1.2). What does 0644 mean?
Ans: owner can read and write, group can read, and everyone else can read.

7. What happens if the initialization function of the module returns -1?

What type of error do you get?

Ans: show loading error messages; operation not permitted.

8. In Section 1.2 – step 6, **modinfo** shows the information of some variables inside the module but two of them are not displayed. Why is it?

Ans: **dummyStudentId** and **dummySecretValue** are not declared as module parameters, so they are not shown in the message of **modinfo**.

9. What is the **/sys/module** folder for?

Ans: store information of each kernel module including parameters and reference counts.

10. In Section 1.2 (paramsModule.c), the variable **charparameter** is of type **charp**. What is charp?
Ans: string

Additional questions (also answer in the report):
11. Which project (01 / 02 / 03) did you like the most? Why?
Project 3. I do not need to rebuild the kernel, which takes a lot of time.

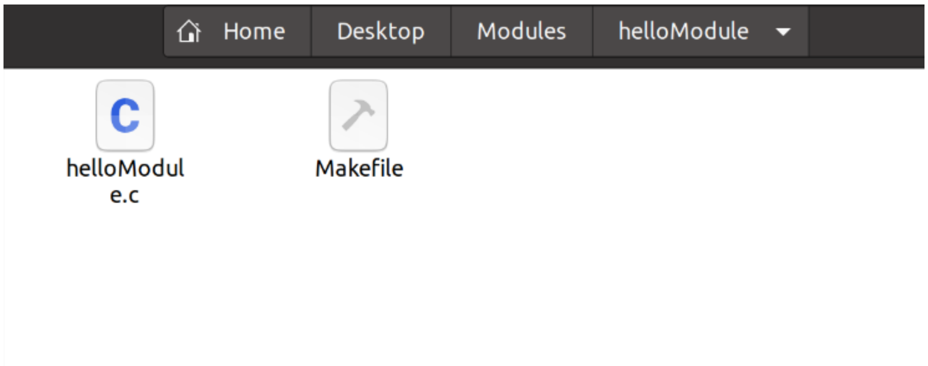
12. Which project (01 / 02 / 03) did you like the least? Why?
Project 1. I must wait for a while when the kernel is rebuilt. However, Project 1 lays the foundation of knowledge, so I think it is worth going through the process.

13. Did you learn anything new with these three projects? What did you learn?
Literally everything in the projects is new for me. I learn about how to work on kernels and also gain experience with Linux environment.

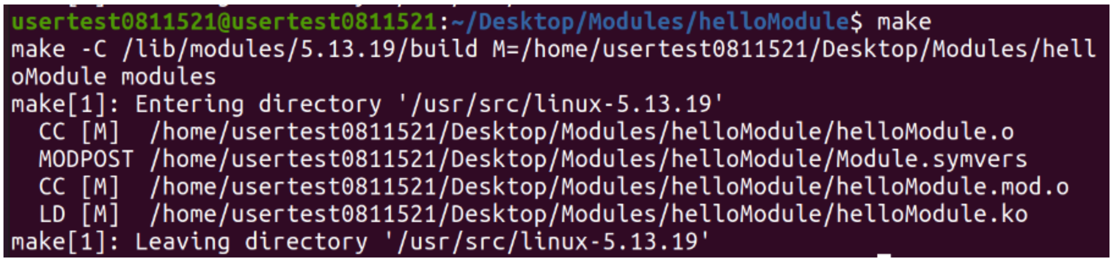
14. Do you think these projects can help you in the future, if you look for a job in the industry?
Even though I might not be researching and programming on operating systems in the near future, it is good to be knowledgeable about the wheels under them as a programmer.

Screenshots

#1 helloModule.c is the module to be compiled.

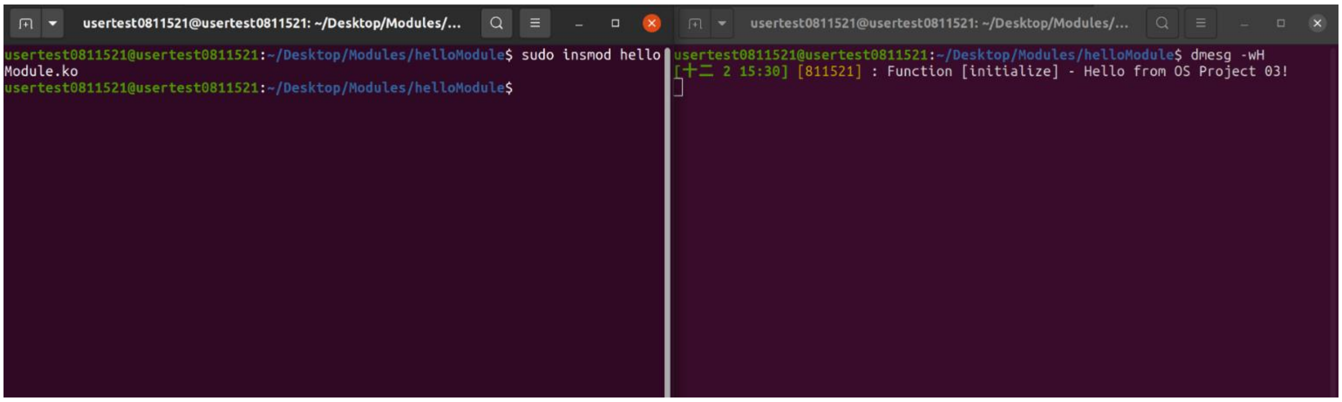


#2 compile the module



```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ make
make -C /lib/modules/5.13.19/build M=/home/usertest0811521/Desktop/Modules/hell
oModule modules
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CC [M]  /home/usertest0811521/Desktop/Modules/helloModule/helloModule.o
  MODPOST /home/usertest0811521/Desktop/Modules/helloModule/Module.symvers
  CC [M]  /home/usertest0811521/Desktop/Modules/helloModule/helloModule.mod.o
  LD [M]  /home/usertest0811521/Desktop/Modules/helloModule/helloModule.ko
make[1]: Leaving directory '/usr/src/linux-5.13.19'
```

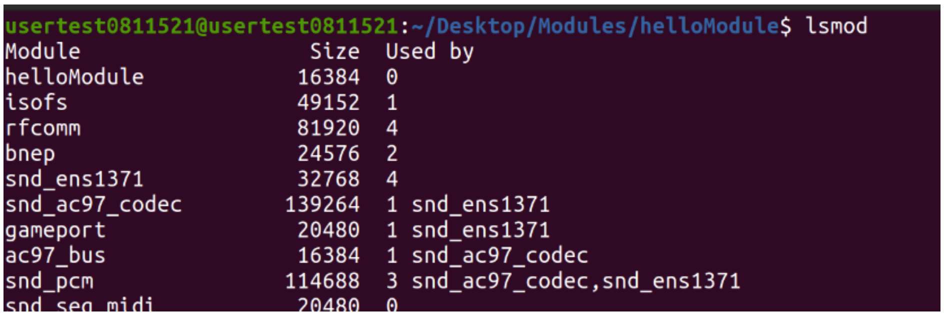
#3 load the module and read the message from the module initialization function in the ring buffer


usertest0811521@usertest0811521: ~/Desktop/Modules/...
usertest0811521@usertest0811521: ~/Desktop/Modules/...

```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ sudo insmod hello
Module.ko
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$
```

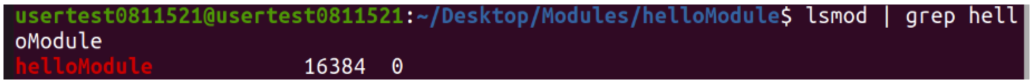
```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ dmesg -wH
[十二 2 15:30] [811521] : Function [initialize] - Hello from OS Project 03!
```

#4 the list of loaded modules



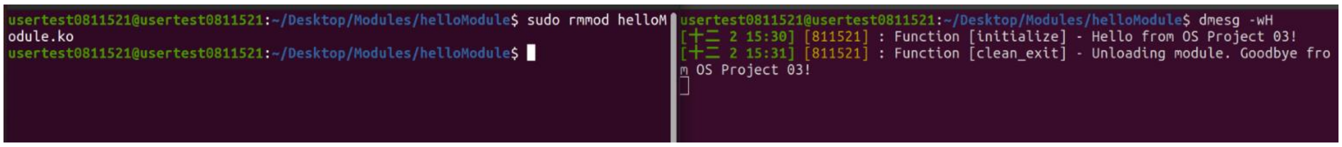
```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ lsmod
Module                  Size  Used by
helloModule            16384  0
isofs                  49152  1
rfcomm                 81920  4
bnep                   24576  2
snd_ens1371            32768  4
snd_ac97_codec        139264  1 snd_ens1371
gameport               20480  1 snd_ens1371
ac97_bus               16384  1 snd_ac97_codec
snd_pcm               114688  3 snd_ac97_codec,snd_ens1371
snd_seq_midi           20480  0
```

search the list with keywords



```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ lsmod | grep hell
oModule
helloModule            16384  0
```

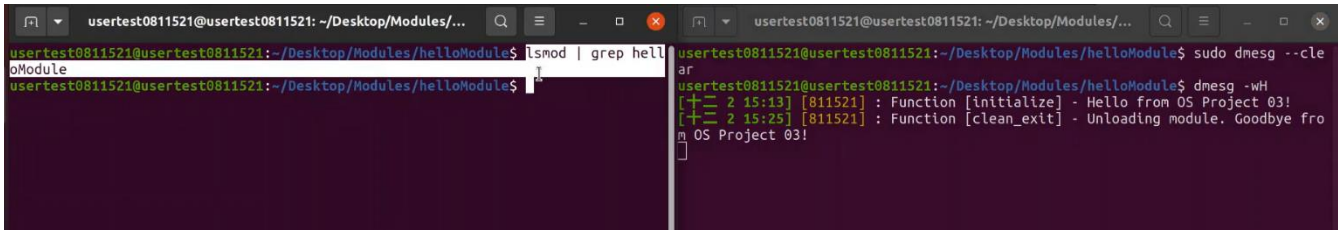
#5 unload the module and read the message from the module exiting function



```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ sudo rmmod helloM
odule.ko
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ dmesg -wH
[十二 2 15:30] [811521] : Function [initialize] - Hello from OS Project 03!
[十二 2 15:31] [811521] : Function [clean_exit] - Unloading module. Goodbye fro
m OS Project 03!
```

#6 after unloading the module, we cannot find it in the list

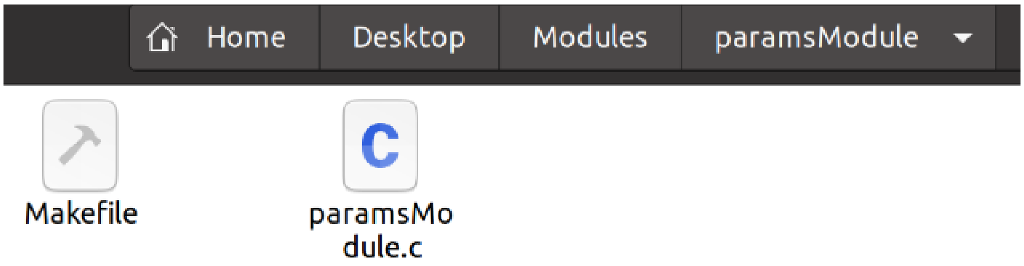


usertest0811521@usertest0811521: ~/Desktop/Modules/...
usertest0811521@usertest0811521: ~/Desktop/Modules/...

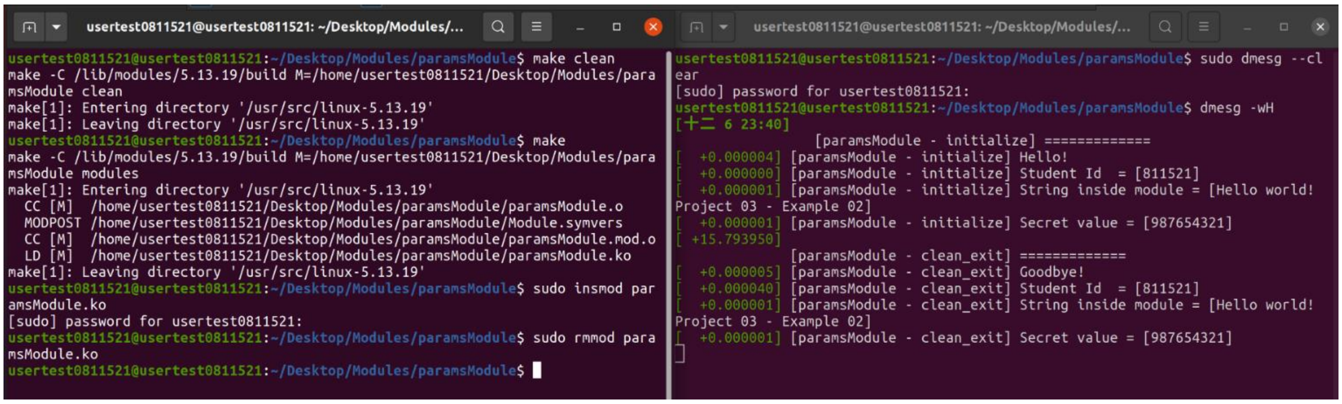
```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ lsmod | grep hell
oModule
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ sudo dmesg --cle
ar
usertest0811521@usertest0811521:~/Desktop/Modules/helloModule$ dmesg -wH
[十二 2 15:13] [811521] : Function [initialize] - Hello from OS Project 03!
[十二 2 15:25] [811521] : Function [clean_exit] - Unloading module. Goodbye fro
m OS Project 03!
```

#7 paramsModule.c is the module to be compiled.



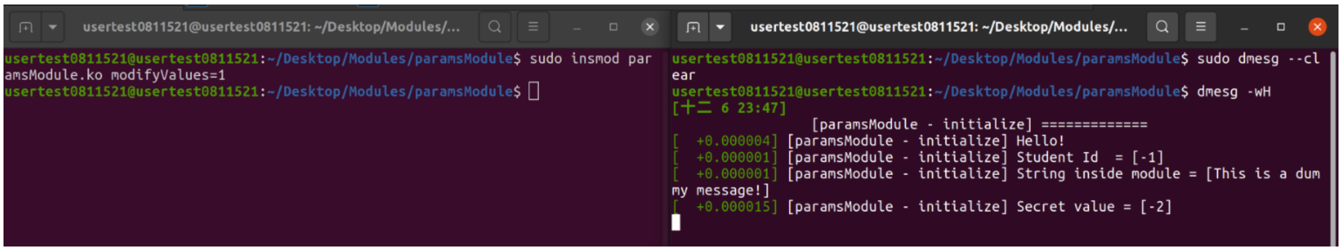
#8 without passing parameters in the terminal, **modifyValues=0** by default. The kernel message displays default values.


usertest0811521@usertest0811521: ~/Desktop/Modules/...
usertest0811521@usertest0811521: ~/Desktop/Modules/...

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ make clean
make -C /lib/modules/5.13.19/build M=/home/usertest0811521/Desktop/Modules/para
msModule clean
make[1]: Entering directory '/usr/src/linux-5.13.19'
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ make
make -C /lib/modules/5.13.19/build M=/home/usertest0811521/Desktop/Modules/para
msModule modules
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CC [M]  /home/usertest0811521/Desktop/Modules/paramsModule/paramsModule.o
  MODPOST /home/usertest0811521/Desktop/Modules/paramsModule/Module.symvers
  CC [M]  /home/usertest0811521/Desktop/Modules/paramsModule/paramsModule.mod.o
  LD [M]  /home/usertest0811521/Desktop/Modules/paramsModule/paramsModule.ko
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo insmod par
amsModule.ko
[sudo] password for usertest0811521:
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo rmmod para
msModule.ko
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ 
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo dmesg --cl
ear
[sudo] password for usertest0811521:
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ dmesg -wH
[十二 6 23:40]
                 [paramsModule - initialize] =============
[  +0.000004] [paramsModule - initialize] Hello!
[  +0.000000] [paramsModule - initialize] Student Id  = [811521]
[  +0.000001] [paramsModule - initialize] String inside module = [Hello world!
Project 03 - Example 02]
[  +0.000001] [paramsModule - initialize] Secret value = [987654321]
[ +15.793950]
              [paramsModule - clean_exit] =============
[  +0.000005] [paramsModule - clean_exit] Goodbye!
[  +0.000040] [paramsModule - clean_exit] Student Id  = [811521]
[  +0.000001] [paramsModule - clean_exit] String inside module = [Hello world!
Project 03 - Example 02]
[  +0.000001] [paramsModule - clean_exit] Secret value = [987654321]
```

#9 When passing the parameter in the terminal with **modifyValues=1**, the initialization function assigns new values to some variables and the terminal T2 shows modified values.

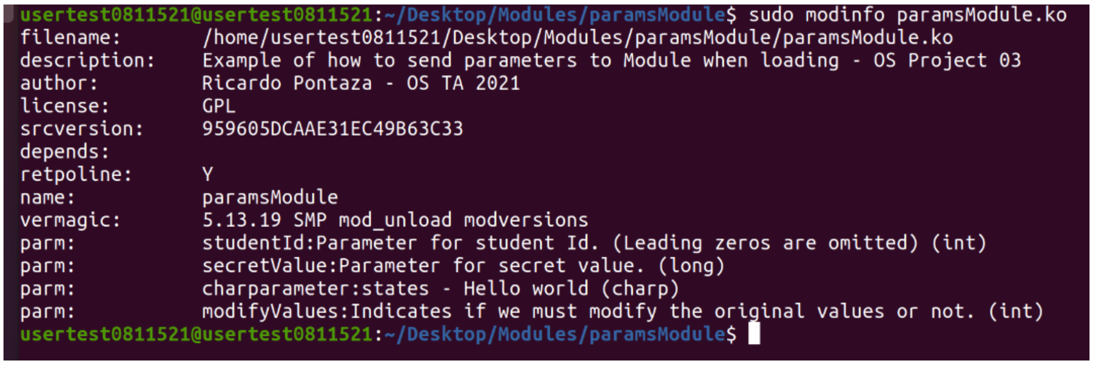

usertest0811521@usertest0811521: ~/Desktop/Modules/...
usertest0811521@usertest0811521: ~/Desktop/Modules/...

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo insmod par
amsModule.ko modifyValues=1
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ 
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo dmesg --cl
ear
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ dmesg -wH
[十二 6 23:47]
                 [paramsModule - initialize] =============
[  +0.000004] [paramsModule - initialize] Hello!
[  +0.000001] [paramsModule - initialize] Student Id  = [-1]
[  +0.000001] [paramsModule - initialize] String inside module = [This is a dum
my message!]
[  +0.000015] [paramsModule - initialize] Secret value = [-2]
```

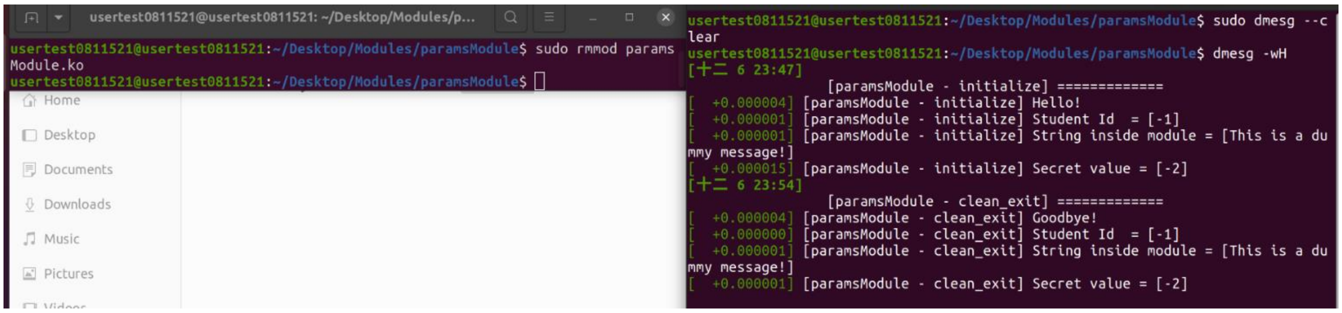
#10 **modinfo** show the author, description, license, module parameters and filename

dummyStudentId and **dummySecretValue** are not declared as module parameters, so they are not shown in the message of **modinfo**.



```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo modinfo paramsModule.ko
filename:       /home/usertest0811521/Desktop/Modules/paramsModule/paramsModule.ko
description:    Example of how to send parameters to Module when loading - OS Project 03
author:         Ricardo Pontaza - OS TA 2021
license:        GPL
srcversion:     959605DCAAE31EC49B63C33
depends:        
retpoline:      Y
name:           paramsModule
vermagic:       5.13.19 SMP mod_unload modversions 
parm:           studentId:Parameter for student Id. (Leading zeros are omitted) (int)
parm:           secretValue:Parameter for secret value. (long)
parm:           charparameter:states - Hello world (charp)
parm:           modifyValues:Indicates if we must modify the original values or not. (int)
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ 
```

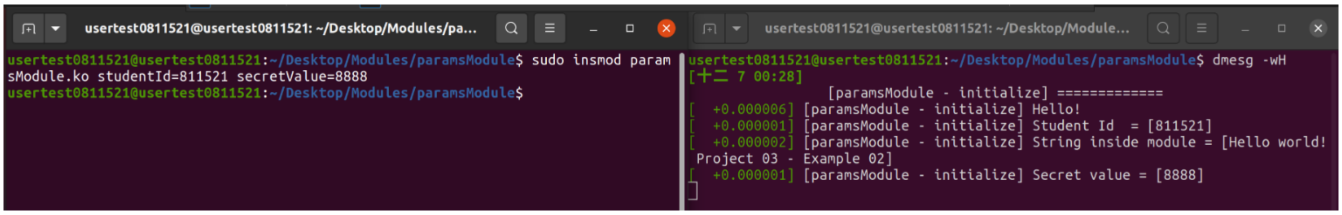
After removing the module, the terminal T2 shows modified values as well.



```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo rmmod params
Module.ko
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ 
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo dmesg --c
lear
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ dmesg -wH
[十二 6 23:47]
              [paramsModule - initialize] ============
[  +0.000004] [paramsModule - initialize] Hello!
[  +0.000001] [paramsModule - initialize] Student Id  = [-1]
[  +0.000001] [paramsModule - initialize] String inside module = [This is a du
mmy message!]
[  +0.000015] [paramsModule - initialize] Secret value = [-2]
[十二 6 23:54]
              [paramsModule - clean_exit] ============
[  +0.000004] [paramsModule - clean_exit] Goodbye!
[  +0.000000] [paramsModule - clean_exit] Student Id  = [-1]
[  +0.000001] [paramsModule - clean_exit] String inside module = [This is a du
mmy message!]
[  +0.000001] [paramsModule - clean_exit] Secret value = [-2]
```

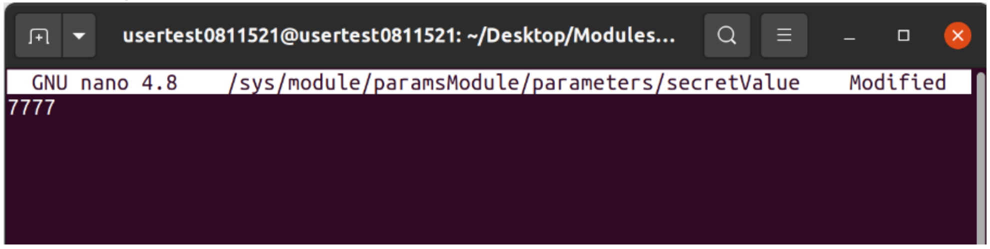
#11 By passing in parameters **studentId** and **secretValue** in the terminal, the terminal will show new values.


usertest0811521@usertest0811521: ~/Desktop/Modules/pa...
usertest0811521@usertest0811521: ~/Desktop/Module...

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo insmod param
sModule.ko studentId=811521 secretValue=8888
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$
```

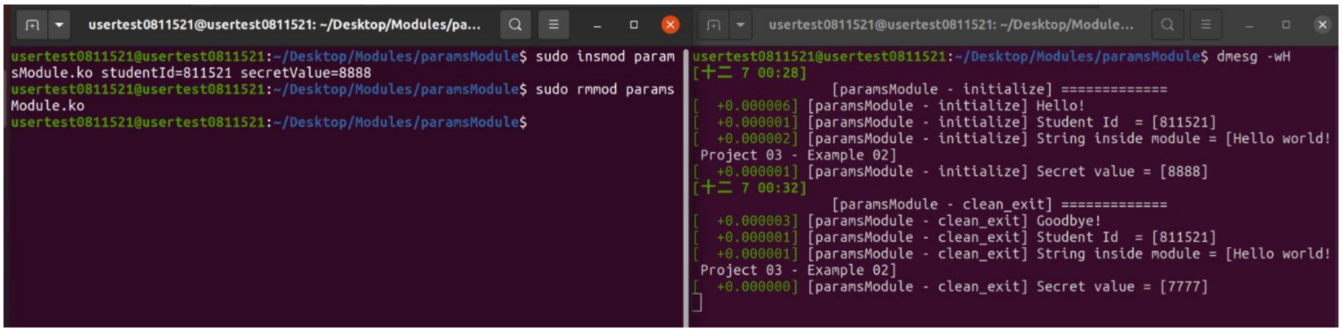
```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ dmesg -wH
[十二 7 00:28]
              [paramsModule - initialize] ============
[  +0.000006] [paramsModule - initialize] Hello!
[  +0.000001] [paramsModule - initialize] Student Id  = [811521]
[  +0.000002] [paramsModule - initialize] String inside module = [Hello world!
 Project 03 - Example 02]
[  +0.000001] [paramsModule - initialize] Secret value = [8888]
```

#12 By modifying the parameter value in the folder **/sys/module/<name of module>/parameters**, we can change the value at runtime.



```
  GNU nano 4.8      /sys/module/paramsModule/parameters/secretValue      Modified
7777
```

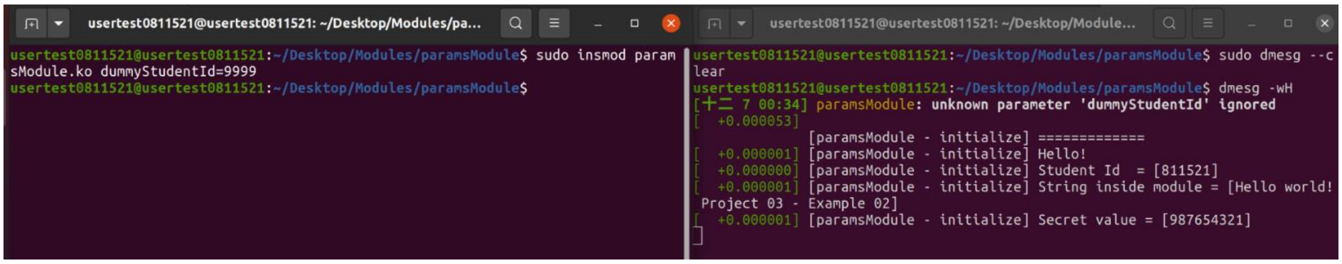
#13 When the module is unloaded, we see the manually assigned value in the message.


usertest0811521@usertest0811521: ~/Desktop/Modules/pa...
usertest0811521@usertest0811521: ~/Desktop/Module...

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo insmod param
sModule.ko studentId=811521 secretValue=8888
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo rmmod params
Module.ko
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ dmesg -wH
[十二 7 00:28]
              [paramsModule - initialize] ============
[  +0.000006] [paramsModule - initialize] Hello!
[  +0.000001] [paramsModule - initialize] Student Id  = [811521]
[  +0.000002] [paramsModule - initialize] String inside module = [Hello world!
 Project 03 - Example 02]
[  +0.000001] [paramsModule - initialize] Secret value = [8888]
[十二 7 00:32]
              [paramsModule - clean_exit] ============
[  +0.000003] [paramsModule - clean_exit] Goodbye!
[  +0.000001] [paramsModule - clean_exit] Student Id  = [811521]
[  +0.000001] [paramsModule - clean_exit] String inside module = [Hello world!
 Project 03 - Example 02]
[  +0.000000] [paramsModule - clean_exit] Secret value = [7777]
```

#14 **dummyStudentId** and **dummySecretValue** are not declared as module parameters, so they are unknown to the kernel module.



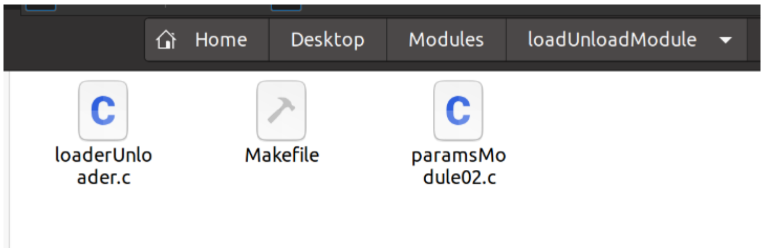
usertest0811521@usertest0811521: ~/Desktop/Modules/pa...
usertest0811521@usertest0811521: ~/Desktop/Module...

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo insmod param
sModule.ko dummyStudentId=9999
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ sudo dmesg --c
lear
usertest0811521@usertest0811521:~/Desktop/Modules/paramsModule$ dmesg -wH
[十二 7 00:34] paramsModule: unknown parameter 'dummyStudentId' ignored
[  +0.000053]
               [paramsModule - initialize] =============
[  +0.000001] [paramsModule - initialize] Hello!
[  +0.000000] [paramsModule - initialize] Student Id  = [811521]
[  +0.000001] [paramsModule - initialize] String inside module = [Hello world!
 Project 03 - Example 02]
[  +0.000001] [paramsModule - initialize] Secret value = [987654321]
]
```

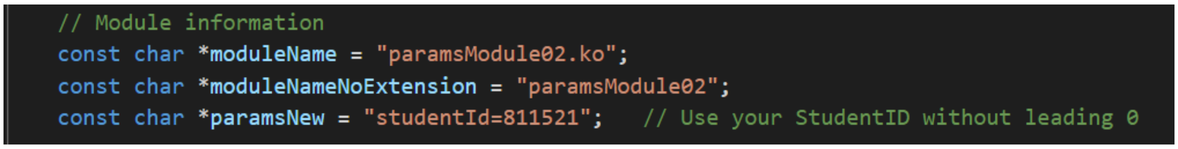
#15

loaderUnloader.c is the program that loads and unloads a module. **paramsModule02.c** is the source code of the module.



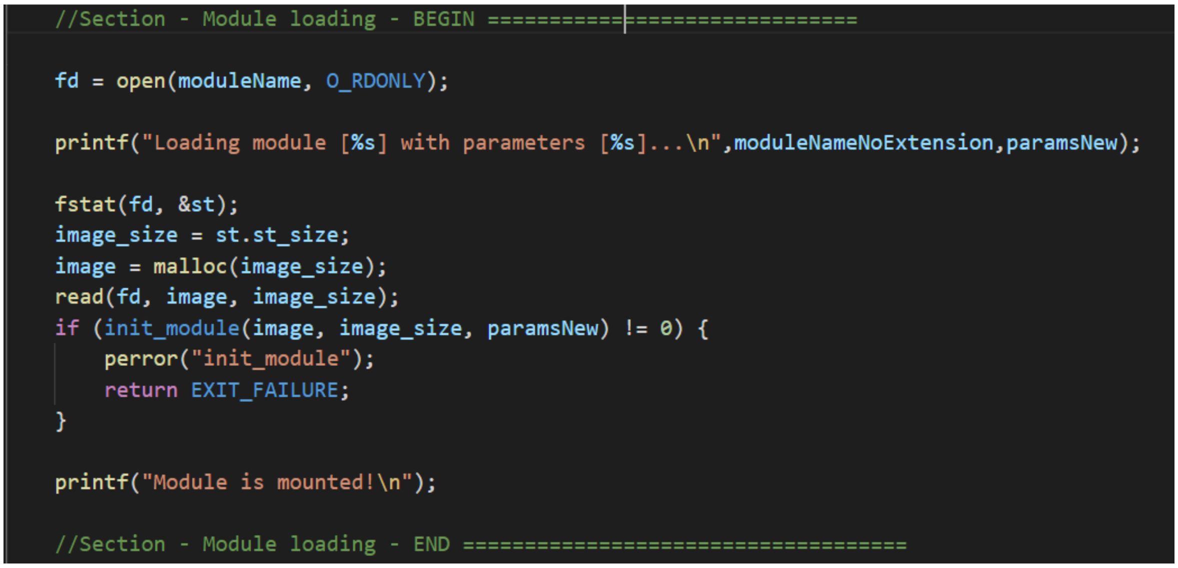
#16

paramsNew contains the parameters to be passed into the module.



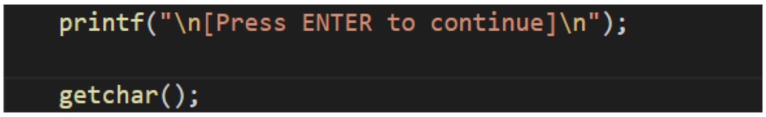
```
// Module information
const char *moduleName = "paramsModule02.ko";
const char *moduleNameNoExtension = "paramsModule02";
const char *paramsNew = "studentId=811521";   // Use your StudentID without leading 0
```

load the module.



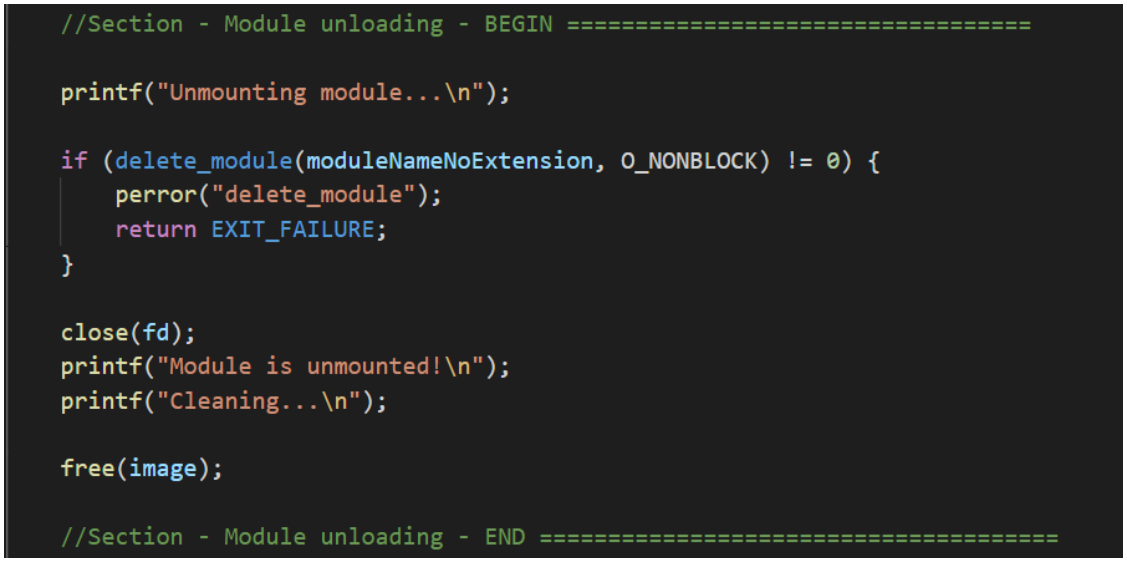
```
    //Section - Module loading - BEGIN ==============================

    fd = open(moduleName, O_RDONLY);

    printf("Loading module [%s] with parameters [%s]...\n",moduleNameNoExtension,paramsNew);

    fstat(fd, &st);
    image_size = st.st_size;
    image = malloc(image_size);
    read(fd, image, image_size);
    if (init_module(image, image_size, paramsNew) != 0) {
        perror("init_module");
        return EXIT_FAILURE;
    }

    printf("Module is mounted!\n");

    //Section - Module loading - END ====================================
```

wait for input and the module is loaded.



```
    printf("\n[Press ENTER to continue]\n");

    getchar();
```

unload the module.



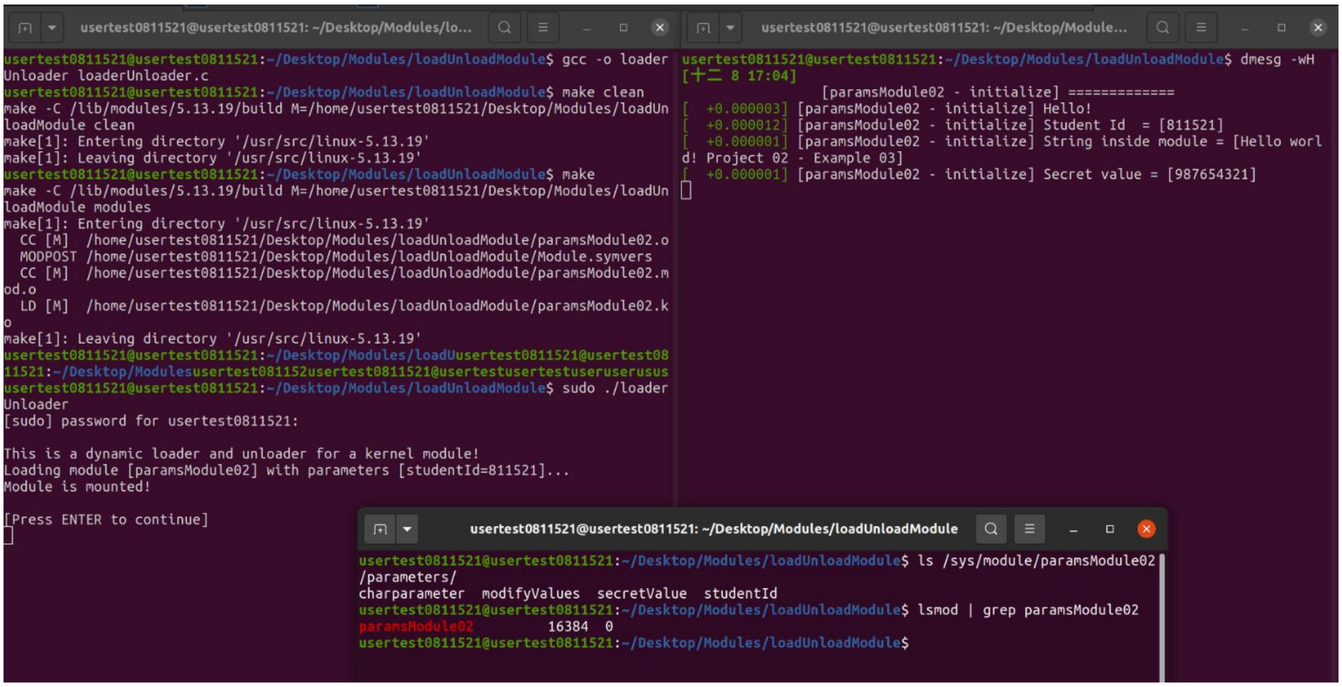
```
    //Section - Module unloading - BEGIN ==================================

    printf("Unmounting module...\n");

    if (delete_module(moduleNameNoExtension, O_NONBLOCK) != 0) {
        perror("delete_module");
        return EXIT_FAILURE;
    }

    close(fd);
    printf("Module is unmounted!\n");
    printf("Cleaning...\n");

    free(image);

    //Section - Module unloading - END ======================================
```

#17 We compile **loaderUnloader.c** and compile the module **paramsModule02.c**.

Then, we execute **./loaderUnloader**, which will load the module, stop at **getchar()** and wait for input.

In terminal T2 (messages in the ring buffer), we can see the value passed from the user program (paramsNew=”studentId=…”).

At this moment, we can check the module parameters under **/sys/module/<name of module>/parameters.** The module is loaded, so we can find it with **lsmod**.

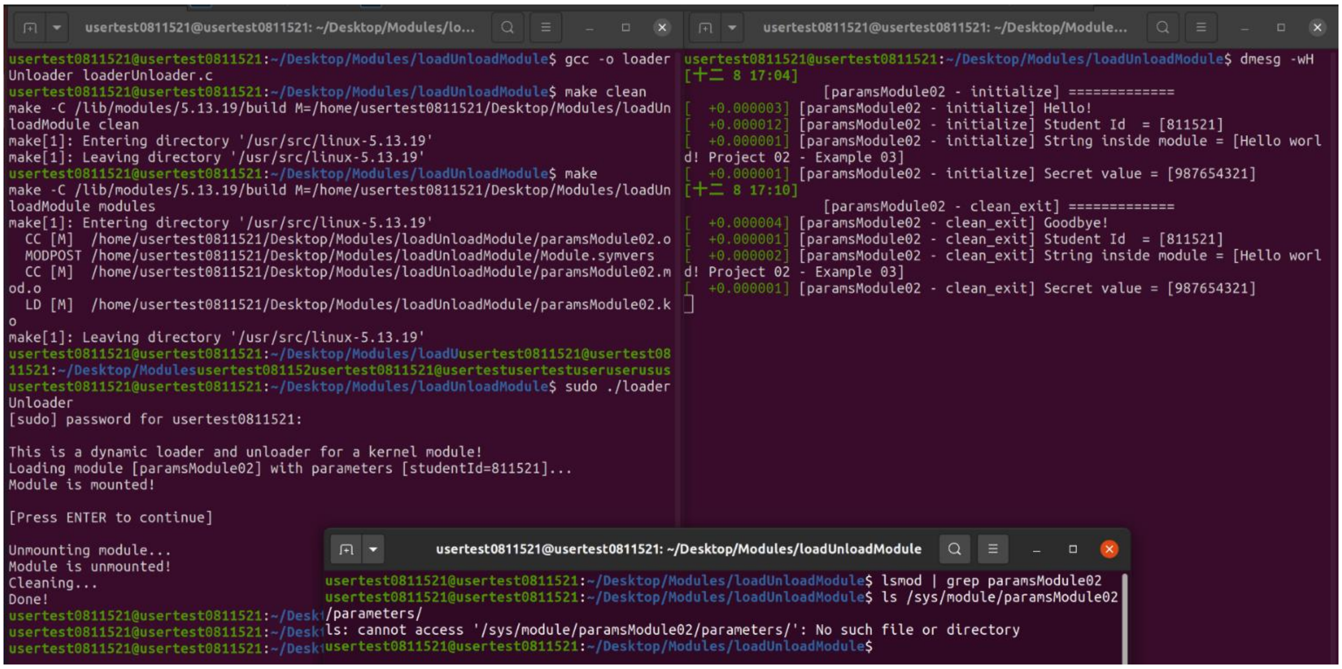

usertest0811521@usertest0811521: ~/Desktop/Modules/lo...
usertest0811521@usertest0811521: ~/Desktop/Module...
usertest0811521@usertest0811521: ~/Desktop/Modules/loadUnloadModule

```
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ gcc -o loader
Unloader loaderUnloader.c
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ make clean
make -C /lib/modules/5.13.19/build M=/home/usertest0811521/Desktop/Modules/loadUn
loadModule clean
make[1]: Entering directory '/usr/src/linux-5.13.19'
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ make
make -C /lib/modules/5.13.19/build M=/home/usertest0811521/Desktop/Modules/loadUn
loadModule modules
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CC [M]  /home/usertest0811521/Desktop/Modules/loadUnloadModule/paramsModule02.o
  MODPOST /home/usertest0811521/Desktop/Modules/loadUnloadModule/Module.symvers
  CC [M]  /home/usertest0811521/Desktop/Modules/loadUnloadModule/paramsModule02.m
od.o
  LD [M]  /home/usertest0811521/Desktop/Modules/loadUnloadModule/paramsModule02.k
o
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest0811521@usertest0811521:~/Desktop/Modules/loadUusertest0811521@usertest08
11521:~/Desktop/Modulesusertest081152usertest0811521@usertestusertestuseruserusus
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ sudo ./loader
Unloader
[sudo] password for usertest0811521:

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=811521]...
Module is mounted!

[Press ENTER to continue]
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ dmesg -wH
[十二 8 17:04]
                 [paramsModule02 - initialize] =============
[  +0.000003] [paramsModule02 - initialize] Hello!
[  +0.000012] [paramsModule02 - initialize] Student Id  = [811521]
[  +0.000001] [paramsModule02 - initialize] String inside module = [Hello worl
d! Project 02 - Example 03]
[  +0.000001] [paramsModule02 - initialize] Secret value = [987654321]
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ ls /sys/module/paramsModule02
/parameters/
charparameter  modifyValues  secretValue  studentId
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ lsmod | grep paramsModule02
paramsModule02         16384  0
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$
```

#18 After we press ENTER, **loaderUnloader** proceeds to unload the module. Terminal T2 shows messages from exiting function in the module.


usertest0811521@usertest0811521: ~/Desktop/Modules/lo...
usertest0811521@usertest0811521: ~/Desktop/Module...
usertest0811521@usertest0811521: ~/Desktop/Modules/loadUnloadModule

```
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ gcc -o loader
Unloader loaderUnloader.c
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ make clean
make -C /lib/modules/5.13.19/build M=/home/usertest0811521/Desktop/Modules/loadUn
loadModule clean
make[1]: Entering directory '/usr/src/linux-5.13.19'
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ make
make -C /lib/modules/5.13.19/build M=/home/usertest0811521/Desktop/Modules/loadUn
loadModule modules
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CC [M]  /home/usertest0811521/Desktop/Modules/loadUnloadModule/paramsModule02.o
  MODPOST /home/usertest0811521/Desktop/Modules/loadUnloadModule/Module.symvers
  CC [M]  /home/usertest0811521/Desktop/Modules/loadUnloadModule/paramsModule02.m
od.o
  LD [M]  /home/usertest0811521/Desktop/Modules/loadUnloadModule/paramsModule02.k
o
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest0811521@usertest0811521:~/Desktop/Modules/loadUusertest0811521@usertest08
11521:~/Desktop/Modulesusertest081152usertest0811521@usertestusertestuseruserusus
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ sudo ./loader
Unloader
[sudo] password for usertest0811521:

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=811521]...
Module is mounted!

[Press ENTER to continue]

Unmounting module...
Module is unmounted!
Cleaning...
Done!
usertest0811521@usertest0811521:~/Desk
usertest0811521@usertest0811521:~/Desk
usertest0811521@usertest0811521:~/Desk
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ dmesg -wH
[十二 8 17:04]
                 [paramsModule02 - initialize] =============
[  +0.000003] [paramsModule02 - initialize] Hello!
[  +0.000012] [paramsModule02 - initialize] Student Id  = [811521]
[  +0.000001] [paramsModule02 - initialize] String inside module = [Hello worl
d! Project 02 - Example 03]
[  +0.000001] [paramsModule02 - initialize] Secret value = [987654321]
[十二 8 17:10]
                 [paramsModule02 - clean_exit] =============
[  +0.000004] [paramsModule02 - clean_exit] Goodbye!
[  +0.000001] [paramsModule02 - clean_exit] Student Id  = [811521]
[  +0.000002] [paramsModule02 - clean_exit] String inside module = [Hello worl
d! Project 02 - Example 03]
[  +0.000001] [paramsModule02 - clean_exit] Secret value = [987654321]
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ lsmod | grep paramsModule02
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$ ls /sys/module/paramsModule02
/parameters/
ls: cannot access '/sys/module/paramsModule02/parameters/': No such file or directory
usertest0811521@usertest0811521:~/Desktop/Modules/loadUnloadModule$
```

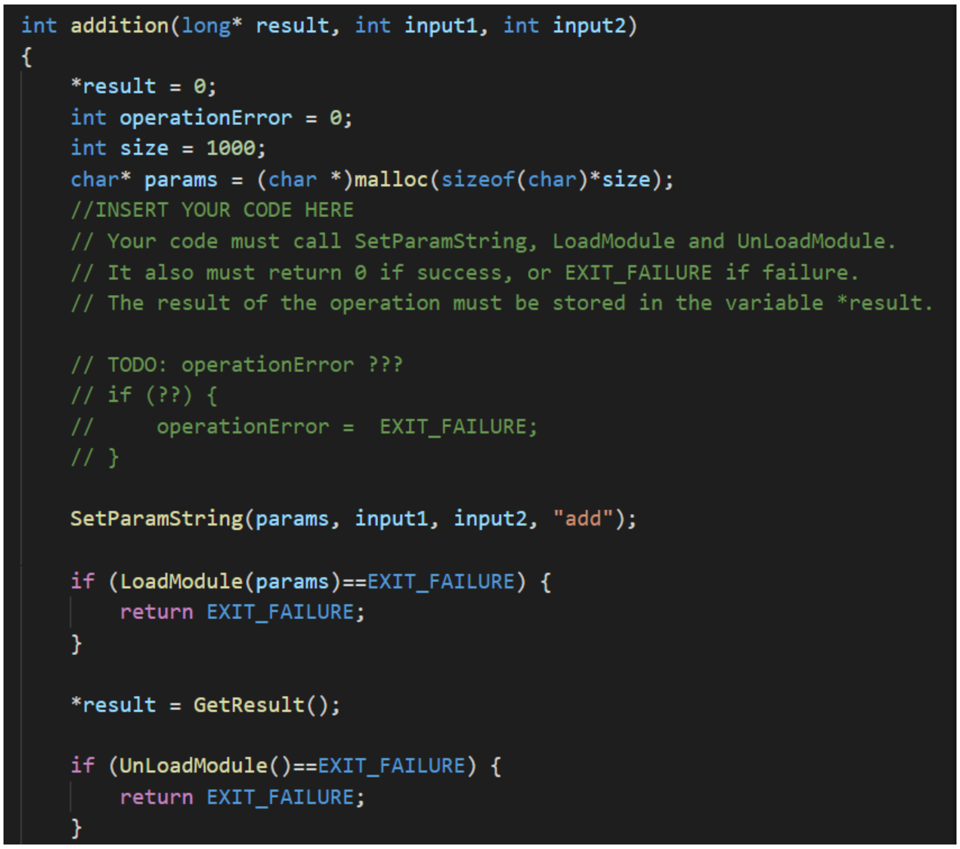
#19

In **calculator.c**, the addition, subtraction and multiplication functions basically do four things:

- call **setParamString()** to make a string argument of parameters
- call **LoadModule()** to load, initialize and pass the parameters to the module

- call **GetResult()** to read the result value from **/sys/module/calculatorModule/parameters/resultParam**
- call **UnloadModule()** to unload the module

calculatorModule.c will do a specific arithmetic calculation on the input values based on the **operationParam**.

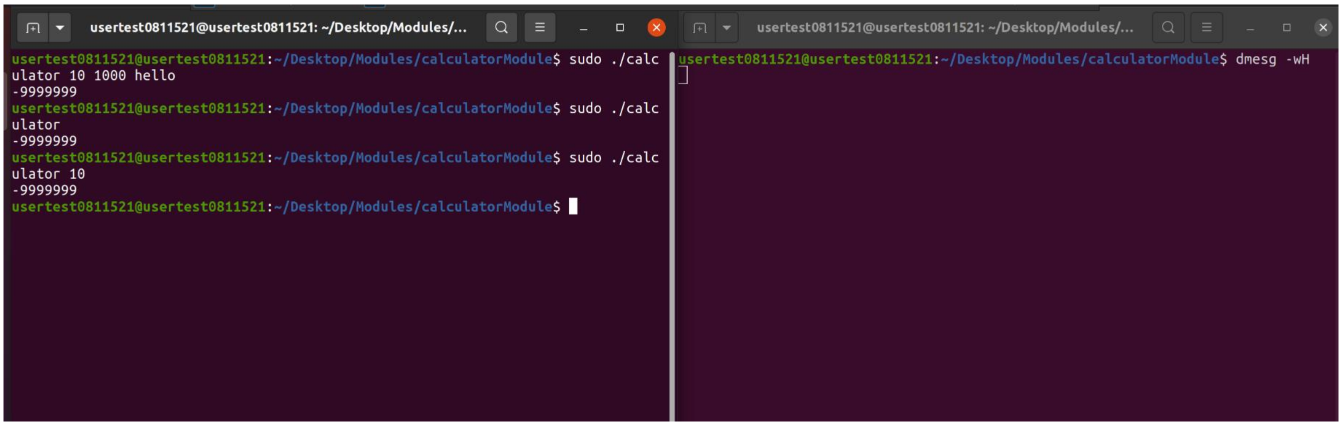


```
int addition(long* result, int input1, int input2)
{
    *result = 0;
    int operationError = 0;
    int size = 1000;
    char* params = (char *)malloc(sizeof(char)*size);
    //INSERT YOUR CODE HERE
    // Your code must call SetParamString, LoadModule and UnLoadModule.
    // It also must return 0 if success, or EXIT_FAILURE if failure.
    // The result of the operation must be stored in the variable *result.

    // TODO: operationError ???
    // if (??) {
    //     operationError =  EXIT_FAILURE;
    // }

    SetParamString(params, input1, input2, "add");

    if (LoadModule(params)==EXIT_FAILURE) {
        return EXIT_FAILURE;
    }

    *result = GetResult();

    if (UnLoadModule()==EXIT_FAILURE) {
        return EXIT_FAILURE;
    }
```

#20

Execution Results

If we pass wrong arguments, the program will not load the module and execute the arithmetic operation.

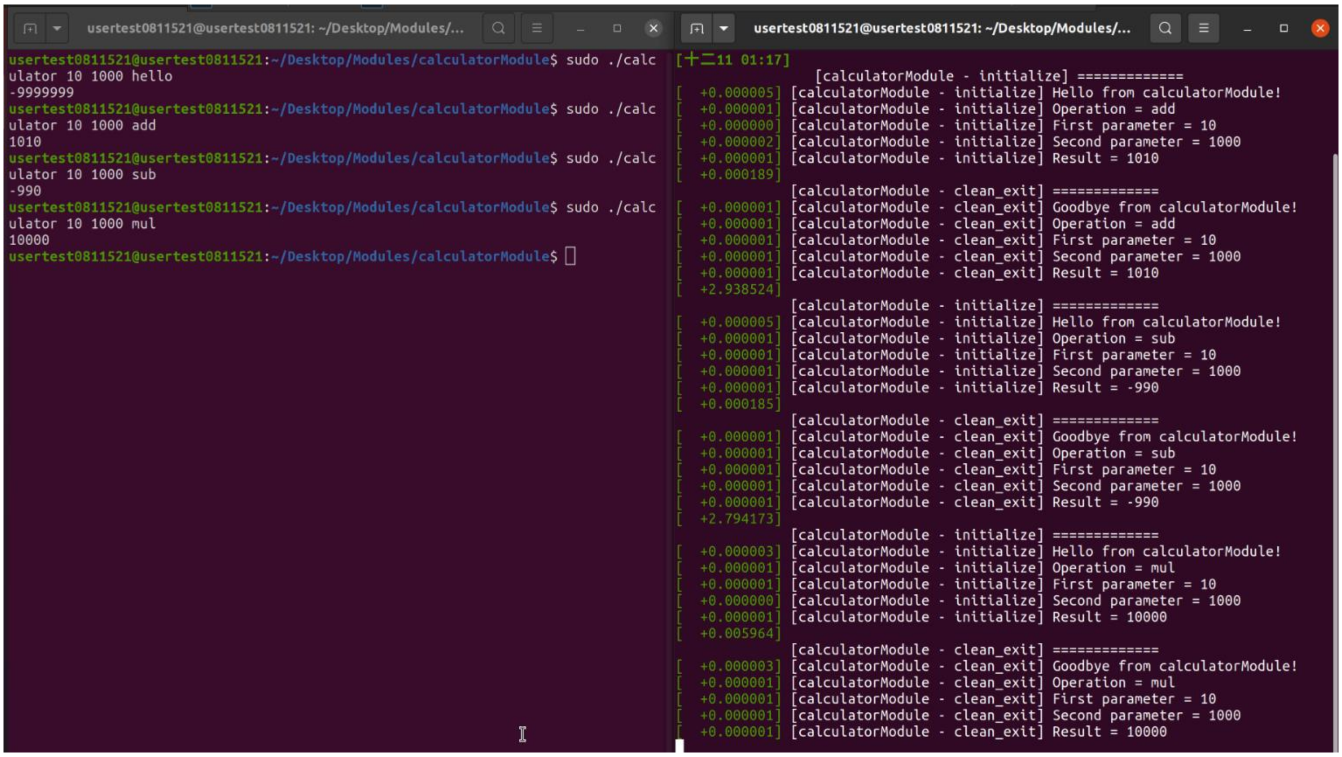


usertest0811521@usertest0811521: ~/Desktop/Modules/...
usertest0811521@usertest0811521: ~/Desktop/Modules/...

```
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$ sudo ./calc
ulator 10 1000 hello
-9999999
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$ sudo ./calc
ulator
-9999999
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$ sudo ./calc
ulator 10
-9999999
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$
```

```
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$ dmesg -wH
```

Results of normal operations and kernel messages in terminal T2.



usertest0811521@usertest0811521: ~/Desktop/Modules/...
usertest0811521@usertest0811521: ~/Desktop/Modules/...

```
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$ sudo ./calc
ulator 10 1000 hello
-9999999
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$ sudo ./calc
ulator 10 1000 add
1010
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$ sudo ./calc
ulator 10 1000 sub
-990
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$ sudo ./calc
ulator 10 1000 mul
10000
usertest0811521@usertest0811521:~/Desktop/Modules/calculatorModule$
```

```
[十二11 01:17]
                 [calculatorModule - initialize] =============
[  +0.000005] [calculatorModule - initialize] Hello from calculatorModule!
[  +0.000001] [calculatorModule - initialize] Operation = add
[  +0.000000] [calculatorModule - initialize] First parameter = 10
[  +0.000002] [calculatorModule - initialize] Second parameter = 1000
[  +0.000001] [calculatorModule - initialize] Result = 1010
[  +0.000189]
              [calculatorModule - clean_exit] =============
[  +0.000001] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[  +0.000001] [calculatorModule - clean_exit] Operation = add
[  +0.000001] [calculatorModule - clean_exit] First parameter = 10
[  +0.000001] [calculatorModule - clean_exit] Second parameter = 1000
[  +0.000001] [calculatorModule - clean_exit] Result = 1010
[  +2.938524]
              [calculatorModule - initialize] =============
[  +0.000005] [calculatorModule - initialize] Hello from calculatorModule!
[  +0.000001] [calculatorModule - initialize] Operation = sub
[  +0.000001] [calculatorModule - initialize] First parameter = 10
[  +0.000001] [calculatorModule - initialize] Second parameter = 1000
[  +0.000001] [calculatorModule - initialize] Result = -990
[  +0.000185]
              [calculatorModule - clean_exit] =============
[  +0.000001] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[  +0.000001] [calculatorModule - clean_exit] Operation = sub
[  +0.000001] [calculatorModule - clean_exit] First parameter = 10
[  +0.000001] [calculatorModule - clean_exit] Second parameter = 1000
[  +0.000001] [calculatorModule - clean_exit] Result = -990
[  +2.794173]
              [calculatorModule - initialize] =============
[  +0.000003] [calculatorModule - initialize] Hello from calculatorModule!
[  +0.000001] [calculatorModule - initialize] Operation = mul
[  +0.000001] [calculatorModule - initialize] First parameter = 10
[  +0.000000] [calculatorModule - initialize] Second parameter = 1000
[  +0.000001] [calculatorModule - initialize] Result = 10000
[  +0.005964]
              [calculatorModule - clean_exit] =============
[  +0.000003] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[  +0.000001] [calculatorModule - clean_exit] Operation = mul
[  +0.000001] [calculatorModule - clean_exit] First parameter = 10
[  +0.000001] [calculatorModule - clean_exit] Second parameter = 1000
[  +0.000001] [calculatorModule - clean_exit] Result = 10000
```