

# Operating System

project 3

Department: 土木5B

ID: 0511330

Name: 劉紘華

Vedio Link: <https://youtu.be/-Z5-M9gFLBI>

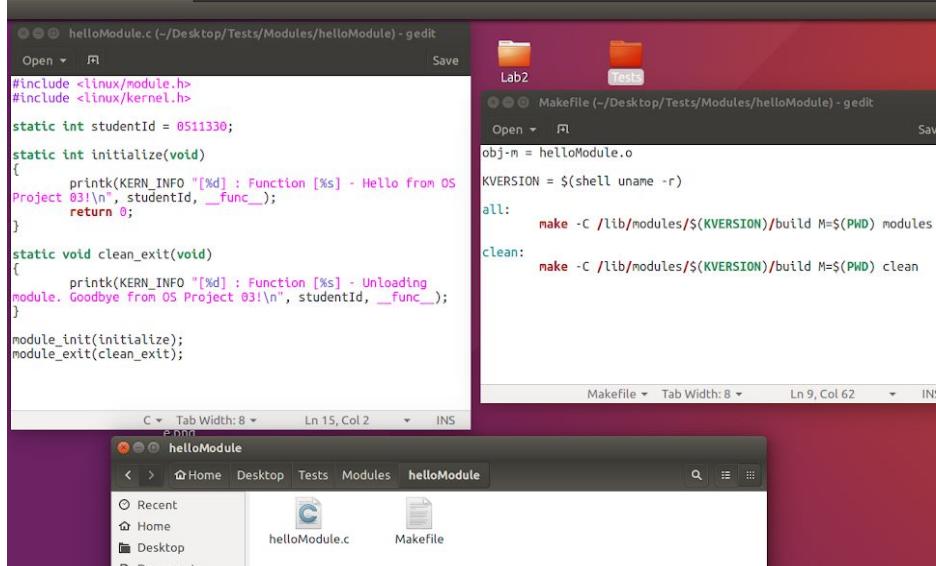
## **Objective:**

To help students to get familiar with Linux Kernel Modules. Students will learn the definition, usage and how to implement custom Linux Kernel Modules, and how to mount and unmount them.

## [Section 1-2] Load the module manually

### [Screenshot 1] helloModule code

The make file is used to tell the compiler the module needed to compile.



The screenshot shows two windows from the Gnome desktop environment. The left window is titled 'helloModule.c' and contains C code for a kernel module. The right window is titled 'Makefile' and contains a Makefile for building the module. Both files are open in Gedit text editors.

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

static int studentId = 0511330;

static int initialize(void)
{
    printk(KERN_INFO "[%d] : Function [%s] - Hello from OS Project 03!\n", studentId, __func__);
    return 0;
}

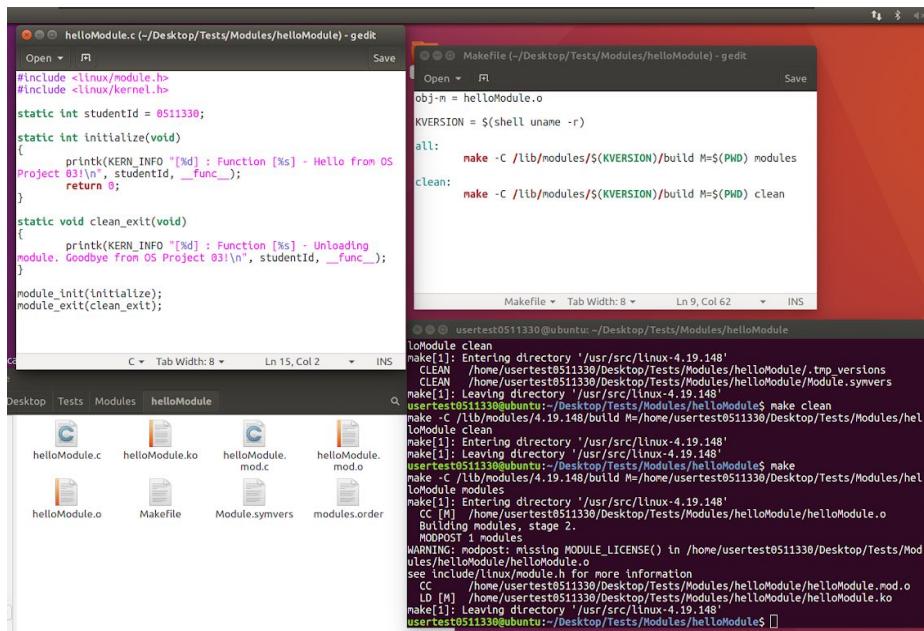
static void clean_exit(void)
{
    printk(KERN_INFO "[%d] : Function [%s] - Unloading module. Goodbye from OS Project 03!\n", studentId, __func__);
}

module_init(initialize);
module_exit(clean_exit);
```

```
obj-m = helloModule.o
KVERSION = $(shell uname -r)
all:
    make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean
```

### [Screenshot 2] make clean & make

Build the module by the command **make** under the module directory.  
We can clean the object code of the module through the command **make clean**.



The screenshot shows a terminal window and a file browser. The terminal window at the bottom shows the execution of 'make clean' followed by 'make'. The file browser above it shows the resulting module files: helloModule.ko, helloModule.mod.c, helloModule.mod.o, helloModule.o, Makefile, Module.symvers, and modules.order.

```
user@user0511330:~/Desktop/Tests/Modules/helloModule$ make clean
make[1]: Entering directory '/usr/src/linux-4.19.148'
CLEAN /home/user0511330/Desktop/Tests/Modules/helloModule/.tmp_versions
CLEAN /home/user0511330/Desktop/Tests/Modules/helloModule/Module.symvers
make[1]: Leaving directory '/usr/src/linux-4.19.148'
user@user0511330:~/Desktop/Tests/Modules/helloModule$ make
make -C /lib/modules/4.19.148/build M=/home/user0511330/Desktop/Tests/Modules/helloModule modules
make[1]: Entering directory '/usr/src/linux-4.19.148'
Building modules, stage 2.
MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/user0511330/Desktop/Tests/Modules/helloModule.o
see include/linux/module.h for more information
CC /home/user0511330/Desktop/Tests/Modules/helloModule/helloModule.mod.o
LD [M] /home/user0511330/Desktop/Tests/Modules/helloModule/helloModule.ko
make[1]: Leaving directory '/usr/src/linux-4.19.148'
user@user0511330:~/Desktop/Tests/Modules/helloModule$
```

## [Screenshot 3]

Use the command **dmesg -wH** to see the messages printed by our module in the **kernel ring buffer**.

The screenshot shows two terminal windows. The top window shows the command `sudo insmod helloModule.ko` being run, which successfully loads the module. The bottom window shows the kernel ring buffer output from `dmesg -wH`. It displays log entries from the `calculatorModule` and `helloModule` at various timestamps, including initialization parameters and cleanup messages.

```
user@user:~/Desktop/Tests/Modules/helloModule$ sudo insmod helloModule.ko
user@user:~/Desktop/Tests/Modules/helloModule$ 

user@user:~/Desktop/Tests/Modules/helloModule$ dmesg -wH
[+6.452895] [calculatorModule - initialize] =====
[+0.000001] [calculatorModule - initialize] Hello from calculatorModule!
[+0.000001] [calculatorModule - initialize] Operation = mul
[+0.000001] [calculatorModule - initialize] First parameter = -2
[+0.000001] [calculatorModule - initialize] Second parameter = 2000
[+0.000001] [calculatorModule - initialize] Result = -4000
[+0.000172] [calculatorModule - clean_exit] =====
[+0.000001] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[+0.000001] [calculatorModule - clean_exit] Operation = mul
[+0.000001] [calculatorModule - clean_exit] First parameter = -2
[+0.000001] [calculatorModule - clean_exit] Second parameter = 2000
[+0.000001] [calculatorModule - clean_exit] Result = -4000
[Dec18 00:09] helloModule: module license 'unspecified' taints kernel.
[+0.000001] Disabling lock debugging due to kernel taint
[+0.000607] [168664] : Function [initialize] - Hello from OS Project 03!
[Dec18 00:10] [168664] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
[+3.035807] [168664] : Function [initialize] - Hello from OS Project 03!
[Dec18 00:12] [168664] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
e. [+7.210845] [168664] : Function [initialize] - Hello from OS Project 03!
```

## [Screenshot 4]

To see all modules are loading currently , use the command **lsmod**.

To check whether the module is loaded successfully, use the command **lsmod | grep helloModule**

The screenshot shows a terminal window displaying the output of the `lsmod` command. The output is filtered using the `grep helloModule` command to show only the information related to the `helloModule`. The table lists the module name, size, and count of dependencies.

| Module           | Size   | Used by       |
|------------------|--------|---------------|
| helloModule      | 16384  | 0             |
| nls_utf8         | 16384  | 1             |
| iso9660          | 45056  | 2             |
| rfcomm           | 77824  | 0             |
| bnep             | 20480  | 2             |
| crct10dif_pclmul | 16384  | 0             |
| snd_ens1371      | 28672  | 2             |
| crc32_pclmul     | 16384  | 0             |
| snd_ac97_codec   | 131072 | 1 snd_ens1371 |
| gamenport        | 16384  | 1 snd_ens1371 |

## [Screenshot 5]

Unload the module by the command **rmmmod helloModule**, the messages printed by module shown in the **ring buffer**

The screenshot shows a terminal window with two parts. The top part displays the kernel ring buffer output, which includes messages from the calculatorModule and helloModule modules. The bottom part shows the terminal history with commands to insmod and rmmmod the helloModule.

```
[ +0.000172] [calculatorModule - clean_exit] =====
[ +0.000000] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[ +0.000000] [calculatorModule - clean_exit] Operation = mul
[ +0.000001] [calculatorModule - clean_exit] First parameter = -2
[ +0.000000] [calculatorModule - clean_exit] Second parameter = 2000
[ +0.000000] [calculatorModule - clean_exit] Result = -4000
[Dec18 00:09] helloModule: module license 'unspecified' taints kernel.
[ +0.000001] Disabling lock debugging due to kernel taint
[ +0.000607] [168664] : Function [initialize] - Hello from OS Project 03!
[Dec18 00:10] [168664] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
[ +3.035807] [168664] : Function [initialize] - Hello from OS Project 03!
[Dec18 00:12] [168664] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
[ +7.210845] [168664] : Function [initialize] - Hello from OS Project 03!
[Dec18 00:46] [168664] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
[Dec18 00:48] [168664] : Function [initialize] - Hello from OS Project 03!
[ +2.500726] [168664] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
user@test0511330@ubuntu: ~/Desktop/Tests/Modules/helloModule
user@test0511330@ubuntu:~/Desktop/Tests/Modules/helloModule$ sudo insmod helloModule.ko
user@test0511330@ubuntu:~/Desktop/Tests/Modules/helloModule$ sudo rmmmod helloModule
user@test0511330@ubuntu:~/Desktop/Tests/Modules/helloModule$
```

## [Screenshot 6]

When the module has been unloaded, **lsmod | grep helloModule** returns nothing, which implies our module unloads successfully.

The screenshot shows a terminal window with three commands: insmod, rmmmod, and lsmod | grep. The output shows the module was successfully loaded and then unloaded, with no trace left in the lsmod output.

```
user@test0511330@ubuntu: ~/Desktop/Tests/Modules/helloModule
user@test0511330@ubuntu:~/Desktop/Tests/Modules/helloModule$ sudo insmod helloModule.ko
user@test0511330@ubuntu:~/Desktop/Tests/Modules/helloModule$ sudo rmmmod helloModule
user@test0511330@ubuntu:~/Desktop/Tests/Modules/helloModule$ lsmod | grep helloModule
user@test0511330@ubuntu:~/Desktop/Tests/Modules/helloModule$
```

## [Section 1-2]

### [Screenshot 7]

The paramModule contains parameters, the module will react to different moves depending on different values of parameters.

The screenshot shows a dual-pane code editor. The left pane displays the C code for the kernel module, and the right pane shows a terminal window with the module's Makefile and a file browser.

**Kernel Module Source Code (paramModule.c):**

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

#define AUTHOR "BS11330 OS Student 2828 NCTU" //Change your name here
MODULE_LICENSE("GPL");
MODULE_AUTHOR(AUTHOR);

static char *kernelModuleName = "paramsModule"; //Change module's name when needed

static int studentId = 511330; // real studentId = 0511330, removed 0 for display purposes
module_param(studentId, int, 0644);
MODULE_PARM_DESC(studentId, "Parameter for student Id. (Leading zeros are omitted)");

static long secretValue = 987654321;
module_param(secretValue, long, 0644);
MODULE_PARM_DESC(secretValue, "Parameter for secret value.");

static char *charparameter = "Hello world! Project 02 -Example 02";
module_param(charparameter, charp, 0644);
MODULE_PARM_DESC(charparameter, "String inside module");

static int modifyValues = 0;
module_param(modifyValues, int, 0644);
MODULE_PARM_DESC(modifyValues, "Indicates if we must modify the original values or not.");

static int dummyStudentId = -1;
static long dummySecretValue = -2;

static int initialize(void) {
    if(modifyValues==1)
    {
        studentId = dummyStudentId;
        secretValue = dummySecretValue;
        charparameter = "This is a dummy message!";
    }
}

static void clean_exit(void) {
    if(modifyValues==1)
    {
        studentId = dummyStudentId;
        secretValue = dummySecretValue;
        charparameter = "Goodbye from OS Project 03!";
    }
}
```

**Makefile (paramsModule.mk):**

```
obj-m = paramsModule.o

KVERSION = $(shell uname -r)

all:
    make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules

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

**Terminal Window:**

```
user@test0511330:~/Desktop/Tests/Modules/paramsModule$ sudo insmod paramsModule.ko
[sudo] password for user@test0511330:
user@test0511330:~/Desktop/Tests/Modules/paramsModule$ sudo rmmod paramsModule.ko
user@test0511330:~/Desktop/Tests/Modules/paramsModule$
```

**File Browser:**

The file browser shows a directory named "paramsModule" containing a "Makefile" and a "paramsModule.c" file.

### [Screenshot 8]

If we are **not sending the parameters**, the default value will remain the same all the time when loading and unloading the module.

The screenshot shows a terminal window displaying kernel module logs and a separate log viewer window.

**Kernel Module Logs:**

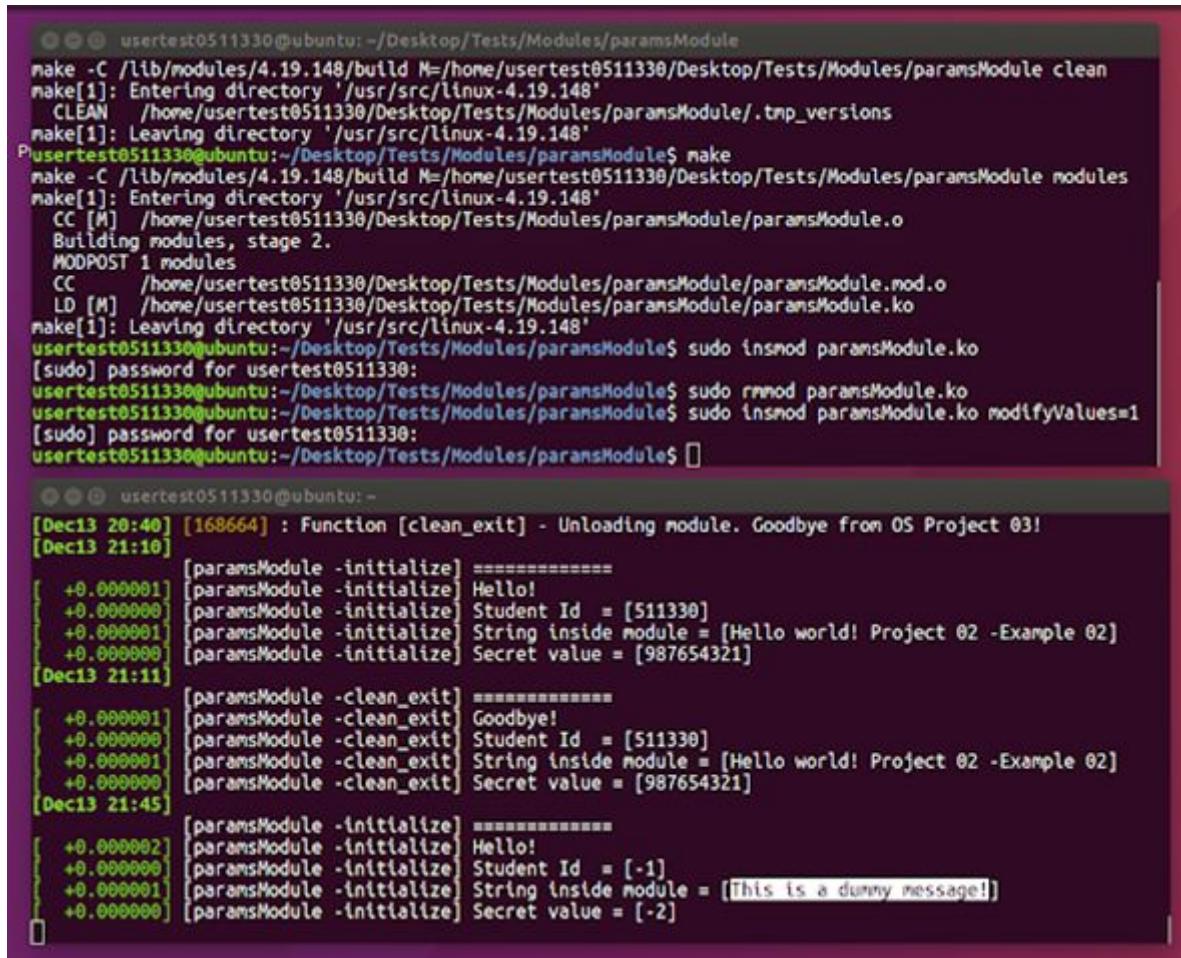
```
[+0.000001] raid6: using avx2x2 recovery algorithm
[+0.196518] xor: automatically using best checksumming function avx
[+0.182952] Btrfs loaded, crc32c=crc32c-Intel
[Dec13 20:19] helloModule: loading out-of-tree module taints kernel.
[+0.000016] helloModule: module license 'unspecified' taints kernel.
[+0.000000] Disabling lock debugging due to kernel taint
[+0.000045] helloModule: module verification failed: signature and/or required key missing - tainting kernel
[+0.000780] [168664] : Function [initialize] - Hello from OS Project 03!
[Dec13 20:40] [168664] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
[Dec13 21:10]
[paramsModule -initialize] *****
[+0.000001] [paramsModule -initialize] Hello!
[+0.000000] [paramsModule -initialize] Student Id = [511330]
[+0.000001] [paramsModule -initialize] String inside module = [Hello world! Project 02 -Example 02]
[+0.000000] [paramsModule -initialize] Secret value = [987654321]
[Dec13 21:11]
[paramsModule -clean_exit] *****
[+0.000001] [paramsModule -clean_exit] Goodbye!
[+0.000000] [paramsModule -clean_exit] Student Id = [511330]
[+0.000001] [paramsModule -clean_exit] String inside module = [Hello world! Project 02 -Example 02]
[+0.000000] [paramsModule -clean_exit] Secret value = [987654321]
```

**Log Viewer:**

The log viewer window displays the same kernel module logs as the terminal, showing the "Hello" message and "Goodbye" message.

## [Screenshot 9]

If we load the module with parameters (**modifyValues=1**), the module will receive it and react differently (e.g. **studentId** will be changed).



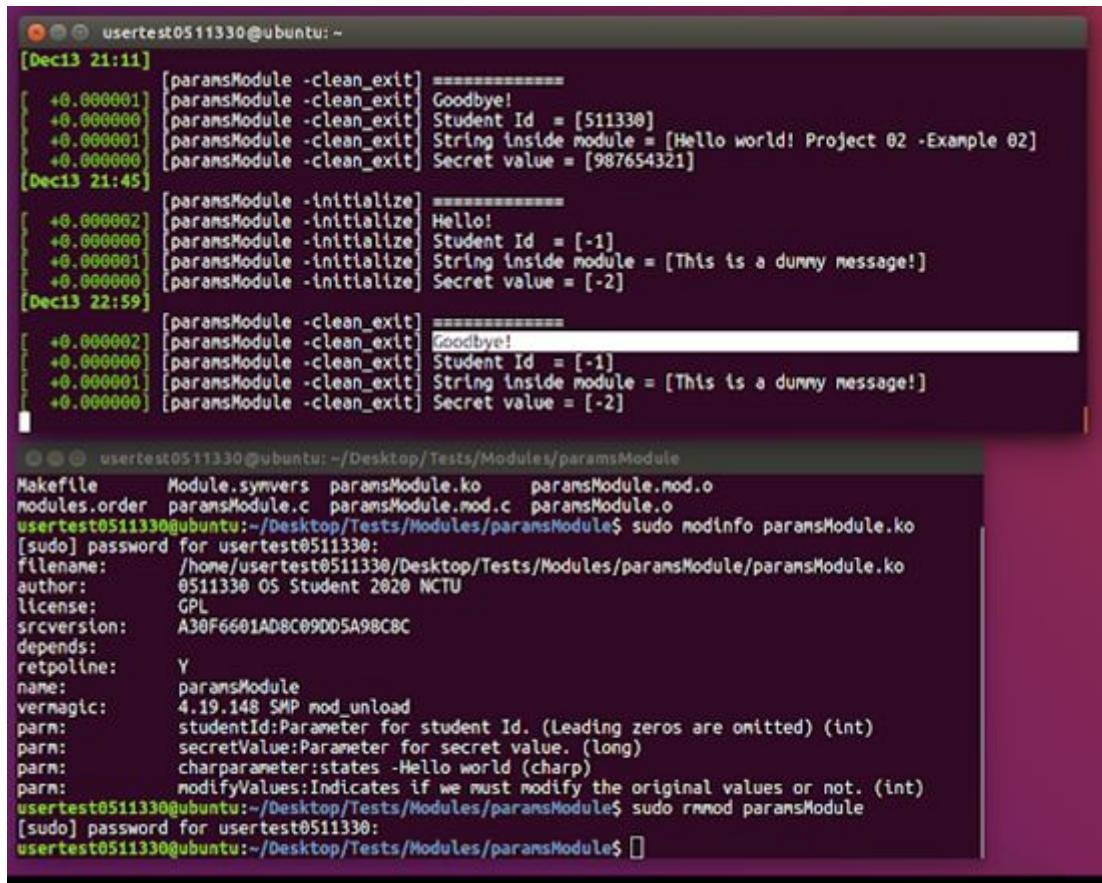
```
user@user:~/Desktop/Tests/Modules/paramsModule$ make -C /lib/modules/4.19.148/build M=/home/user/Desktop/Tests/Modules/paramsModule clean
make[1]: Entering directory '/usr/src/linux-4.19.148'
  CLEAN  /home/user/Desktop/Tests/Modules/paramsModule/.tmp_versions
make[1]: Leaving directory '/usr/src/linux-4.19.148'
user@user:~/Desktop/Tests/Modules/paramsModule$ make modules
make -C /lib/modules/4.19.148/build M=/home/user/Desktop/Tests/Modules/paramsModule modules
make[1]: Entering directory '/usr/src/linux-4.19.148'
  CC [M] /home/user/Desktop/Tests/Modules/paramsModule/paramsModule.o
Building modules, stage 2.
MODPOST 1 modules
  CC     /home/user/Desktop/Tests/Modules/paramsModule/paramsModule.mod.o
  LD [M] /home/user/Desktop/Tests/Modules/paramsModule/paramsModule.ko
make[1]: Leaving directory '/usr/src/linux-4.19.148'
user@user:~/Desktop/Tests/Modules/paramsModule$ sudo insmod paramsModule.ko
[sudo] password for user:
user@user:~/Desktop/Tests/Modules/paramsModule$ sudo rmmod paramsModule.ko
user@user:~/Desktop/Tests/Modules/paramsModule$ sudo insmod paramsModule.ko modifyValues=1
[sudo] password for user:
user@user:~/Desktop/Tests/Modules/paramsModule$ 

user@user:~-
[Dec13 20:40] [168664] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
[Dec13 21:10]
[+0.000001] [paramsModule -initialize] =====
[+0.000000] [paramsModule -initialize] Hello!
[+0.000001] [paramsModule -initialize] Student Id = [511330]
[+0.000001] [paramsModule -initialize] String inside module = [Hello world! Project 02 -Example 02]
[+0.000000] [paramsModule -initialize] Secret value = [987654321]
[Dec13 21:11]
[+0.000001] [paramsModule -clean_exit] =====
[+0.000000] [paramsModule -clean_exit] Goodbye!
[+0.000000] [paramsModule -clean_exit] Student Id = [511330]
[+0.000001] [paramsModule -clean_exit] String inside module = [Hello world! Project 02 -Example 02]
[+0.000000] [paramsModule -clean_exit] Secret value = [987654321]
[Dec13 21:45]
[+0.000002] [paramsModule -initialize] =====
[+0.000000] [paramsModule -initialize] Hello!
[+0.000000] [paramsModule -initialize] Student Id = [-1]
[+0.000001] [paramsModule -initialize] String inside module = [This is a dummy message!]
[+0.000000] [paramsModule -initialize] Secret value = [-2]
```

## [Screenshot 10]

With parameters triggering, we changed other parameters permanently. So the changed value will also be updated in the text file under the **/paramsModule/parameter** folder.

Use **modinfo paramsModule.ko** to show more information about the module.



The screenshot shows two terminal windows side-by-side. The left window displays kernel module logs with timestamps from Dec 13 21:11 to Dec 13 22:59. It shows several log entries for the 'paramsModule' module, specifically for the 'clean\_exit' function. These entries include messages like 'Goodbye!', 'Hello!', and 'String inside module = [Hello world! Project 02 -Example 02]'. The right window shows the output of the 'modinfo paramsModule.ko' command. It provides detailed information about the module, including its Makefile, author (0511330 OS Student 2020 NCTU), license (GPL), source version (A30F6601AD8C09DD5A98C8C), dependencies, and various parameters and their descriptions. The 'parm' section lists 'studentId', 'secretValue', 'charparameter', and 'modifyValues'. The 'modinfo' command was run with sudo privileges.

```
user@user:~$ modinfo paramsModule.ko
filename: /home/user/Desktop/Tests/Modules/paramsModule.ko
author: 0511330 OS Student 2020 NCTU
license: GPL
srcversion: A30F6601AD8C09DD5A98C8C
depends:
retpoline: Y
name: paramsModule
vermagic: 4.19.148 SMP mod_unload
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)
```

## [Screenshot 11]

We load the module again with more parameters:**studentId = 511330**, **secretValue = 8888**.

```
[sudo] password for userstest0511330:
userstest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ sudo insmod paramsModule.ko studentId=511330 secretValue=8888

userstest0511330@ubuntu:~/.config/autostart$ ./paramsModule
[+0.000002] [paramsModule -initialize] =====
[+0.000007] [paramsModule -initialize] Hello!
[+0.000001] [paramsModule -initialize] Student Id = [-1]
[+0.000001] [paramsModule -initialize] String inside module = [This is a dummy message!]
[+0.000001] [paramsModule -initialize] Secret value = [-2]
[Dec13 22:59]
[+0.000002] [paramsModule -clean_exit] =====
[+0.000007] [paramsModule -clean_exit] Goodbye!
[+0.000001] [paramsModule -clean_exit] Student Id = [-1]
[+0.000001] [paramsModule -clean_exit] String inside module = [This is a dummy message!]
[+0.000001] [paramsModule -clean_exit] Secret value = [-2]
[Dec13 23:06]
[+0.000001] [paramsModule -initialize] =====
[+0.000001] [paramsModule -initialize] Hello!
[+0.000001] [paramsModule -initialize] Student Id = [511330]
[+0.000001] [paramsModule -initialize] String inside module = [Hello world! Project 02 -Example 02]
[+0.000016] [paramsModule -initialize] Secret value = [8888]
[Dec13 23:06] FAT-fs (fd0): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
[Dec13 23:11]
```

## [Screenshot 12] modify secretValue to 7777

Further, we modify the parameter **secretValue** to **7777**, which is saved in the text file. By command:

**nano /sys/module/paramsModule/parameters/secretValue.**

```
userstest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ modinfo paramsModule.ko
filename: /home/userstest0511330/Desktop/Tests/Modules/paramsModule/paramsModule.ko
author: 0511330 OS Student 2020 NCTU
license: GPL
srcversion: A30F6601AD8C09D05A98C8C
depends:
retpoline: Y
name: paramsModule
vermagic: 4.19.148 SMP mod_unload
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)
userstest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ sudo rmmod paramsModule
[sudo] password for userstest0511330:
userstest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ sudo nano /sys/module/paramsModule/parameters/secretValue
```

### [Screenshot 13] remove module, secretValue = 7777

When we unload the module, the message shows that secreValue has been changed to 7777, which is shown in the ring buffer.

The screenshot shows a terminal window with two panes. The top pane displays a ring buffer log from a kernel module named 'paramsModule'. It shows the module initializing with 'Hello!', 'Student Id = [511330]', 'String inside module = [Hello world! Project 02 -Example 02]', and 'Secret value = [8888]'. It then receives a 'FAT-fs (fd0): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.' message. When it unloads, it says 'Goodbye!', 'Student Id = [511330]', 'String inside module = [Hello world! Project 02 -Example 02]', and 'Secret value = [7777]'. The bottom pane shows the command-line interface where the module is loaded with 'sudo modinfo paramsModule.ko' and then unloaded with 'sudo rmmod paramsModule'. Finally, 'sudo rmmod paramsModule' is run again.

```
+0.000001 [paramsModule -initialize] =====
+0.000001 [paramsModule -initialize] Hello!
+0.000001 [paramsModule -initialize] Student Id = [511330]
+0.000001 [paramsModule -initialize] String inside module = [Hello world! Project 02 -Example 02]
+0.000016 [paramsModule -initialize] Secret value = [8888]
[Dec13 23:08] FAT-fs (fd0): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
[Dec13 23:11]

+0.000002 [paramsModule -clean_exit] =====
+0.000001 [paramsModule -clean_exit] Goodbye!
+0.000001 [paramsModule -clean_exit] Student Id = [511330]
+0.000001 [paramsModule -clean_exit] String inside module = [Hello world! Project 02 -Example 02]
+0.000001 [paramsModule -clean_exit] Secret value = [7777]

useretest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule
useretest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ sudo modinfo paramsModule.ko
[sudo] password for useretest0511330:
filename: /home/useretest0511330/Desktop/Tests/Modules/paramsModule/paramsModule.ko
author: 0511330 OS Student 2020 NCTU
license: GPL
srcversion: A30F6601AD8C09DD5A98C8C
depends:
retpoline: Y
name: paramsModule
vermagic: 4.19.148 SMP mod_unload
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)
useretest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ sudo rmmod paramsModule
[sudo] password for useretest0511330:
useretest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ sudo nano /sys/module/paramsModule
/parameters/secretValue
useretest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ sudo rmmod paramsModule
```

### [Screenshot 14] modinfo showing “Unknown parameter” messages

The module will only receive the parameters, when we try to send a value to a variable which is not a parameter of the module, it shows an ignored message.

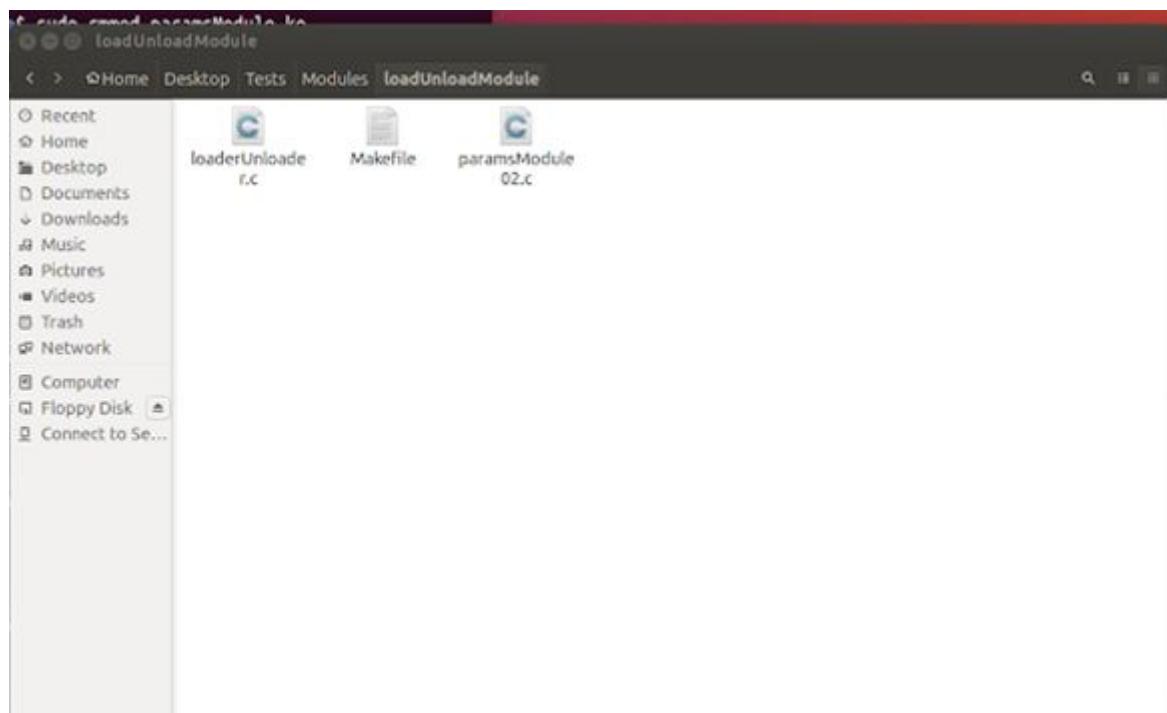
The screenshot shows a terminal window with two panes. The top pane shows the module loading with 'Hello!', 'Student Id = [511330]', 'String inside module = [Hello world! Project 02 -Example 02]', and 'Secret value = [8888]'. It then receives a 'FAT-fs (fd0): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.' message. When it unloads, it says 'Goodbye!', 'Student Id = [511330]', 'String inside module = [Hello world! Project 02 -Example 02]', and 'Secret value = [7777]'. The bottom pane shows the command-line interface where the module is loaded with 'sudo insmod paramsModule.ko dummyStudentId=9999'. It then shows an 'unknown parameter' message: 'paramsModule: unknown parameter 'dummyStudentId' ignored'. Finally, the module is unloaded with 'sudo rmmod paramsModule'.

```
useretest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ sudo insmod paramsModule.ko dummyStudentId=9999
[sudo] password for useretest0511330:
useretest0511330@ubuntu:~/Desktop/Tests/Modules/paramsModule$ 

useretest0511330@ubuntu:-
+0.000001 [paramsModule -initialize] Hello!
+0.000001 [paramsModule -initialize] Student Id = [511330]
+0.000001 [paramsModule -initialize] String inside module = [Hello world! Project 02 -Example 02]
+0.000016 [paramsModule -initialize] Secret value = [8888]
[Dec13 23:08] FAT-fs (fd0): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
[Dec13 23:11]

+0.000002 [paramsModule -clean_exit] =====
+0.000001 [paramsModule -clean_exit] Goodbye!
+0.000001 [paramsModule -clean_exit] Student Id = [511330]
+0.000001 [paramsModule -clean_exit] String inside module = [Hello world! Project 02 -Example 02]
+0.000001 [paramsModule -clean_exit] Secret value = [7777]
[Dec13 23:41] paramsModule: unknown parameter 'dummyStudentId' ignored
[+0.000119] [paramsModule -initialize] =====
[+0.000001] [paramsModule -initialize] Hello!
[+0.000001] [paramsModule -initialize] Student Id = [511330]
```

## [Screenshot 15]



## [Screenshot 16]

We use the **loaderUnloader** program to load the **paramsModule02** dynamically, with parameter “**studentId=511330**”.

We also define the system calls which load and unload the module as **init\_module**, **delete\_modue** by macro, to improve code readability.

```
loader.c (-/Desktop/Tests/Modules/loadUnloadModule) - gedit
Open ▾
/* Author: Heyward Liu
 * StudentID: 0511330

int main(int argc, char**argv) {
    printf("\nThis is a dynamic loader and unloader for a kernel module!\n");

    // Module information
    const char *moduleName = "paramsModule02.ko";
    const char *moduleNameNoExtension = "paramsModule02";
    const char *paramsNew = "studentId=511330"; // Use your StudentID without leading 0

    int fd, use_init;
    size_t image_size;
    struct statst;
    void *image;

    //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

    // At this point the module is mounted.
    // You can check it with $ lsmod | grep <name of module without extension>
    // You can access its variables in /sys/module/<name of module without extention>/parameters
    // WARNING: IF YOU MODIFY THE VARIABLES WITHOUT FOLLOWING THE CORRECT
    // DATATYPE YOUR MODULE WILL GET LOCKED AND YOU MUSST FIX THE VARIABLES
    // AND RESTART YOUR MACHINE

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

    //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");

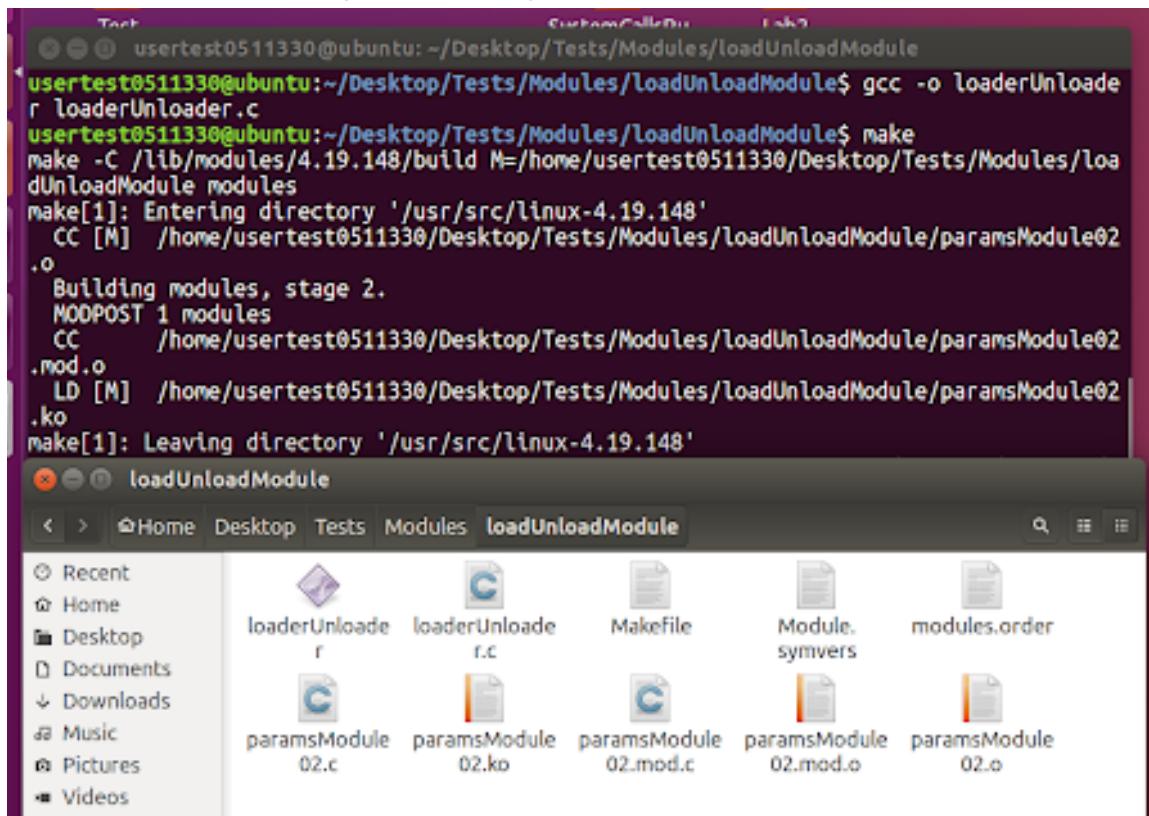
    free(image);
    //Section -Module unloading -END

    printf("Done!\n");
}

return 0;
```

## [Screenshot 17]

Using gcc to compile the **loaderUnloader** C file to the object code, and build it into kernel object code by command \$make.



The screenshot shows a Linux desktop environment with two windows open. The top window is a terminal window titled 'loadUnloadModule' containing the following command-line session:

```
user@user:~/Desktop/Tests/Modules/loadUnloadModule$ gcc -o loaderUnloader loaderUnloader.c
user@user:~/Desktop/Tests/Modules/loadUnloadModule$ make
make -C /lib/modules/4.19.148/build M=/home/user/Desktop/Tests/Modules/loadUnloadModule modules
make[1]: Entering directory '/usr/src/linux-4.19.148'
  CC [M] /home/user/Desktop/Tests/Modules/loadUnloadModule/paramsModule02.o
Building modules, stage 2.
MOPOST 1 modules
  CC      /home/user/Desktop/Tests/Modules/loadUnloadModule/paramsModule02.mod.o
  LD [M] /home/user/Desktop/Tests/Modules/loadUnloadModule/paramsModule02.ko
make[1]: Leaving directory '/usr/src/linux-4.19.148'
```

The bottom window is a file manager titled 'loadUnloadModule' showing the contents of the directory:

| File/Folder          | Description              |
|----------------------|--------------------------|
| loaderUnloader.c     | C source file            |
| loaderUnloader.c     | Object file (.c)         |
| Makefile             | Build configuration file |
| Module.symvers       | Symbol versioning file   |
| modules.order        | Module ordering file     |
| paramsModule02.c     | C source file            |
| paramsModule02.ko    | Kernel object file (.ko) |
| paramsModule02.mod.c | C source file            |
| paramsModule02.mod.o | Object file (.mod.o)     |
| paramsModule02.o     | Object file (.o)         |

## [Screenshot 18]

By running the **loaderUnloader** program, the module **paramsModule02** has been loaded with a received parameter: **studentId = 511330**. Now the loaderUnloader program is stuck in statement - **getchar()**. We can check the messages shown in **ring buffer** by the command **\$dmesg -wH**

The screenshot shows a terminal window with two main sections. The top section shows the command-line interface where a user named 'usertest0511330' is building a kernel module named 'loaderUnloader'. They run 'gcc -o loaderUnloader loaderUnloader.c', then 'make' which creates a module 'paramsModule02.ko'. They then run 'sudo ./loaderUnloader' which outputs a message about loading the module with parameters. The bottom section shows the kernel's ring buffer output, which includes timestamped messages from the module 'paramsModule02' during its initialization and cleanup processes. These messages include 'Hello!', 'Student Id = [511330]', 'String inside module = [Hello world! Project 02 -Example 03]', and 'Secret value = [987654321]'.

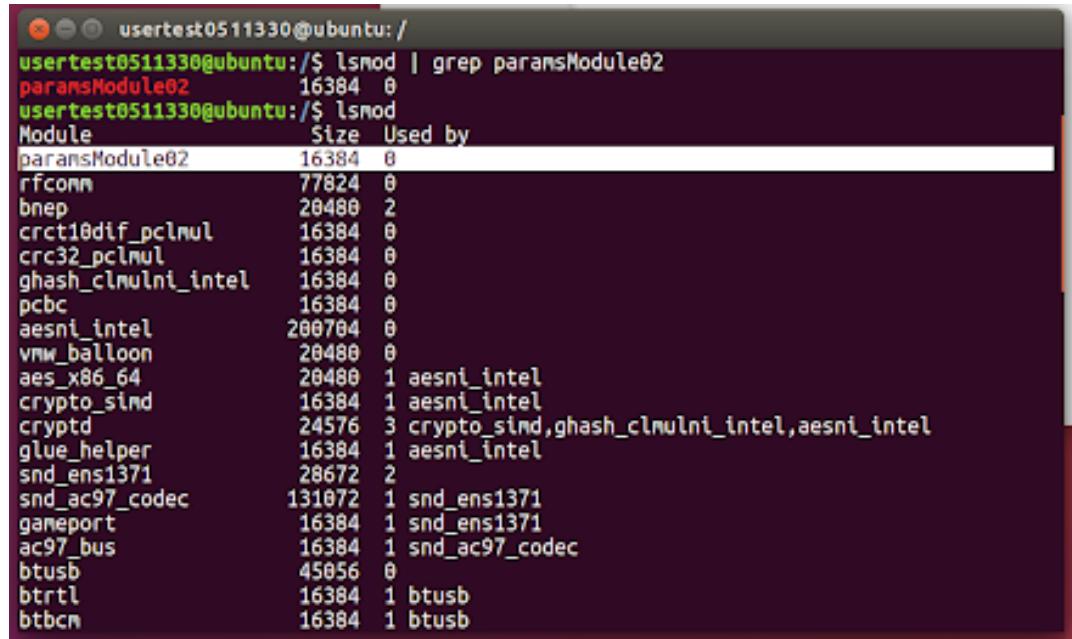
```
usertest0511330@ubuntu:~/Desktop/Tests/Modules/LoadUnloadModule$ gcc -o loaderUnloader loaderUnloader.c
usertest0511330@ubuntu:~/Desktop/Tests/Modules/LoadUnloadModule$ make
make -C /lib/modules/4.19.148/build M=/home/usertest0511330/Desktop/Tests/Modules/LoadUnloadModule modules
make[1]: Entering directory '/usr/src/linux-4.19.148'
  CC [M] /home/usertest0511330/Desktop/Tests/Modules/LoadUnloadModule/paramsModule02.mod.o
Building modules, stage 2.
MOPOST 1 modules
  CC      /home/usertest0511330/Desktop/Tests/Modules/LoadUnloadModule/paramsModule02.ko
make[1]: Leaving directory '/usr/src/linux-4.19.148'
usertest0511330@ubuntu:~/Desktop/Tests/Modules/LoadUnloadModule$ sudo ./loaderUnloader
This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=511330]...
Module is mounted!

[Press ENTER to continue]

[+0.000045] paramsModule02: module verification failed: signature and/or required key missing - tainting kernel
[+0.000369] [paramsModule02 -initialize] =====
[+0.000001] [paramsModule02 -initialize] Hello!
[+0.000000] [paramsModule02 -initialize] Student Id = [511330]
[+0.000001] [paramsModule02 -initialize] String inside module = [Hello world! Project 02 -Example 03]
[+0.000000] [paramsModule02 -initialize] Secret value = [987654321]
[Dec14 14:14] [paramsModule02 -clean_exit] =====
[+0.000001] [paramsModule02 -clean_exit] Goodbye!
[+0.000001] [paramsModule02 -clean_exit] Student Id = [511330]
[+0.000000] [paramsModule02 -clean_exit] String inside module = [Hello world! Project 02 -Example 03]
[+0.000001] [paramsModule02 -clean_exit] Secret value = [987654321]
[+26.720739] [paramsModule02 -initialize] =====
[+0.000002] [paramsModule02 -initialize] Hello!
[+0.000000] [paramsModule02 -initialize] Student Id = [511330]
[+0.000001] [paramsModule02 -initialize] String inside module = [Hello world! Project 02 -Example 03]
[+0.000000] [paramsModule02 -initialize] Secret value = [987654321]
```

## [Screenshot 19]

Further, we can also check it is loaded successfully by the commands **lsmod / lsmod | grep paramsModule02**.



```
user@user0511330:~$ lsmod | grep paramsModule02
paramsModule02 16384 0
user@user0511330:~$ lsmod
Module           Size  Used by
paramsModule02  16384  0
rfcomm          77824  0
bnep            20480  2
crc32_pclmul    16384  0
ghash_clmulni_intel 16384  0
pcbc             16384  0
aesni_intel     200704  0
vmw_balloon     20480  0
aes_x86_64      20480  1 aesni_intel
crypto_simd     16384  1 aesni_intel
cryptd          24576  3 crypto_simd,ghash_clmulni_intel,aesni_intel
glue_helper     16384  1 aesni_intel
snd_ens1371     28672  2
snd_ac97_codec   131072  1 snd_ens1371
gameport         16384  1 snd_ens1371
ac97_bus         16384  1 snd_ac97_codec
btusb            45056  0
btrtl            16384  1 btusb
btbcm            16384  1 btusb
```

## [Screenshot 20]

### [upper terminal]

When we proceed to continue the loadUnload file which is currently stuck in `getchar()`, our module will be unloaded.

### [middle terminal]

Messages in the **ring buffer** shows our module was cleaned up.

### [lower terminal]

Further, we check whether it was unloaded successfully by the command **lsmod | grep paramsModule02**, and we get nothing in return. It means our module is no longer in the module list.

The screenshot displays three terminal windows on a Linux system:

- Top Terminal:** Shows the compilation of the kernel module. The user runs `make` in the directory `/home/usertest0511330/Desktop/Tests/Modules/loadUnloadModule`. The output shows the compilation of `paramsModule02.c` into object files and a final module `paramsModule02.ko`.
- Middle Terminal:** Shows the execution of a dynamic loader. The user runs `sudo ./loaderUnloader`. The output indicates the module is being loaded with parameters `[studentId=511330]`. The module is successfully mounted.
- Bottom Terminal:** Shows the ring buffer messages. The module is initialized, printing "Hello!", "Student Id = [511330]", and "String inside module = [Hello world! Project 02 -Example 03]". It then exits, printing "Goodbye!".
- Bottom-most Terminal:** Shows the result of the `lsmod` command. There are no results for `paramsModule02`, confirming the module has been unloaded.

## [Screenshot 21] Implemented by loading the module

We figure out the result by loading **calculatorModule** in our calculator program, and send operand and operation to the module to get the result. All operations can be divided it into 3 major moves:

### 1.LoadModule

### 2.ReadResult

### 3.UnLoadModule

```
● ● ●

long addition (char* AllParams)
{
    long result = 0;
    LoadModule(AllParams);
    result = ReadResult();
    UnLoadModule();

    return result;
}

long subtraction (char* AllParams)
{
    long result = 0;
    LoadModule(AllParams);
    result = ReadResult();
    UnLoadModule();

    return result;
}

long multiplication (char* AllParams)
{
    long result = 0;
    LoadModule(AllParams);
    result = ReadResult();
    UnLoadModule();

    return result;
}
```

## [Screenshot 22] Parameters sended to the module

All parameters sended to the module are concatenated to a string called **AllParams**, which includes operation, input1 and input2.

```
//concat All parameters into a string  
  
sprintf(paramsNew, "operationParam=%s firstParam=%d secondParam=%d",  
operation, input1, input2);
```

## [Screenshot 22] Load & UnLoad

To our convenience and improving code readability, We define the function by macro. So we can load the **calculatorModule** by calling **init\_module** and unload it by **delete\_module**.

```
#define init_module(module_image, len, param_values)  
syscall(__NR_init_module, module_image, len, param_values)  
  
#define finit_module(fd, param_values, flags)  
syscall(__NR_finit_module, fd, paramvalues, flags)  
  
#define delete_module(name, flags) syscall(__NR_delete_module,  
name, flags)
```

Further, we pack all procedures when loading / unloading a module into functions called **LoadModule** / **UnLoadModule** in order to reuse the code.

```
void LoadModule(char* Params) {
    int fd, use_finit;
    size_t image_size;
    struct stat st;
    void *image;

    fd = open(moduleName, O_RDONLY);
    fstat(fd, &st);
    image_size = st.st_size;
    image = malloc(image_size);
    read(fd, image, image_size);
    if(init_module(image, image_size, Params) != 0)
    {
        perror("init_module");
    }
}
```

```
void UnLoadModule() {
    if(delete_module(moduleNameNoExtension,
O_NONBLOCK)!=0) {
        perror("delete_module");
    }
}
```

## [Screenshot 23] Read the Result

When the result is figured out by the module, it is saved into a text file called **resultParam** under **/sys/module/calculatorModule/parameters**. We simply use **fopen** to open the file and **fscanf** to read the result into our variable.



```
long ReadResult() {
    long result=0;
    FILE *file = fopen("/sys/module/calculatorModule/parameters/resultParam", "r");
    if(file) {
        fscanf(file, "%ld", &result);
    }
    return result;
}
```

## [Screenshot 24] Final result

```
usertest0511330@ubuntu: ~/Desktop/Tests/Modules/calculatorModule$ gcc -o calculator calculator.c
usertest0511330@ubuntu: ~/Desktop/Tests/Modules/calculatorModule$ sudo ./calculator
=====
Enter operation [sum - sub - mul - exit]: sum
Enter two operands (space separated): 100 -321
Operation: [sum] - Operands [100 -321] - Result:[-221]
=====
Enter operation [sum - sub - mul - exit]: sub 300 400
Enter two operands (space separated): Operation: [sub] - Operands [300 400] - Result:[-100]
=====
Enter operation [sum - sub - mul - exit]: mul
Enter two operands (space separated): 10 -3
Operation: [mul] - Operands [10 -3] - Result:[-30]
=====
Enter operation [sum - sub - mul - exit]: ■
```

```
[Dec18 01:52] usertest0511330@ubuntu: ~
[calculatorModule - initialize] =====
[+0.000003] [calculatorModule - initialize] Hello from calculatorModule!
[+0.000001] [calculatorModule - initialize] Operation = sum
[+0.000001] [calculatorModule - initialize] First parameter = 100
[+0.000001] [calculatorModule - initialize] Second parameter = -321
[+0.000001] [calculatorModule - initialize] Result = -221
[+0.000388]
[calculatorModule - clean_exit] =====
[+0.000001] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[+0.000001] [calculatorModule - clean_exit] Operation = sum
[+0.000001] [calculatorModule - clean_exit] First parameter = 100
[+0.000000] [calculatorModule - clean_exit] Second parameter = -321
[+0.000001] [calculatorModule - clean_exit] Result = -221
[+25.070554]
[calculatorModule - initialize] =====
[+0.000001] [calculatorModule - initialize] Hello from calculatorModule!
[+0.000000] [calculatorModule - initialize] Operation = sub
[+0.000001] [calculatorModule - initialize] First parameter = 300
[+0.000000] [calculatorModule - initialize] Second parameter = 400
[+0.000000] [calculatorModule - initialize] Result = -100
[+0.000151]
[calculatorModule - clean_exit] =====
[+0.000001] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[+0.000000] [calculatorModule - clean_exit] Operation = sub
[+0.000000] [calculatorModule - clean_exit] First parameter = 300
[+0.000000] [calculatorModule - clean_exit] Second parameter = 400
[+0.000001] [calculatorModule - clean_exit] Result = -100
[+11.422502]
[calculatorModule - initialize] =====
[+0.000001] [calculatorModule - initialize] Hello from calculatorModule!
[+0.000000] [calculatorModule - initialize] Operation = mul
[+0.000000] [calculatorModule - initialize] First parameter = 10
[+0.000001] [calculatorModule - initialize] Second parameter = -3
[+0.000000] [calculatorModule - initialize] Result = -30
[+0.000166]
[calculatorModule - clean_exit] =====
[+0.000000] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[+0.000001] [calculatorModule - clean_exit] Operation = mul
[+0.000000] [calculatorModule - clean_exit] First parameter = 10
[+0.000000] [calculatorModule - clean_exit] Second parameter = -3
[+0.000001] [calculatorModule - clean_exit] Result = -30
```

## [Quetions]

1. What is a static / dynamic kernel module ?  
What is the other name of a dynamic kernel module?  
What are the differences between system calls and dynamic kernel modules?
  - (i) **static modules** are compiled as part of the base kernel and it is available at any time.  
**dynamic** modules are compiled as modules separately and loaded based on user demand.
  - (ii) **Loadable Kernel Modules**
  - (iii) (a) **System calls** will be in the kernel all the time as the API to serve all user programs; **Loadable Kernel Modules** will load only when a specific needs.  
(b) **System calls** need to be compiled as part of kernel, **LKM** can be compiled at any time.  
(c) We will load **LKM** into the kernel by a **system call**.
2. Why does adding a system call require kernel recompilation, while adding a kernel module does not?

System call is the part of the kernel as APIs to provide OS services to user programs. Fundamentally, system call is treated as kernel.  
Kernel module is more like a user program. It's only loaded into the kernel when we need it. It makes the kernel more flexible.
3. What are the commands insmod, rmmod and modinfo for?

**insmod dummyModule.ko** means insert the dummyModule.  
**rmmod dummyModule.ko** means remove the dummyModule.  
**modinfo dummyModule.ko** shows the information of dummyModule.

4. Write the usage of the following commands:

- a. **module\_init** is a **macro** that defines which function is to be called at insertion time.
- b. **module\_exit** is a **macro** that defines which function to be called at removal time.
- c. **MODULE\_LICENSE** is the license of the module, a **macro** which allows LKM to declare their license to the world.
- d. **module\_param** is a **macro** that indicates the variables are module parameters and its data type.
- e. **MODULE\_PARAM\_DESC** is a **macro** that is used to document arguments that the module can take, and will be written into **modinfo**.

5. What do the following commands mean

- a. **cat <filename>** is used to print the whole file on the terminal.
- b. **ls -l** print the file-access date of each file of the whole directory.
- c. **dmesg** shows kernel log, and **-wH** means that **keep reading** a new log in **human readable form**.
- d. **lsmod** will list all modules that have been loaded currently.
- e. **lsmod | grep <module name>** return the specific module state which has been loaded currently, and return nothing if the module has not been loaded.

6. What does 0644 mean in module\_param()?

It refers to permission of accessing the parameters file, 0644 means modules allow us to access the parameter files freely.

7. What happens if the initialization function of the module returns -1? What type error do you get?

When initialization returns -1, it means that loading the module fails. We will receive **Operation not permitted** from the terminal. It's because insert the module requiring the root permission.

8. Two of the variables don't show in the **modinfo** in section 1.3, why?

Because only parameters will show on the **modinfo**, two of them are not parameters.

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

To place code, parameters and other stuff of modules which have been loaded into the kernel currently.

10. What is charp in section 1.3?

You will use a type of **charp** to define a module parameter that takes a string.