

Introduction to Operating Systems
Project 3: Dynamically-Loadable Kernel Modules (DLKMs)
0616087 賴沛冠

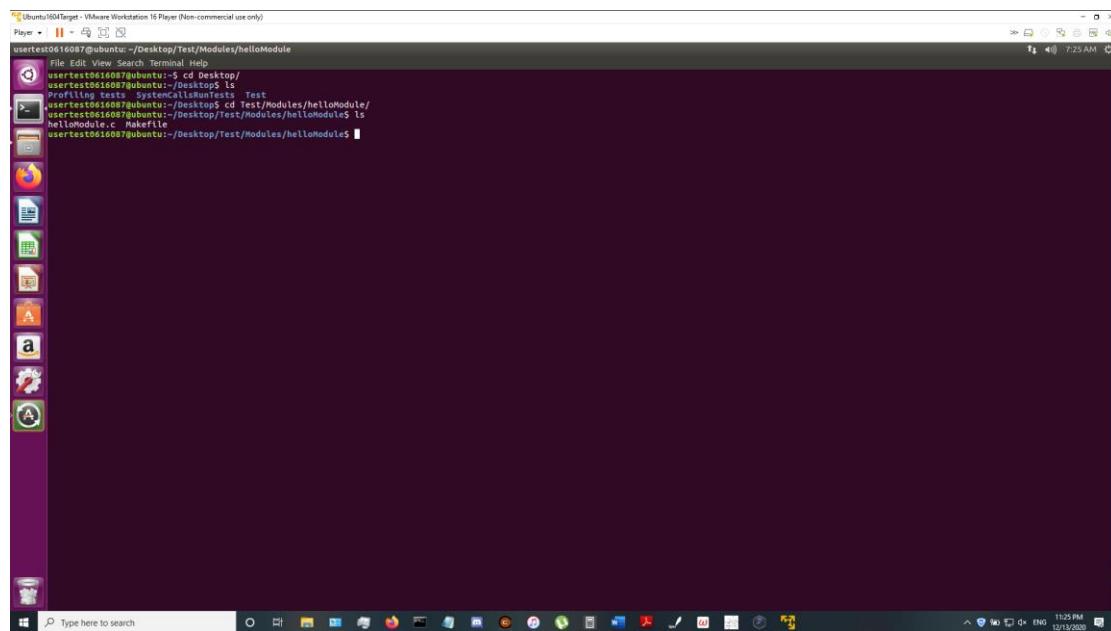
1. Demo video link

<https://youtu.be/1pgu1upz2Aw>

2. Report

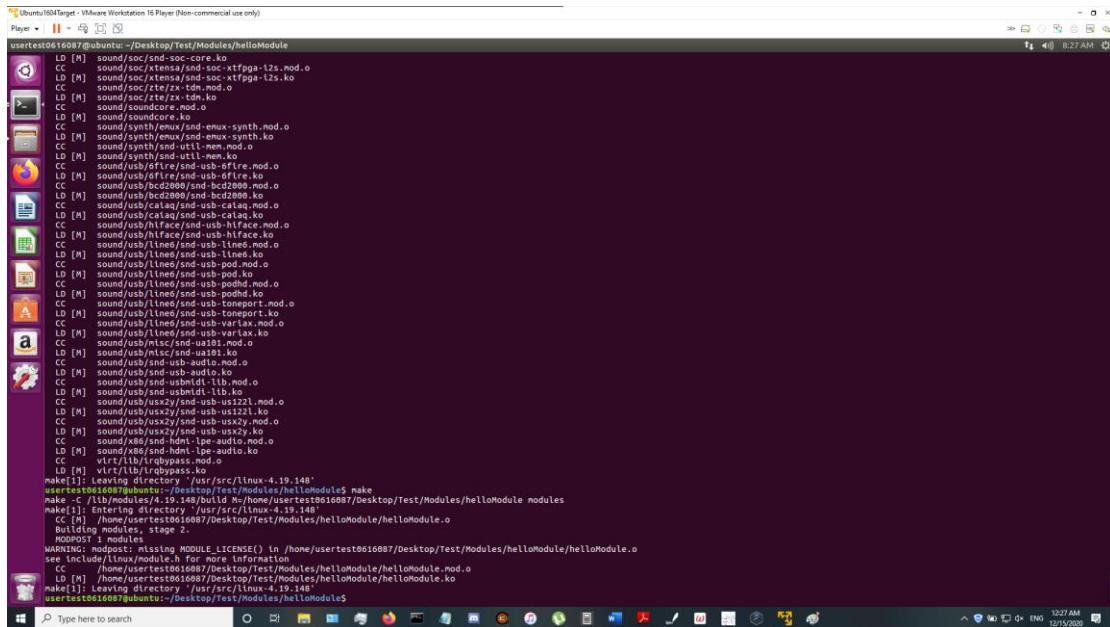
[Screenshot #1: Create a screenshot showing these two files and the folder where you save them.]

As you can see, I created a folder /helloModule under /Desktop/Test/Module, and created two files: helloModule.c and Makefile.



[Screenshot #2: Create a screenshot showing the result of the make process]

The screenshot shows the result of \$make.

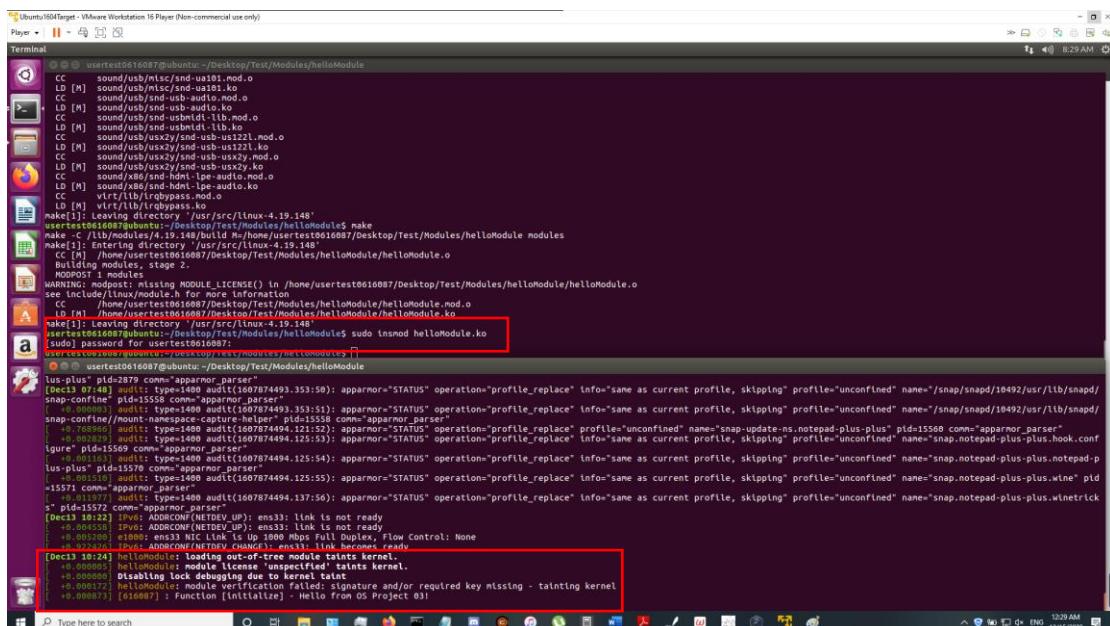


```
user@test0616087:~/Desktop/Test/Modules/helloModule
make[1]: Leaving directory '/usr/src/linux-4.19.148'
make[1]: Entering directory '/home/user/test0616087/Desktop/Test/Modules/helloModule'
  CC [M] /lib/modules/4.19.148/build M=/home/user/test0616087/Desktop/Test/Modules/helloModule modules
Building modules, stage 2.
MODPOST 1 modules
WARNING: module missing MODULE_LICENSE() in /home/user/test0616087/Desktop/Test/Modules/helloModule.ko
see include/linux/module.h for more information
  CC [M] /home/user/test0616087/Desktop/Test/Modules/helloModule.ko
make[1]: Leaving directory '/home/user/test0616087/Desktop/Test/Modules/helloModule'
user@test0616087:~/Desktop/Test/Modules/helloModule$
```

[Screenshot #3: Create a screenshot showing both Terminal 1 and 2 when you load the module]

The upside is Terminal 1, and it has installed a module “helloModule.ko” created by making the makefile, with the command \$sudo insmod helloModule.ko.

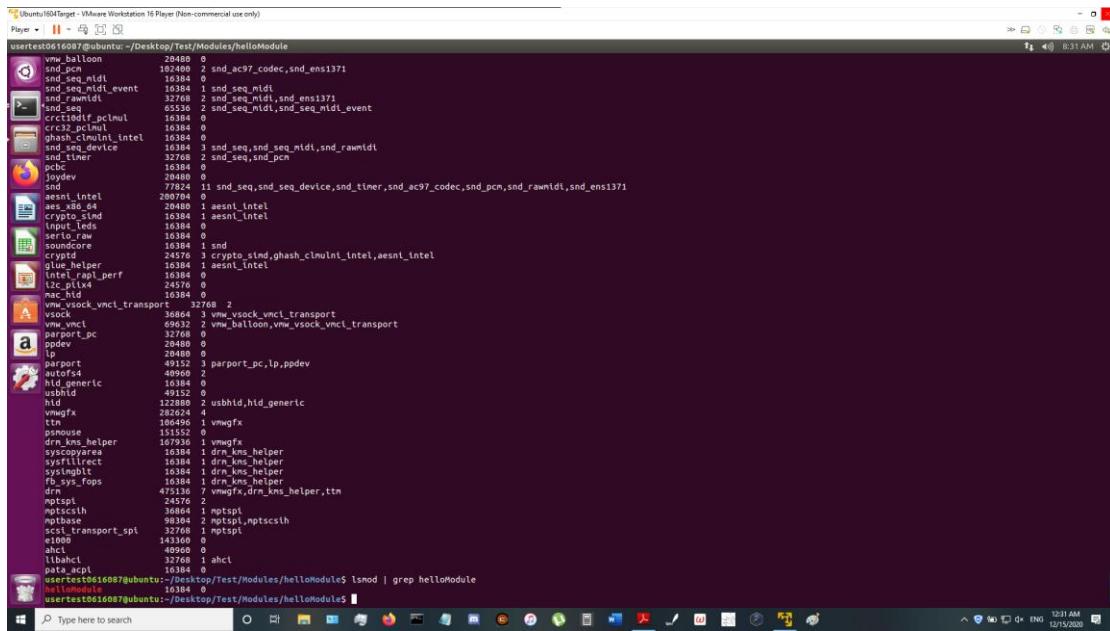
The downside is Terminal 2, it uses \$dmesg -wH to print the kernel ring buffer, as you can see there is a sentence “Hello from OS Project 03” means we successfully make the module.



```
user@test0616087:~/Desktop/Test/Modules/helloModule
make[1]: Leaving directory '/usr/src/linux-4.19.148'
user@test0616087:~/Desktop/Test/Modules/helloModule$ make
make[1]: Entering directory '/home/user/test0616087/Desktop/Test/Modules/helloModule'
  CC [M] /lib/modules/4.19.148/build M=/home/user/test0616087/Desktop/Test/Modules/helloModule modules
Building modules, stage 2.
MODPOST 1 modules
WARNING: module missing MODULE_LICENSE() in /home/user/test0616087/Desktop/Test/Modules/helloModule.ko
see include/linux/module.h for more information
  CC [M] /home/user/test0616087/Desktop/Test/Modules/helloModule.ko
make[1]: Leaving directory '/home/user/test0616087/Desktop/Test/Modules/helloModule'
user@test0616087:~/Desktop/Test/Modules/helloModule$ sudo insmod helloModule.ko
[sudo] password for usertest0616087:
user@test0616087:~/Desktop/Test/Modules/helloModule
[ 0.000000] pid=2879 comm='apparmor_parser'
[Dec13 07:48] audit: type=1400 audit(1607874493.353:5): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="/snap/snapd/10492/usr/lib/snapd/snap-confine"
[Dec13 07:48] audit: type=1400 audit(1607874493.353:5): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="/snap/snapd/10492/usr/lib/snapd/snap-confine/mount Namespace/capture-helper" pid=15589 comm='apparmor_parser'
[ 0.769360] audit: type=1400 audit(1607874494.121:52): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="snap-update-notepad-plus-plus" pid=15580 comm='apparmor_parser'
[ 0.769360] audit: type=1400 audit(1607874494.121:52): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="snap.update-notepad-plus-plus.hook.config" pid=15569 comm='apparmor_parser'
[ 0.801033] audit: type=1400 audit(1607874494.125:54): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="snap.notepad-plus-plus.notepad-pid=15571" pid=15571 comm='apparmor_parser'
[ 0.801033] audit: type=1400 audit(1607874494.125:55): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="snap.notepad-plus-plus.wine" pid=15571 comm='apparmor_parser'
[ 0.801033] audit: type=1400 audit(1607874494.125:56): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="snap.notepad-plus-plus.winetrack" pid=15572 comm='apparmor_parser'
[Dec13 10:22] IPv6: ADDRCONF(NETDEV_UP): ens3: link is not ready
[ 0.804538] IPv6: ADDRCONF(NETDEV_UP): ens3: link is not ready
[ 0.804538] IPv6: ADDRCONF(NETDEV_UP): ens3: link becomes ready
[ 0.804538] IPv6: ADDRCONF(NETDEV_CHANGE): ens3: link becomes ready
[Dec13 10:24] helloModule: loading out-of-tree module taints kernel.
[Dec13 10:24] helloModule: module verification failed: signature and/or required key missing - tainting kernel.
[ 0.000000] Disabling lock debugging due to kernel taint
[ 0.000172] helloModule: module verification failed: signature and/or required key missing - tainting kernel.
[ 0.000872] (010007) : Function [initialize] - Hello from OS Project 03!
```

[Screenshot #4: Create a screenshot showing the list of loaded modules from step 8 and 9]

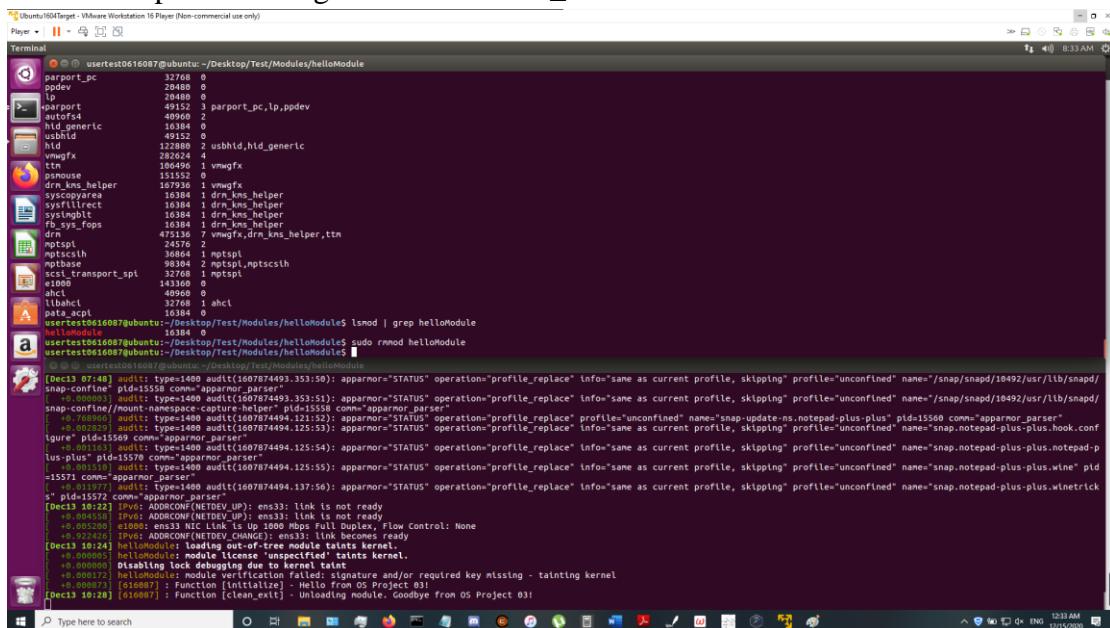
By running \$ lsmod or \$ lsmod | grep helloModule in Terminal 1, we can see that there is a module called helloModule now in the modules list.



```
user@test0616087:~$ lsmod
[...]
helloModule 123456 0
user@test0616087:~$ lsmod | grep helloModule
```

[Screenshot #5: Create a screenshot showing both terminals when unloading the module]

By running \$ sudo rmmod helloModule, you can see Terminal 1 worked, and Terminal 2 prints message inside the clean_exit function of the helloModule.c file.



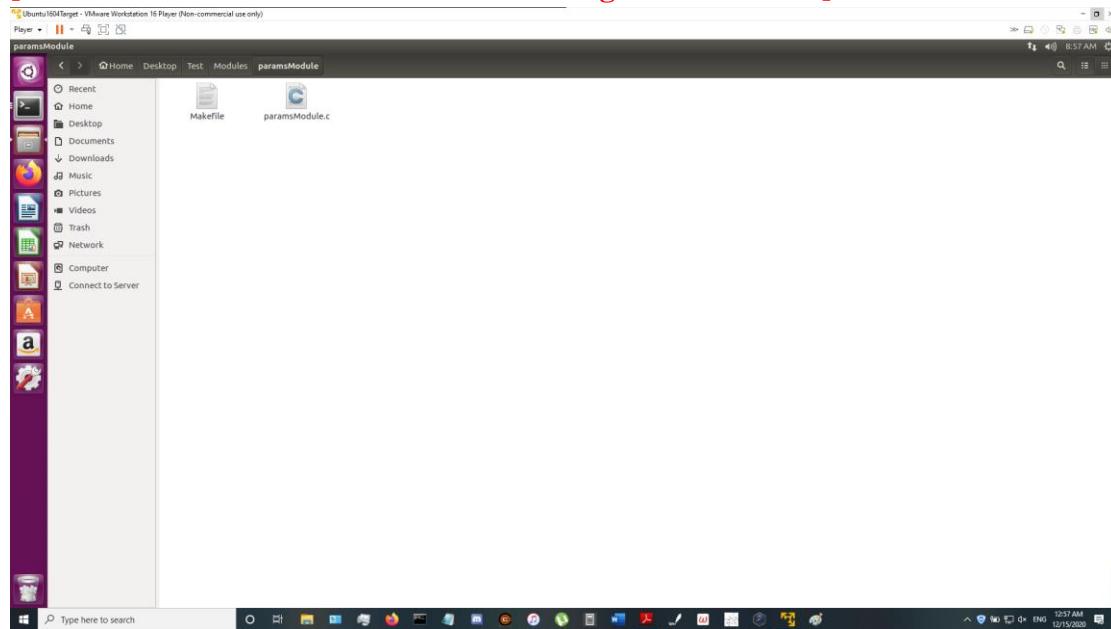
```
user@test0616087:~$ lsmod | grep helloModule
user@test0616087:~$ sudo rmmod helloModule
user@test0616087:~$
```

```
[Dec13 07:48] audit: type=1400 audit(160874493.353:5): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="/snap/snapd/10492/usr/lib/snapd/snap-confine"
[Dec13 07:48] audit: type=1400 audit(160874493.353:5): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="/snap/snapd/10492/usr/lib/snapd/snap-confine/mount-namespaces-capture-helper"
[Dec13 07:48] audit: type=1400 audit(160874493.353:5): apparmor="STATUS" operation="profile_replace" info="unconfined" name="snap-update-ns-notepad-plus-plus" pid=15500 comm="apparmor_parser"
[+0.00285] audit: type=1400 audit(160874494.125:32): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="snap.notepad-plus-plus.hook.config" pid=15569 comm="apparmor_parser"
[Dec13 07:48] audit: type=1400 audit(160874494.125:34): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="snap.notepad-plus-plus.notepad-plus-plus.wine" pid=15571 comm="apparmor_parser"
[Dec13 07:48] audit: type=1400 audit(160874494.137:56): apparmor="STATUS" operation="profile_replace" info="same as current profile, skipping" profile="unconfined" name="snap.notepad-plus-plus.winetricks" pid=15572 comm="apparmor_parser"
[Dec13 10:22] IPv6: ADDRCONF(NETDEV_UP): ens3: link is not ready
[Dec13 10:22] IPv6: ADDRCONF(NETDEV_CHANGE): ens3: link becomes ready
[+0.00285] si000: ens3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[+0.92742] IPv6: ADDRCONF(NETDEV_CHANGE): ens3: link becomes ready
[Dec13 10:24] helloModule: warning: lock-free tree is not yet available
[Dec13 10:24] helloModule: module is loaded in the specified hosts kernel.
[+0.00000] Disabling lock debugging due to kernel taint
[+0.00017] helloModule: module verification failed: signature and/or required key missing - tainting kernel
[Dec13 10:26] [0000000000000000] : Function [initialize] - Hello from OS Project 03!
[Dec13 10:26] [0000000000000000] : Function [clean_exit] - Unloading module. Goodbye From OS Project 03!
```

[Screenshot #6: Create a screenshot showing the result of when searching for the module after it was unloaded].

When searching for the module after it was unloaded, we can't see anything because here is no helloModule mounted.

[Screenshot #7: Create a screenshot showing these two files.]



[Screenshot #8: Create a screenshot showing the values you obtain when mounting and unmounting the module, and what commands you ran to do so.]

I use \$make clean and \$make to re-build my module, and \$ sudo insmod paramsModule.ko to mount the three values(studentId (616087), secretValue (987654321), and charparameter (“Hello world! Project 02 – Example 02”)). Finally, use \$ sudo rmmod paramsModule.ko to unmounts the module.

[Screenshot #9-10: Create two screenshots explaining steps 7 and 8.]

#9

In step 7 we modified the value of modifyValues to 1 by using **\$ sudo insmod paramsModule.ko modifyValues=1**, and check weather it changes on Terminal 2.

Ubuntu16.04Target - VMware Workstation 16 Player (Non-commercial use only)

Player | || | ☰ 9:16 AM

Terminal

```
user@user:~/Desktop/Test/Modules/paramsModule$
```

author: perry
license: GPL
sccversion: A995669ABA544B7929CD90
retpline: Y
name: paramsModule
version: 4.2.0-15 SMP mod_unload
param: studentId=Parameter for student Id. (Leading zeros are omitted) (int)
param: secretValue=Parameter for secret value. (long)
param: charparameterStates = How to work. (char)
modparam: if you want to modify the original values or not. (int)

```
user@user:~/Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko modifyValues=1
insmod: ERROR: could not insert module paramsModule.ko: File exists
user@user:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule.ko
rmmod: ERROR: Module modifyValues is not currently loaded
user@user:~/Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko modifyValues=1
user@user:~/Desktop/Test/Modules/paramsModule$
```

[Doc14 09:10]

```
[paramsModule - initialize] ****
[+0.00000] [paramsModule - initialize] Hello!
[+0.00000] [paramsModule - initialize] Student Id = [-1]
[+0.00000] [paramsModule - initialize] String inside module = [This is a dummy message!]
[+0.00000] [paramsModule - initialize] Secret value = [-2]
```

[Doc14 09:15]

```
[paramsModule - clean_exit] ****
[+0.00000] [paramsModule - clean_exit] Goodbye!
[+0.00000] [paramsModule - clean_exit] Student Id = [-1]
[+0.00000] [paramsModule - clean_exit] String inside module = [This is a dummy message!]
[+0.00000] [paramsModule - clean_exit] Secret value = [-2]
```

[Doc14 09:16]

```
[paramsModule - initialize] ****
[+0.00000] [paramsModule - initialize] Hello!
[+0.00000] [paramsModule - initialize] Student Id = [-1]
[+0.00000] [paramsModule - initialize] String inside module = [This is a dummy message!]
[+0.00000] [paramsModule - initialize] Secret value = [-2]
```

Type here to search

#10

In step 8 we use `$ sudo modinfo paramsModule.ko` to check the information inside the module. It also shows the datatype of the variables (int, long, charp).

Then unmount the module in Terminal 1, and watch that the variables when the clean_exit function is executed in Terminal 2.

```
Ubuntu/16.04Target - VMware Workstation 15 Player (Non-commercial use only)
Player - II - ☰
Terminal user@user0616087:~/Desktop/Test/Modules/paramsModule
llicense: GPL
srcversion: A9985669A5A44B7929C096
depends:
retpoline: Y
name: paramsmodule
vermagic: 4.19.148 SMP mod_unload
parm: studentIdParameterForStudentId: (Leading zeros are omitted) (int)
parm: secretValue:Parameter for secret value. (long)
parm: charparameter:states Hello world (char)
parm: modifyValues:Indicates if we must modify the original values or not. (int)
user@test0616087:~/Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko modifyValues=1
insmod: ERROR: could not insert module paramsModule.ko: File exists
user@test0616087:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule.ko modifyValues=1
user@test0616087:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule
user@test0616087:~/Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko modifyValues=1
user@test0616087:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule
user@test0616087:~/Desktop/Test/Modules/paramsModule$ 
[Dec14 09:15] [paramsModule - clean_exit] *****
[+0.000001] [paramsModule - clean_exit] Goodbye!
[+0.000000] [paramsModule - clean_exit] Student Id = []
[+0.000001] [paramsModule - clean_exit] String Inside module = [This is a dummy message!]
[+0.000000] [paramsModule - clean_exit] Secret value = [-2]
[Dec14 09:16] [paramsModule - initialize] *****
[+0.000001] [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]
[Dec14 09:17] [paramsModule - clean_exit] *****
[+0.000001] [paramsModule - clean_exit] Goodbye!
[+0.000000] [paramsModule - clean_exit] Student Id = [-1]
[+0.000001] [paramsModule - clean_exit] String Inside module = [This is a dummy message!]
user@test0616087:~/Desktop/Test/Modules/paramsModule
user@test0616087:~/Desktop/Test/Modules/paramsModule$ 
user@test0616087:~/Desktop/Test/Modules/paramsModule$ cd paramsModule/
user@test0616087:~/Desktop/Test/Modules/paramsModule$ sudo modinfo paramsModule.ko
[sudo] password for user@test0616087:
filename: /home/user/test0616087/Desktop/Test/Modules/paramsModule/paramsModule.ko
author: perfry
license: GPL
srcversion: A9985669A5A44B7929C096
depends:
retpoline: Y
name: paramsmodule
vermagic: 4.19.148 SMP mod_unload
parm: studentIdParameterForStudentId: (Leading zeros are omitted) (int)
parm: secretValue:Parameter for secret value. (long)
parm: charparameter:states Hello world (char)
parm: modifyValues:Indicates if we must modify the original values or not. (int)
user@test0616087:~/Desktop/Test/Modules/paramsModule$ 
```

[Screenshot #11-12-13: Create three screenshots explaining steps 9, 10 and 11.]

#11

In step 9 we mount again the module in Terminal 1 with values studentId=[616087] and secretValue=8888. Terminal 2 displays the messages with those values.

Ubuntu/64Target - VMware Workstation 16 Player (Non-commercial use only)

Play

Ubuntu Desktop

```
srcversion: A9985669ABA544B7929C090
depends:
name: paramsModule
versionline: V
vermagic: 4.19.14 SMP mod_unload
parm: studentId=616087 For student Id. (Leading zeros are omitted) (int)
parm: secretValue=8888 For secret value. (long)
parm: modifyValues=0 indicates if we must modify the original values or not. (int)
parm: modulePath=/lib/modules/4.19.14-041914-generic/extra/paramsModule.ko
insmod: ERROR: could not insert module paramsModule.ko: File exists
usersert0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule.ko
modifyValues=1
rmmod: Module paramsModule not currently loaded
usersert0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko
modifyValues=1
usersert0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule
usersert0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko studentId=616087 secretValue=8888
usersert0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ [
```

[Dec14 09:16] [paramsModule - initialize] *****
[+0.000001] [paramsModule - initialize] Hello world
[+0.000001] [paramsModule - initialize] Student Id = [-1]
[+0.000001] [paramsModule - initialize] String Inside module = [This is a dummy message!]
[+0.000000] [paramsModule - initialize] Secret value = [-2]
[Dec14 09:17]
[paramsModule - clean_exit] *****
[+0.000001] [paramsModule - clean_exit] Goodbye!
[+0.000000] [paramsModule - clean_exit] Student Id = [-1]
[+0.000001] [paramsModule - clean_exit] String Inside module = [This is a dummy message!]
[+0.000000] [paramsModule - clean_exit] Secret value = [-2]
[Dec14 09:18]
[paramsModule - initialize] *****
[+0.000001] [paramsModule - initialize] Hello world
[+0.000000] [paramsModule - initialize] Student Id = [616087]
[+0.000001] [paramsModule - initialize] String Inside module = [Hello world! Project 02 - Example 02]
[+0.000018] [paramsModule - initialize] Secret value = [8888]

#12

In step 10 we can see Linux manages module variables as files. When the module is mounted, we can find them in /sys/module/paramsModule/parameters

Ans in Terminal 3, we can modify the value of those variables with \$ sudo vim

/sys/module/paramsModule/parameters/secretValue and make secretValue=7777.

The screenshot shows a terminal window titled "Ubuntu/64Target - VMware Workstation 16 Player (Non-commercial use only)". The terminal output is as follows:

```
root@user0616087:~/Desktop/Test/Modules/paramsModule
lsmod | grep paramsModule
lsmod: ERROR! could not insert module paramsModule.ko: File exists
rmmod: ERROR! module paramsModule is not currently loaded
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko modifyValues=1
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ ls /sys/module/paramsModule/parameters/
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ ls /sys/module/paramsModule/parameters/secretValue
charparameter modifyValues secretValue studentId
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo vim /sys/module/paramsModule/parameters/secretValue
[Dec14 09:16] [paramsModule - initialize] =====
[+0.00000] [paramsModule - initialize] Hello!
[+0.00001] [paramsModule - initialize] Student Id = [-1]
[+0.00001] [paramsModule - initialize] String inside module = [This is a dummy message!]
[+0.00001] [paramsModule - initialize] Secret value = [-2]
[Dec14 09:17] [paramsModule - clean_exit] =====
[+0.00000] [paramsModule - clean_exit] Goodbye!
[+0.00001] [paramsModule - clean_exit] Student Id = [-1]
[+0.00001] [paramsModule - clean_exit] String inside module = [This is a dummy message!]
[+0.00001] [paramsModule - clean_exit] Secret value = [-2]
[Dec14 09:18] [paramsModule - initialize] =====
[+0.00000] [paramsModule - initialize] Hello!
[+0.00001] [paramsModule - initialize] Student Id = [616087]
[+0.00001] [paramsModule - initialize] String inside module = [Hello world! Project 02 - Example 02]
[+0.00001] [paramsModule - initialize] Secret value = [8888]
root@user0616087:~/Desktop/Test/Modules/paramsModule
777
```

The terminal window has a dark theme with light-colored text. The status bar at the bottom shows "Type here to search", "1,1 All", and the date/time "12/15/2020 1:24 AM".

#13

In step 11, that the secretValue file has been modified, when we unload the module in Terminal 1, we will see the new value in Terminal 2

The screenshot shows a terminal window titled "Ubuntu/64Target - VMware Workstation 16 Player (Non-commercial use only)". The terminal output is as follows:

```
root@user0616087:~/Desktop/Test/Modules/paramsModule
lsmod | grep paramsModule
lsmod: ERROR! could not insert module paramsModule.ko: File exists
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule.ko modifyValues=1
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ ls /sys/module/paramsModule/parameters/
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ ls /sys/module/paramsModule/parameters/secretValue
charparameter modifyValues secretValue studentId
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule
user0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule
[Dec14 09:17] [paramsModule - clean_exit] =====
[+0.00000] [paramsModule - clean_exit] Goodbye!
[+0.00001] [paramsModule - clean_exit] Student Id = [-1]
[+0.00001] [paramsModule - clean_exit] String inside module = [This is a dummy message!]
[+0.00001] [paramsModule - clean_exit] Secret value = [-2]
[Dec14 09:18] [paramsModule - initialize] =====
[+0.00000] [paramsModule - initialize] Hello!
[+0.00001] [paramsModule - initialize] Student Id = [616087]
[+0.00001] [paramsModule - initialize] String inside module = [Hello world! Project 02 - Example 02]
[+0.00001] [paramsModule - initialize] Secret value = [8888]
[Dec14 09:19] [paramsModule - clean_exit] =====
[+0.00000] [paramsModule - clean_exit] Goodbye!
[+0.00001] [paramsModule - clean_exit] Student Id = [616087]
[+0.00001] [paramsModule - clean_exit] String inside module = [Hello world! Project 02 - Example 02]
[+0.00001] [paramsModule - clean_exit] Secret value = [7777]
root@user0616087:~/Desktop/Test/Modules/paramsModule
777
```

The terminal window has a dark theme with light-colored text. The status bar at the bottom shows "Type here to search", "1,1 All", and the date/time "12/15/2020 1:28 AM".

[Screenshot #14: Create a screenshot explaining step 12.]

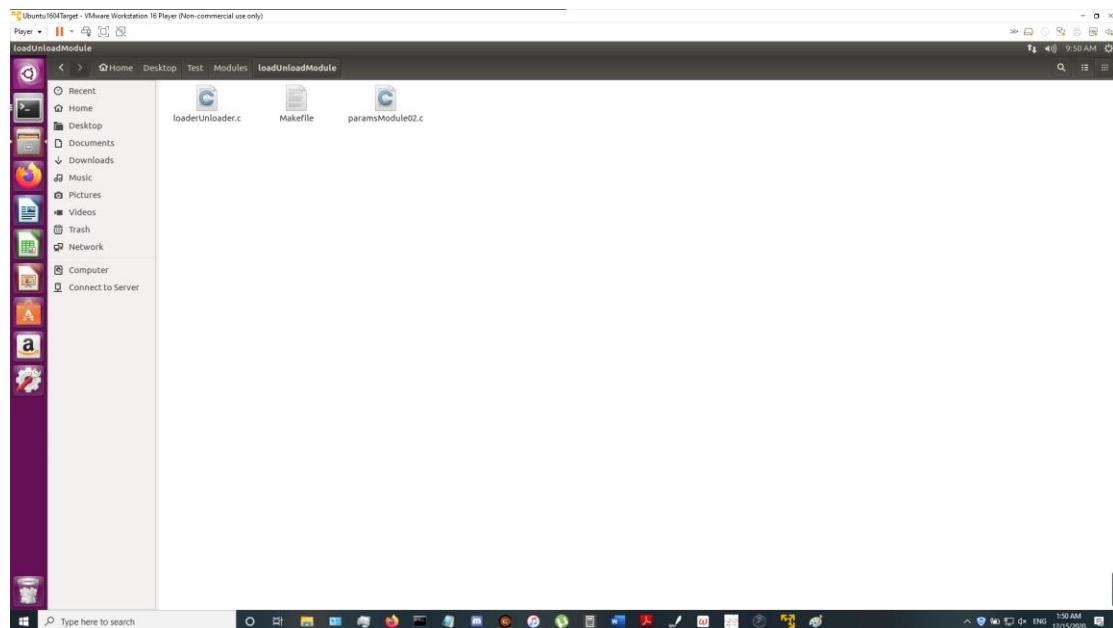
In step 12, the reason **paramsModule: unknown parameter ‘dummyStudentId’ ignored** appears is because there is no parameter called dummyStudentId, so the kernel will ignore it and print the original values.

```
Ubuntu/16.04Target - VMware Workstation 16 Player (Non-commercial use only)
Player
Ubuntu Desktop
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule
vermagic: 4.19.148 SMP mod_unload
parm:    studentId= [0] for Student Id. (Leading zeros are omitted) (int)
parm:    secretValue= [0] for secret value. (long)
parm:    charparameter=states - Hello world (charp)
parm:    modifyValues=indicates if we must modify the original values or not. (int)
parm:    dummystudentId= [0] for dummy student id. (long)
insmod: ERROR: could not insert module paramsModule.ko: File exists
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule.ko modifyValues=1
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko modifyValues=1
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule.ko
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko studentId=6160887 secretValue=8888
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule$ ./sys/module/paramsModule/parameters/
charparameter: modifyValues secretValue studentId
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule$ sudo rmmod paramsModule.ko
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule$ sudo insmod paramsModule.ko dummystudentId=9999
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule$ 
user@test0616087:~/ubuntu$ ./Desktop/Test/Modules/paramsModule
[Dec14 09:28] [paramsModule - initialize] =====
+0.000001 | [paramsModule - initialize] Student Id = [6160887]
+0.000001 | [paramsModule - initialize] String Inside module = [Hello world! Project 02 - Example 02]
+0.000001 | [paramsModule - initialize] Secret value = [8888]
[Dec14 09:28] [paramsModule - clean_exit] =====
+0.000001 | [paramsModule - clean_exit] dummystudentId= [9999]
+0.000001 | [paramsModule - clean_exit] Student Id = [6160887]
+0.000001 | [paramsModule - clean_exit] String Inside module = [Hello world! Project 02 - Example 02]
+0.000001 | [paramsModule - clean_exit] Secret value = [7777]
[Dec14 09:29] [paramsModule] unknown parameter 'dummystudentId' ignored
[Dec14 09:56] [paramsModule - initialize] =====
+0.000001 | [paramsModule - initialize] dummystudentId= [9999]
+0.000001 | [paramsModule - initialize] Student Id = [6160887]
+0.000001 | [paramsModule - initialize] String Inside module = [Hello world! Project 02 - Example 02]
+0.000001 | [paramsModule - initialize] Secret value = [987654321]
```

[Screenshot #15-16: Create two screenshots showing the files above in their folder and EXPLAIN the loaderUnloader.c file]

#15

Files above in their folder



#16

The loaderUnloader.c is like putting parameters **paramsNew** into init_module, moduleName into fd, use init_module to deliver paramsNew into module and mount the module. After that we close(fd) to unmounts the module.

The image shows two terminal windows side-by-side, both titled '(loaderUnloader.c /Desktop/Test/Modules/loadUnloadModule) - edit'. The left window displays the initial code for a dynamic loader and unloader. The right window shows the completed code with additional comments and logic to handle module parameters and ensure proper cleanup.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <unistd.h>
#include <dlib.h>

#define int_module(module_image, len, param_values) syscall(__NR_init_module, module_image, len, param_values)
#define delete_module(fd, name, flags) syscall(__NR_fini_module, fd, param_values, flags)

// Change your data accordingly
// StudentID: 61087

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=61087"; // Use your StudentID without leading 0

    int fd, use_size;
    size_t image_size;
    struct stat st;
    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 (int_module(image, image_size, paramsNew) != 0) {
        perror("int module");
        return EXIT_FAILURE;
    }
    printf("Module is mounted!\n");
    //Section - Module loading - END
    // At this point the module is mounted.

    //Section - Module unloading - BEGIN
    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;
}
```

```
//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 (int_module(image, image_size, paramsNew) != 0) {
    perror("int 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 extension>/parameters
// Note: DO NOT CHANGE THE VARIABLES WITHOUT FOLLOWING THE CORRECT DATATYPE YOUR MODULE WILL GET LOCKED AND YOU MUST FIX THE VARIABLES
// AND RESTART YOUR MACHINE.

printf("[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 #17-18-19-20: Create four screenshots explaining steps 4,5,6,7,8.]

#17 compile loaderUnloader.c with \$ **gcc -o loaderUnloader** loaderUnloader.c.

And then make the module.

#18 \$ dmesg -wH in Terminal 2

```
Ubuntu:000@Ubuntu: ~Desktop/Test/Modules/loadUnloadModule
CC      /home/ubuntu/test8616087/Desktop/test/Modules/loadUnloadModule/paramsModule02.mod.o
LD      /home/ubuntu/test8616087/Desktop/test/Modules/loadUnloadModule/paramsModule02.ko
make[1]: Leaving directory '/usr/src/linux-4.19.148'
user@test8616087:~/Desktop/Test/Modules/loadUnloadModule$ sudo ./loaderUnloader
[sudo] password for user@test8616087:

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=610087]...
Module is mounted!

[Press ENTER to continue]

Unmounting module...
Module is unmounted!
Cleaning...
user@test8616087:~/Desktop/Test/Modules/loadUnloadModule$ sudo ./loaderUnloader

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=610087]...
Module is mounted!

[Press ENTER to continue]

A Unmounting module...
Module is unmounted!
Cleaning...
Done!
user@test8616087:~/Desktop/Test/Modules/loadUnloadModule$ gcc -o loaderUnloader loaderUnloader.c

user@test8616087:~/Desktop/Test/Modules/loadUnloadModule$ make
No command 'ale' found, did you mean:
  Command 'ale' from package 'ale' (universe)
  Command 'alexa' from package 'alexa' (multiverse)
  Command 'name' from package 'make' (main)
  Command 'name' from package 'make-guile' (universe)
nothing else available
user@test8616087:~/Desktop/Test/Modules/loadUnloadModule$ gcc -o loaderUnloader loaderUnloader.c

user@test8616087:~/Desktop/Test/Modules/loadUnloadModule$ make
make[1]: Entering directory '/usr/src/linux-4.19.148'
  Building file: CMakeFiles/loaderUnloader.dir/loaderUnloader.cpp.o
  Making dependency tree
  Generating dot files
  Stage 2.
  MODPOST 1 modules.
make[1]: Leaving directory '/usr/src/linux-4.19.148'
user@test8616087:~/Desktop/Test/Modules/loadUnloadModule$ sudo ./loaderUnloader
[sudo] password for user@test8616087:

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=610087]...
Module is mounted!

[Press ENTER to continue]

user@test8616087:~/Desktop/Test/Modules/loadUnloadModule$ ./paramsModule02
[+0.000000] [paramsModule - clean_exit] String inside module = [Hello world! Project 02 - Example 02]
[+0.000000] [paramsModule - clean_exit] Secret value = [7777]
[Dec14 09:29] paramsModule: unknown parameter 'dummyStudentId' ignored
[+0.000050] [paramsModule - initialize] *****
[+0.000000] [paramsModule - initialize] Hello!
[+0.000000] [paramsModule - initialize] Student Id = [610087]
[+0.000000] [paramsModule - initialize] String inside module = [Hello world! Project 02 - Example 02]
[+0.000000] [paramsModule - initialize] Secret value = [987654321]
[Dec14 09:30] [paramsModule - clean_exit] *****
[+0.000000] [paramsModule - clean_exit] Goodbye!
[+0.000000] [paramsModule - clean_exit] Student Id = [610087]
[+0.000000] [paramsModule - clean_exit] String inside module = [Hello world! Project 02 - Example 02]
[+0.000000] [paramsModule - clean_exit] Secret value = [987654321]
[Dec14 09:52] [paramsModule02 - initialize] *****
[+0.000000] [paramsModule02 - initialize] Hello!
[+0.000000] [paramsModule02 - initialize] Student Id = [610087]
[+0.000000] [paramsModule02 - initialize] String inside module = [Hello world! Project 02 - Example 02]
[+0.000000] [paramsModule02 - initialize] Secret value = [987654321]
[Dec14 09:53] [paramsModule02 - clean_exit] *****
[+0.000000] [paramsModule02 - clean_exit] Goodbye!
[+0.000000] [paramsModule02 - clean_exit] Student Id = [610087]
[+0.000000] [paramsModule02 - clean_exit] String inside module = [Hello world! Project 02 - Example 02]
[+0.000000] [paramsModule02 - clean_exit] Secret value = [987654321]
[Dec14 09:56] [paramsModule02 - initialize] *****
[+0.000000] [paramsModule02 - initialize] Hello!
[+0.000000] [paramsModule02 - initialize] Student Id = [610087]
[+0.000000] [paramsModule02 - initialize] String inside module = [Hello world! Project 02 - Example 02]
[+0.000000] [paramsModule02 - initialize] Secret value = [987654321]
[Dec14 09:57] [paramsModule02 - clean_exit] *****
[+0.000000] [paramsModule02 - clean_exit] Goodbye!
[+0.000000] [paramsModule02 - clean_exit] Student Id = [610087]
[+0.000000] [paramsModule02 - clean_exit] String inside module = [Hello world! Project 02 - Example 02]
[+0.000000] [paramsModule02 - clean_exit] Secret value = [987654321]
[Dec14 10:01] [paramsModule02 - initialize] *****
[+0.000000] [paramsModule02 - initialize] Hello!
[+0.000000] [paramsModule02 - initialize] Student Id = [610087]
[+0.000000] [paramsModule02 - initialize] String inside module = [Hello world! Project 02 - Example 02]
[+0.000000] [paramsModule02 - initialize] Secret value = [987654321]
```

#19 \$ sudo ./loaderUnloader. We can see the parameters shows on the screen. At this point the module has been loaded with studentId=9999. Because the module is loaded, we can read the variables inside it in Terminal 3

```

Ubuntu:004Target - VMware Workstation 16 Player (Non-commercial use only)
Player | || - & □ □
Terminal
user@test0616087@ubuntu:~/Desktop/Test/Modules/loadUnloadModule
>Loading module [paramsModule02] with parameters [studentId=616087]...
Module is mounted!
<Press ENTER to continue>
Unmounting module...
Module is unmounted!
Cleaning...
Done!
user@test0616087@ubuntu:~/Desktop/Test/Modules/loadUnloadModule$ gcc -o loaderUnloader loaderUnloader.c
user@test0616087@ubuntu:~/Desktop/Test/Modules/loadUnloadModule$ ./loaderUnloader loaderUnloader.c
[Dec14 09:53]
[+0.000001] [paramsModule02 - initialize] Student Id = [616087]
[+0.000001] [paramsModule02 - initialize] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - initialize] Secret value = [987654321]
[Dec14 09:56]
[+0.000001] [paramsModule02 - clean_exit] =====
[+0.000001] [paramsModule02 - clean_exit] Goodbye!
[+0.000001] [paramsModule02 - clean_exit] Student Id = [616087]
[+0.000001] [paramsModule02 - clean_exit] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] Secret value = [987654321]
[Dec14 10:03]
[+0.000001] [paramsModule02 - initialize] =====
[+0.000001] [paramsModule02 - initialize] Hello!
[+0.000001] [paramsModule02 - initialize] Student Id = [616087]
[+0.000001] [paramsModule02 - initialize] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] =====
[+0.000001] [paramsModule02 - clean_exit] Goodbye!
[+0.000001] [paramsModule02 - clean_exit] Student Id = [616087]
[+0.000001] [paramsModule02 - clean_exit] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] Secret value = [987654321]
[Dec14 10:03]
[+0.000001] [paramsModule02 - initialize] =====
[+0.000001] [paramsModule02 - initialize] Hello!
[+0.000001] [paramsModule02 - initialize] Student Id = [616087]
[+0.000001] [paramsModule02 - initialize] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] =====
[+0.000001] [paramsModule02 - clean_exit] Goodbye!
[+0.000001] [paramsModule02 - clean_exit] Student Id = [616087]
[+0.000001] [paramsModule02 - clean_exit] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] Secret value = [987654321]

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=616087]...
Module is mounted!
<Press ENTER to continue>

user@test0616087@ubuntu:~/Desktop/Test/Modules/loadUnloadModule$ ls /sys/module/paramsModule02/parameters/
charparameter modifyValues secretValue studentId
user@test0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$ lsmod | grep paramsModule02
paramsModule02 16384 0
user@test0616087@ubuntu:~/Desktop/Test/Modules/paramsModule$
```

#20 In Terminal 1 press ENTER. This will continue the execution of ./loaderUnloader, and it will proceed to unload the module. So In Terminal 2 we will see that the module was unloaded. And in Terminal 3 you will see that you cannot access the module's folder in /sys/ anymore and the module has been removed from the modules list.

```

Ubuntu:004Target - VMware Workstation 16 Player (Non-commercial use only)
Player | || - & □ □
Terminal
user@test0616087@ubuntu:~/Desktop/Test/Modules/loadUnloadModule
>Unmounting module...
Module is unmounted!
Cleaning...
user@test0616087@ubuntu:~/Desktop/Test/Modules/loadUnloadModule$ ./loaderUnloader loaderUnloader.c
[Dec14 09:56]
[+0.000001] [paramsModule02 - clean_exit] Student Id = [616087]
[+0.000001] [paramsModule02 - clean_exit] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] Secret value = [987654321]
[Dec14 09:57]
[+0.000001] [paramsModule02 - initialize] =====
[+0.000001] [paramsModule02 - initialize] Hello!
[+0.000001] [paramsModule02 - initialize] Student Id = [616087]
[+0.000001] [paramsModule02 - initialize] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] =====
[+0.000001] [paramsModule02 - clean_exit] Goodbye!
[+0.000001] [paramsModule02 - clean_exit] Student Id = [616087]
[+0.000001] [paramsModule02 - clean_exit] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] Secret value = [987654321]
[Dec14 10:03]
[+0.000001] [paramsModule02 - initialize] =====
[+0.000001] [paramsModule02 - initialize] Hello!
[+0.000001] [paramsModule02 - initialize] Student Id = [616087]
[+0.000001] [paramsModule02 - initialize] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] =====
[+0.000001] [paramsModule02 - clean_exit] Goodbye!
[+0.000001] [paramsModule02 - clean_exit] Student Id = [616087]
[+0.000001] [paramsModule02 - clean_exit] String Inside module = [Hello world! Project 02 - Example
03]
[+0.000001] [paramsModule02 - clean_exit] Secret value = [987654321]

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=616087]...
Module is mounted!
<Press ENTER to continue>

Unmounting module...
Module is unmounted!
Cleaning...
Done!
user@test0616087@ubuntu:~/Desktop/Test/Modules/loadUnloadModule$
```

[Screenshot #21-22-23-24: Create four screenshots explaining how you solve this problem.]

In screenshot#21-22 is my calculator.c. I add a variable in every function call, and use sprintf to put string in paramsNew, then put the parameters into module by using init_module. When one input “operator” “input1” “input2”, it will mount and deliver to calculatorModule.c and put into /sys/module/calculatorModule/parameters. The module would calculate and return results into returnParam, then calculator.c would read the files and print value inside resultParam. After that the module will dynamically unmount.

The image consists of two side-by-side screenshots of a Linux desktop environment. Both screenshots show a window titled 'calculator.c' from the file path '/Desktop/Test/Modules/calculatorModule'. The top screenshot shows the beginning of the code, including the main() function and several arithmetic operators (addition, subtraction, multiplication). The bottom screenshot shows the continuation of the code, specifically the implementation of the 'long addition' function, which involves opening a module file, writing parameters to it, and reading the result back. The desktop background is visible at the bottom, showing icons for various applications like a browser, file manager, and system tools.

```
calculator.c
#define finit_module(fd, param_values, flags) syscall__NR_finit_module(fd, param_values, flags)
#define delete_module(name, flags) syscall__NR_delete_module(name, flags)

long addition (int input1, int input2, char* sum);
long subtraction (int input1, int input2, char* sub);
long multiplication (int input1, int input2, char* mul);

// StudentID - 616087

int main()
{
    char operator;
    char *operation;

    int input1=0;
    int input2=0;
    long result = 0;

    while(1)
    {
        input1=0;
        input2=0;
        result = 0;

        printf("*****\n");
        printf("Enter operation [sum - sub - mul - exit]: ");
        scanf("%s",operation);

        if(strcmp(operation,"exit")==0){
            break;
        }

        printf("Enter two operands (space separated): ");
        scanf("%d %d", &input1, &input2);

        if(strcmp(operation,"sum")==0){
            result = addition(input1, input2, operation);
            printf("Operation: [%s] - Operands [%d %d] - Result:[%ld]\n",operation, input1, input2, result);
        }else if(strcmp(operation,"sub")==0){
            result = subtraction(input1, input2, operation);
            printf("Operation: [%s] - Operands [%d %d] - Result:[%ld]\n",operation, input1, input2, result);
        }else if(strcmp(operation,"mul")==0){
            result = multiplication(input1, input2, operation);
            printf("Operation: [%s] - Operands [%d %d] - Result:[%ld]\n",operation, input1, input2, result);
        }else{
            printf("Invalid option\n");
        }
    }
    return 0;
}

long addition (int input1, int input2, char* sum)
{
    long result = 0;

    const char *moduleName = "calculatorModule.ko";
    const char *moduleNameNoExtension = "calculatorModule";

    int fd = use_fcntl();
    size_t image_size;
    struct stat st;
    void *image;

    char paramsNew[500];
    sprintf(paramsNew, "operationParam=%s firstParam=%d secondParam=%d", sum, input1, input2 );

    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, paramsNew) != 0) {
        perror("init module failed");
        return EXIT_FAILURE;
    }

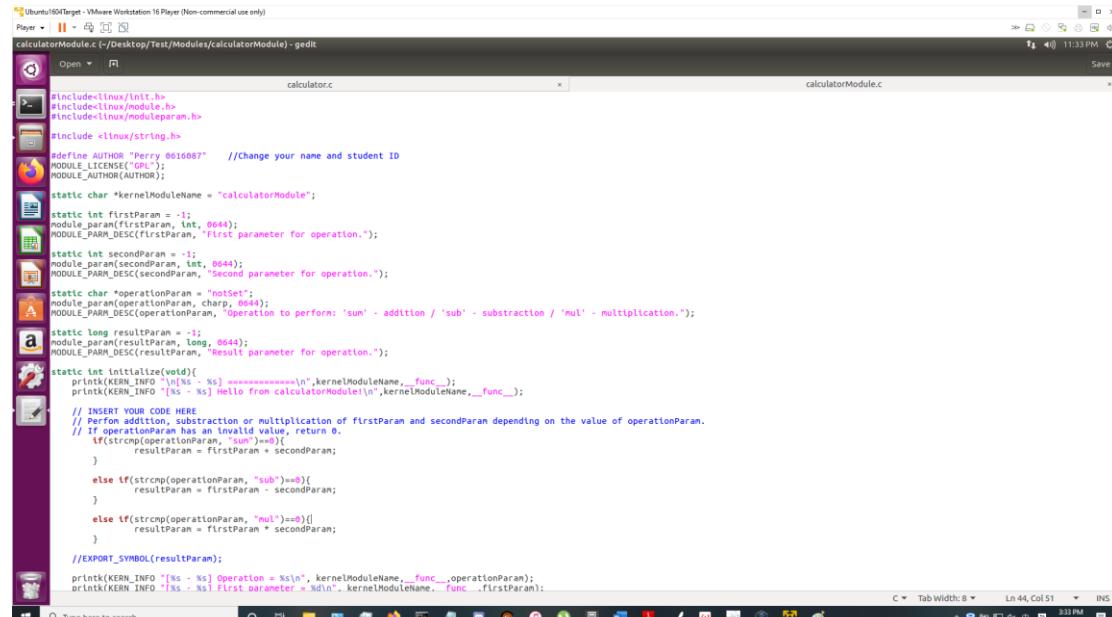
    FILE *fptr;
    fptr = fopen("/sys/module/calculatorModule/parameters/resultParam", "r");
    if (!fptr) {
        printf("open failed...\n");
        exit(1);
    }
    fscanf(fptr, "%ld", &result);
    fclose(fptr);

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

    close(fd);
    free(image);
    return result;
}

long subtraction (int input1, int input2, char* sub)
```

#23 This is my calculatorModule.c. I use strcmp to identify the operation and calculate the result. Because the values (firstParam, secondParam, operationParam) has been modified by init_module, the result=firstParam(operation)secondParam, and finish the calculation.



```

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

#include <linux/string.h>

#define AUTHOR "Perry 0616087" //Change your name and student ID
MODULE_LICENSE("GPL");
MODULE_AUTHOR(AUTHOR);

static char *kernelModuleName = "calculatorModule";

static int firstParam = -1;
module_param(firstParam,int,_0644);
MODULE_PARM_DESC(firstParam, "First parameter for operation.");

static int secondParam = -1;
module_param(secondParam,int,_0644);
MODULE_PARM_DESC(secondParam, "Second parameter for operation.");

static char *operationParam = "notSet";
module_param(operationParam,charp,_0644);
MODULE_PARM_DESC(operationParam, "Operation to perform: 'sum' - addition / 'sub' - subtraction / 'mul' - multiplication.");

static long resultParam = -1;
module_param(resultParam,long,_0644);
MODULE_PARM_DESC(resultParam, "Result parameter for operation.");

static int initialize(void){
    printk(KERN_INFO "[ks - ks] =====\n", kernelModuleName, __func__);
    printk(KERN_INFO "[ks - ks] Hello from calculatorModule\n", kernelModuleName, __func__);

    // INSERT YOUR CODE HERE
    // If operationParam is not set, subtraction or multiplication of firstParam and secondParam depending on the value of operationParam.
    // If operationParam has an invalid value, return 0.
    if(strcmp(operationParam, "sum") == 0){
        resultParam = firstParam + secondParam;
    }
    else if(strcmp(operationParam, "sub") == 0){
        resultParam = firstParam - secondParam;
    }
    else if(strcmp(operationParam, "mul") == 0){
        resultParam = firstParam * secondParam;
    }

    //EXPORT_SYMBOL(resultParam);
    printk(KERN_INFO "[ks - ks] Operation = %s\n", kernelModuleName, __func__, operationParam);
    printk(KERN_INFO "[ks - ks] First parameter = %d\n", kernelModuleName, __func__, firstParam);
}

//INSERT YOUR CODE HERE
//If operationParam has an invalid value, return 0.
if(strcmp(operationParam, "sum") == 0){
    resultParam = firstParam + secondParam;
}
else if(strcmp(operationParam, "sub") == 0){
    resultParam = firstParam - secondParam;
}
else if(strcmp(operationParam, "mul") == 0){
    resultParam = firstParam * secondParam;
}

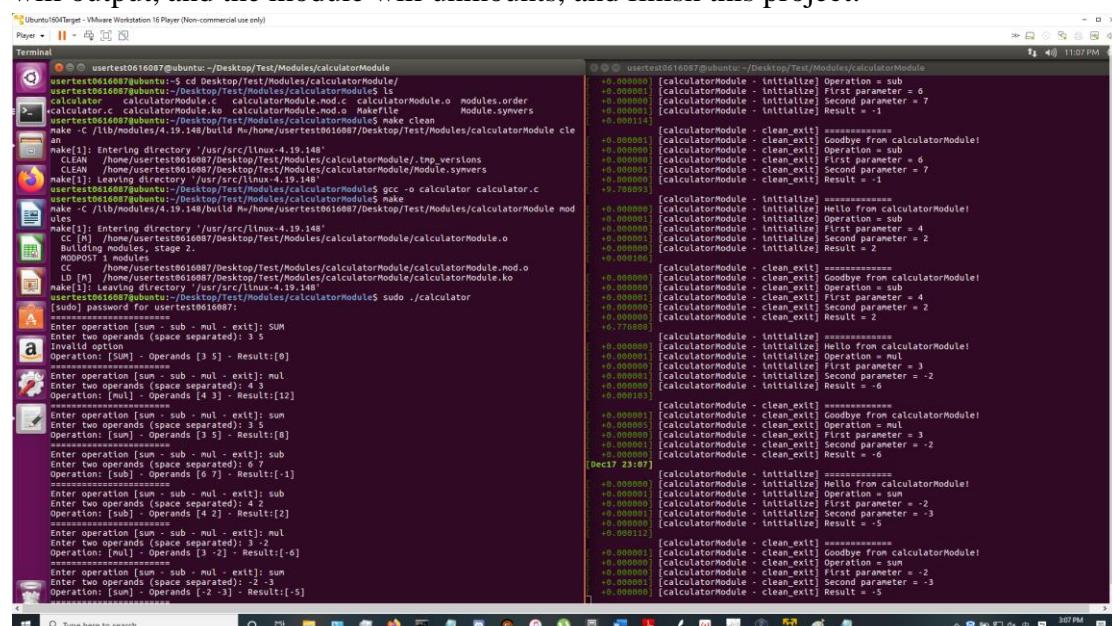
//EXPORT_SYMBOL(resultParam);

printk(KERN_INFO "[ks - ks] Operation = %s\n", kernelModuleName, __func__, operationParam);
printk(KERN_INFO "[ks - ks] First parameter = %d\n", kernelModuleName, __func__, firstParam);

```

#24

This screenshot is my demo result. As you can see every time I enter operation and inputs, the module will mount automatically, and after the calculate is over, the result will output, and the module will unmounts, and finish this project.



```

user@0616087:~$ cd Desktop/Test/Modules/calculatorModule
user@0616087:~/Desktop/Test/Modules/calculatorModule$ make clean
user@0616087:~/Desktop/Test/Modules/calculatorModule$ make
make < /lib/modules/4.19.148/build M=/home/user@0616087/Desktop/Test/Modules/calculatorModule cle
an
make[1]: Entering directory '/usr/src/linux-4.19.148'
  CC [M] /home/user@0616087/Desktop/Test/Modules/calculatorModule/tmp_versions
  LD [M] /home/user@0616087/Desktop/Test/Modules/calculatorModule/module.symvers
make[1]: Leaving directory '/usr/src/linux-4.19.148'
user@0616087:~/Desktop/Test/Modules/calculatorModule$ pcc -o calculator calculator.c
user@0616087:~/Desktop/Test/Modules/calculatorModule$ make
make -C /lib/modules/4.19.148/build M=/home/user@0616087/Desktop/Test/Modules/calculatorModule mod
ules
make[1]: Entering directory '/usr/src/linux-4.19.148'
  CC [M] /home/user@0616087/Desktop/Test/Modules/calculatorModule/module.o
Building modules, stage 2.
MODPOST 1 modules
/home/user@0616087/Desktop/Test/Modules/calculatorModule/calculatorModule.mod.o
LD [M] /home/user@0616087/Desktop/Test/Modules/calculatorModule/calculatorModule.ko
make: Leaving directory '/usr/src/linux-4.19.148'
user@0616087:~/Desktop/Test/Modules/calculatorModule$ sudo ./calculator
[sudo] password for user@0616087:
A
Enter operation [sum - sub - mul - ext]: sum
Enter two operands (space separated): 3 5
Invalid option
Operation: [sum] - Operands [3 5] - Result:[0]
=====
Enter operation [sum - sub - mul - ext]: sub
Enter two operands (space separated): 3 5
Operation: [sub] - Operands [3 5] - Result:[-2]
=====
Enter operation [sum - sub - mul - ext]: mul
Enter two operands (space separated): 3 5
Operation: [mul] - Operands [3 5] - Result:[15]
=====
Enter operation [sum - sub - mul - ext]: sum
Enter two operands (space separated): 3 5
Operation: [sum] - Operands [3 5] - Result:[8]
=====
Enter operation [sum - sub - mul - ext]: sub
Enter two operands (space separated): 4 2
Operation: [sub] - Operands [4 2] - Result:[2]
=====
Enter operation [sum - sub - mul - ext]: mul
Enter two operands (space separated): 3 2
Operation: [mul] - Operands [3 2] - Result:[6]
=====
Enter operation [sum - sub - mul - ext]: sum
Enter two operands (space separated): 3 2
Operation: [sum] - Operands [3 2] - Result:[-6]
=====
Enter operation [sum - sub - mul - ext]: sub
Enter two operands (space separated): 3 2
Operation: [sub] - Operands [3 2] - Result:[-1]
=====
Enter operation [sum - sub - mul - ext]: mul
Enter two operands (space separated): 3 2
Operation: [mul] - Operands [3 2] - Result:[-6]
=====
Enter operation [sum - sub - mul - ext]: sum
Enter two operands (space separated): 3 2
Operation: [sum] - Operands [3 2] - Result:[-3]
=====
Enter operation [sum - sub - mul - ext]: sub
Enter two operands (space separated): 3 2
Operation: [sub] - Operands [3 2] - Result:[-5]
=====

user@0616087:~/Desktop/Test/Modules/calculatorModule$ [calculatorModule - initialize] Operation = sub
[calculatorModule - initialize] First parameter = 6
[calculatorModule - initialize] Second parameter = 7
[calculatorModule - initialize] Result = -1
[calculatorModule - clean_exit] =====
[calculatorModule - clean_exit] Goodbye from calculatorModule!
[calculatorModule - clean_exit] Operation = sub
[calculatorModule - clean_exit] First parameter = 6
[calculatorModule - clean_exit] Second parameter = 7
[calculatorModule - clean_exit] Result = -1
[calculatorModule - initialize] =====
[calculatorModule - initialize] Hello from calculatorModule!
[calculatorModule - initialize] Operation = sub
[calculatorModule - initialize] First parameter = 4
[calculatorModule - initialize] Second parameter = 2
[calculatorModule - initialize] Result = 2
[calculatorModule - clean_exit] =====
[calculatorModule - clean_exit] Goodbye from calculatorModule!
[calculatorModule - clean_exit] Operation = sub
[calculatorModule - clean_exit] First parameter = 4
[calculatorModule - clean_exit] Second parameter = 2
[calculatorModule - clean_exit] Result = 2
[calculatorModule - initialize] =====
[calculatorModule - initialize] Hello from calculatorModule!
[calculatorModule - initialize] Operation = mul
[calculatorModule - initialize] First parameter = 3
[calculatorModule - initialize] Second parameter = -2
[calculatorModule - initialize] Result = -6
[calculatorModule - clean_exit] =====
[calculatorModule - clean_exit] Goodbye from calculatorModule!
[calculatorModule - clean_exit] Operation = sum
[calculatorModule - initialize] First parameter = -2
[calculatorModule - initialize] Second parameter = -3
[calculatorModule - initialize] Result = -5
[calculatorModule - clean_exit] =====
[calculatorModule - clean_exit] Goodbye from calculatorModule!
[calculatorModule - clean_exit] Operation = sum
[calculatorModule - clean_exit] First parameter = -2
[calculatorModule - clean_exit] Second parameter = -3
[calculatorModule - clean_exit] Result = -5

```

Part2-Q&A

- 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:

- Static kernel modules are those which are compiled as part of the base kernel and it is available at any time.
- Dynamic kernel modules are compiled as modules separately and loaded based on user demand.
- Dynamic kernel module is also called as Loadable Kernel Module(LKM)
- System call is a function in the kernel, and is visible for users. Also a user need to execute a syscall to get services from kernel. While dynamic kernel module is an object file that can be insert into kernel. It's used for expanding the kerne's functionality, also it don't need to re-compile the kernel to add new dynamic kernel module.

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

Ans:

Because it's not possible to hot-patch a new syscall into a running kernel. The only way of properly doing so is by getting the kernel source, modifying it, configuring it, and then re-compiling it as a brand new kernel.

However, kernel modules cannot add new system calls, so build and load kernel modules don't need to recompiling the kernel.

- 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 module into kernel. **\$sudo insmod dummyModule.ko**

rmmod:

Remove loaded module from kernel. **\$sudo rmmod dummyModule.ko**

modinfo:

Check base information of module. **\$sudo 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 is used to mark a function to be used as the entry-point of a Linux device-driver. It is called during do_initcalls() (for a builtin driver) or at module insertion time (for a *.ko module)

b. module_exit will wrap the driver clean-up code with cleanup_module when used with rmmod when the driver is a module. If the driver is statically compiled into the kernel, module_exit has no effect.

c. MODULE_LICENSE() is a macro which allows loadable kernel modules to declare their license to the world. Its purpose is to let the kernel developers know when a non-free module has been inserted into a given kernel.

d. The module_param() macro takes 3 arguments: the name of the variable, its type and permissions for the corresponding file in sysfs, and bring them to module.

e. MODULE_PARM_DESC() is used to document arguments that the module can take. It takes two parameters: a variable name and a free form string describing that 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. The 'cat' [short for "concatenate"] command is one of the most frequently used commands in Linux and other operating systems. The cat command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.

b. The 'ls' means list all contents under the path. The option '-l' tells the command to use a long list format.

c. 'dmesg' (diagnostic message) is a command that prints the message buffer of the kernel. The output includes messages produced by the device drivers. '-wH' enables user-friendly features like colors or relative time.

d. The 'lsmod' command is used to display the status information of the modules that have been loaded into the kernel. After executing the lsmod command, all modules loaded into the system will be listed.

e. Since the output result is very long, we can use lsmod | grep to display a specific module string

- 6. There is a 0644 in the line module_param(studentId, int, 0644); inside paramsModule.c (Section 1.3). What does 0644 mean?**

Ans:

0644 means which users have permission could modify the parameter.

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

Ans:

```
if (init_module(image, image_size, buffer) != 0) {  
    perror("init_module");  
    return EXIT_FAILURE;  
}
```

If init_module() !=0, it will print descriptive error messages stderr, and return the parameter value of EXIT_FAILURE, which is 1.

- 8. In Section 1.3 – step 6, modinfo shows the information of some variables inside the module but two of them are not displayed. Why is it?**

Ans:

the reason, modinfo shows the information of some variables inside the module but two of them are not displayed, is because there is no parameter called dummyStudentId, so the kernel will ignore it and print the original values.

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

Ans:

For storing those parameters delivered by init_module and could be used by mounted module.

- 10. In Section 1.3 (paramsModule.c), the variable charparameter is of type charp. What is charp?**

Ans:

If the module parameter is a string, the ‘charp’ type is usually used to define the module parameter. The core copies the string provided by the user to the memory, and the corresponding variable points to this string.