Operating Systems Project Report

Project Number (01 / 02 / 03):	03
Name:	陳宥安 CHEN, Yu-An
Student ID:	109550073
YouTube link (Format youtube.com/watch?v=[key]):	https://youtu.be/8NI_6ig-LvOU
Date (YYYY-MM-DD):	2021/12/27
Names of the files uploaded to E3:	calculator.c calculatorModule.c OS_Project03_109550073.pdf
Physical Machine Total RAM (Example: 8.0 GB):	16.0 GB
Physical Machine CPU (Example: Intel i7-2600K):	Intel(R) Core(TM) i5-8250U CPU @1.60GHz

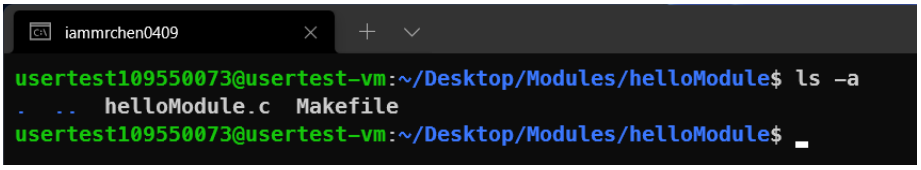
Checklist	
Yes/No	**Item**
YES	**The report name follows the format "OS_ProjectXX_StudentID.pdf".**
YES	**The report was uploaded to E3 before the deadline.**
YES	**The YouTube video is public, and anyone with the link can watch it.**
YES	**The audio of the video has a good volume.**
YES	**The pictures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

1. Screenshots

Section 1.1: Dynamically-Loadable Kernel Modules – Example 1: Hello world

SCREENSHOT #1 helloModule

The screenshots show the folder helloModule and its contents, helloModule.c and Makefile.



```
  GNU nano 4.8                                          helloModule.c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#define DRIVER_AUTHOR "CHEN, Yu-An - 109550073" // Replace with your name and student ID
#define DRIVER_DESC "A sample driver - OS Project 03"
static int studentId = 109550073;
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);
MODULE_LICENSE("GPL");
MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESC);
```

```
  GNU nano 4.8                                          Makefile
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 module

The screenshot shows the result of make process

```
iammrchen0409        iammrchen0409@Ubuntu
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ make clean
make -C /lib/modules/5.13.19/build M=/home/usertest109550073/Desktop/Modules/helloModule clean
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CLEAN   /home/usertest109550073/Desktop/Modules/helloModule/Module.symvers
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ make
make -C /lib/modules/5.13.19/build M=/home/usertest109550073/Desktop/Modules/helloModule modules
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CC [M]  /home/usertest109550073/Desktop/Modules/helloModule/helloModule.o
  MODPOST /home/usertest109550073/Desktop/Modules/helloModule/Module.symvers
  CC [M]  /home/usertest109550073/Desktop/Modules/helloModule/helloModule.mod.o
  LD [M]  /home/usertest109550073/Desktop/Modules/helloModule/helloModule.ko
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$
```

SCREENSHOT #3 load module

The screenshot shows the result of load process on both terminal 1 and terminal 2

```
iammrchen0409        iammrchen0409@Ubuntu
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ make clean
make -C /lib/modules/5.13.19/build M=/home/usertest109550073/Desktop/Modules/helloModule clean
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CLEAN   /home/usertest109550073/Desktop/Modules/helloModule/Module.symvers
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ make
make -C /lib/modules/5.13.19/build M=/home/usertest109550073/Desktop/Modules/helloModule modules
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CC [M]  /home/usertest109550073/Desktop/Modules/helloModule/helloModule.o
  MODPOST /home/usertest109550073/Desktop/Modules/helloModule/Module.symvers
  CC [M]  /home/usertest109550073/Desktop/Modules/helloModule/helloModule.mod.o
  LD [M]  /home/usertest109550073/Desktop/Modules/helloModule/helloModule.ko
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo insmod helloModule.ko
[sudo] password for usertest109550073:
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$
```

```
iammrchen0409        iammrchen0409@Ubuntu
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo dmesg --clear
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ dmesg -wH
[十二14 16:10] helloModule: loading out-of-tree module taints kernel.
[  +0.000040] helloModule: module verification failed: signature and/or required key missing - tainting kernel
[  +0.000410] [109550073] : Function [initialize] - Hello from OS Project 03!
```

SCREENSHOT #4 loaded module

The screenshot shows the list of loaded modules.

```
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo insmod helloModule.ko
[sudo] password for usertest109550073:
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ lsmod
Module                  Size  Used by
helloModule            16384  0
isofs                  49152  1
rfcomm                 81920  4
bnep                   24576  2
intel_rapl_msr         20480  0
intel_rapl_common      24576
crct10dif_pclmul       16384
ghash_clmulni_intel    16384
aesni_intel           376832
crypto_simd            16384
snd_ens1371            32768
cryptd                 24576
vmw_balloon            24576
rapl                   20480
snd_ac97_codec        139264
gameport               20480
ac97_bus               16384
snd_pcm                114688
joydev                 28672
input_leds             16384
snd_seq_midi           20480
snd_seq_midi_event     16384
snd_rawmidi            36864
serio_raw              20480
snd_seq                73728
btusb                  61440
snd_seq_device         16384
snd_timer              40960
btrtl                  24576  1 btusb
btbcm                  16384  1 btusb
btintel                32768  1 btusb
bluetooth             543072  27 btrtl,btintel,btbcm,bnep,btusb,rfcomm
snd                    94208  11 snd_seq,snd_seq_device,snd_timer,snd_ac97_codec,snd_pcm,snd_rawmidi,snd_ens1371
ecdh_generic           16384  1 bluetooth
ecc                    36864  1 ecdh_generic
soundcore              16384  1 snd
vsock_loopback         16384  0
vmw_vsock_virtio_transport_common    36864  1 vsock_loopback
vmw_vsock_vmci_transport    28672  2
vsock                  45056  7 vmw_vsock_virtio_transport_common,vsock_loopback,vmw_vsock_vmci_transport
vmw_vmci               69632  2 vmw_balloon,vmw_vsock_vmci_transport
mac_hid                16384  0
sch_fq_codel           20480  2
vmwgfx                319488  5
ttm                    69632  1 vmwgfx
drm_kms_helper        253952  1 vmwgfx
cec                    53248  1 drm_kms_helper
rc_core                61440  1 cec
fb_sys_fops            16384  1 drm_kms_helper
syscopyarea            16384  1 drm_kms_helper
sysfillrect            16384  1 drm_kms_helper
sysimgblt              16384  1 drm_kms_helper
msr                    16384  0
parport_pc             45056  0
ppdev                  24576  0
lp                     20480
parport                65536
drm                   557056
ip_tables              32768
x_tables               49152
autofs4                45056
hid_generic            16384
crc32_pclmul           16384  0
psmouse               155648  0
usbhid                 57344  0
mptspi                 24576  2
hid                   135168  2 usbhid,hid_generic
ahci                   40960  1
libahci                36864  1 ahci
e1000                 143360  0
mptscsih               45056  1 mptspi
mptbase                98304  2 mptspi,mptscsih
i2c_piix4              28672  0
scsi_transport_spi     32768  1 mptspi
pata_acpi              16384  0
floppy                 81920  0
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ lsmod
```

```
mptscsih               45056  1 mptspi
mptbase                98304  2 mptspi,mptscsih
i2c_piix4              28672  0
scsi_transport_spi     32768  1 mptspi
pata_acpi              16384  0
floppy                 81920  0
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ lsmod | grep helloModule
helloModule            16384  0
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$
```

SCREENSHOT #5 unload module

The screenshot shows the process of unload module on both terminal 1 and terminal 2.

```
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo rmmod helloModule
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$
```

```
iammrchen0409    iammrchen0409@Ubuntu
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo dmesg --clear
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ dmesg -wH
[十二14 16:10] helloModule: loading out-of-tree module taints kernel.
[  +0.000040] helloModule: module verification failed: signature and/or required key missing - tainting kernel
[  +0.000410] [109550073] : Function [initialize] - Hello from OS Project 03!
[十二14 16:16] [109550073] : Function [clean_exit] - Unloading module. Goodbye from OS Project 03!
```

SCREENSHOT #6 search unloaded module

The screenshot shows the result of searching unloaded module.

```
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo rmmod helloModule
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ lsmod | grep helloModule
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$
```

<u>Section 1.2: Dynamically-Loadable Kernel Modules – Example 2: Sending parameters to a module and reading and modifying module variables</u>

SCREENSHOT #7 paramsModule

The screenshot shows the folder paramModule and its contents, paramModule.c and Makefile.

```
  GNU nano 4.8                                   paramsModule.c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/string.h>
#define DRIVER_AUTHOR "CHEN, Yu-An - 109550073" // Replace with your name and student ID
#define DRIVER_DESC "Example of how to send parameters to Module when loading - OS Project 03"
static char *kernelModuleName = "paramsModule"; //Change module's name when needed
static int studentId = 109550073; // real studentId = 012345, 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 03 - Example 02";
module_param(charparameter, charp, 0644);
MODULE_PARM_DESC(charparameter,"states - Hello world");
```

```
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!";
        }

        printk(KERN_INFO "\n[%s - %s] =============\n",kernelModuleName,__func__);
        printk(KERN_INFO "[%s - %s] Hello!\n",kernelModuleName,__func__);
        printk(KERN_INFO "[%s - %s] Student Id = [%d]\n",kernelModuleName, __func__, studentId);
        printk(KERN_INFO "[%s - %s] String inside module = [%s]\n", kernelModuleName, __func__, charparameter);
        printk(KERN_INFO "[%s - %s] Secret value = [%ld]\n", kernelModuleName, __func__, secretValue);
        return 0;
}

static void clean_exit(void){
        printk(KERN_INFO "\n[%s - %s] =============\n",kernelModuleName,__func__);
        printk(KERN_INFO "[%s - %s] Goodbye!\n",kernelModuleName,__func__);
        printk(KERN_INFO "[%s - %s] Student Id = [%d]\n",kernelModuleName, __func__, studentId);
        printk(KERN_INFO "[%s - %s] String inside module = [%s]\n", kernelModuleName, __func__, charparameter);
        printk(KERN_INFO "[%s - %s] Secret value = [%ld]\n", kernelModuleName, __func__, secretValue);
}
module_init(initialize);
module_exit(clean_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESC);
```

```
  GNU nano 4.8                                          Makefile
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
```

SCREENSHOT #8 load module

The screenshot shows the result of load process on both terminal 1 and terminal 2.

```
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo insmod paramsModule.ko
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo rmmod paramsModule
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo dmesg --clear
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ dmesg -wH
[十二14 16:26]
                [paramsModule - initialize] =============
[  +0.000004] [paramsModule - initialize] Hello!
[  +0.000001] [paramsModule - initialize] Student Id = [109550073]
[  +0.000001] [paramsModule - initialize] String inside module = [Hello world! Project 03 - Example 02]
[  +0.000001] [paramsModule - initialize] Secret value = [987654321]
[十二14 16:28]
                [paramsModule - clean_exit] =============
[  +0.000004] [paramsModule - clean_exit] Goodbye!
[  +0.000000] [paramsModule - clean_exit] Student Id = [109550073]
[  +0.000002] [paramsModule - clean_exit] String inside module = [Hello world! Project 03 - Example 02]
[  +0.000000] [paramsModule - clean_exit] Secret value = [987654321]
```

SCREENSHOT #9 load module with parameter

The screenshot shows the result of load process with parameter on both terminal 1 and terminal 2.

```
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo insmod paramsModule.ko modifyValues=1
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo dmesg --clear
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ dmesg -wH
[十二14 16:35]
                 [paramsModule - initialize] =============
[  +0.000004] [paramsModule - initialize] Hello!
[  +0.000001] [paramsModule - initialize] Student Id = [-1]
[  +0.000002] [paramsModule - initialize] String inside module = [This is a dummy message!]
[  +0.000001] [paramsModule - initialize] Secret value = [-2]
```

SCREENSHOT #10 module information and unload module

The screenshot shows the information of module and the process of unload module on terminal 1, terminal 2 and terminal 3.

```
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo modinfo paramsModule.ko
filename:       /home/usertest109550073/Desktop/Modules/paramsModule/paramsModule.ko
description:    Example of how to send parameters to Module when loading - OS Project 03
author:         CHEN, Yu-An - 109550073
license:        GPL
srcversion:     E10BE0A1FA5385677AE0506
depends:
retpoline:      Y
name:           paramsModule
vermagic:       5.13.19 SMP mod_unload modversions
parm:           studentId:Parameter for student Id. (Leading zeros are omitted) (int)
parm:           secretValue:Parameter for secret value. (long)
parm:           charparameter:states - Hello world (charp)
parm:           modifyValues:Indicates if we must modify the original values or not. (int)
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo rmmod paramsModule
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$
```

```
[十二14 16:37]
                 [paramsModule - clean_exit] =============
[  +0.000006] [paramsModule - clean_exit] Goodbye!
[  +0.000001] [paramsModule - clean_exit] Student Id = [-1]
[  +0.000002] [paramsModule - clean_exit] String inside module = [This is a dummy message!]
[  +0.000001] [paramsModule - clean_exit] Secret value = [-2]
```

SCREENSHOT #11 load module with parameters

The screenshot shows the result of load process with parameters on both terminal 1 and terminal 2

```
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo insmod paramsModule.ko studentId=109550073 secretValu
e=8888
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo dmesg --clear
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ dmesg -wH
[十二14 16:39]
                 [paramsModule - initialize] =============
[  +0.000006] [paramsModule - initialize] Hello!
[  +0.000001] [paramsModule - initialize] Student Id = [109550073]
[  +0.000002] [paramsModule - initialize] String inside module = [Hello world! Project 03 - Example 02]
[  +0.000001] [paramsModule - initialize] Secret value = [8888]
```

SCREENSHOT #12 modify module parameter's value

The screenshot shows the process that we change the parameter's value after the module is loaded.

```
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo nano /sys/module/paramsModule/parameters/secretValue
```

```
GNU nano 4.8
8888
```

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

SCREENSHOT #13 unload module after parameter modification

The screenshot shows the process of unloading module after the parameter is modified on both terminal 1 and terminal 2.

```
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo rmmod paramsModule
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$
```

```
[十二14 16:43]
                 [paramsModule - clean_exit] =============
[  +0.000004] [paramsModule - clean_exit] Goodbye!
[  +0.000002] [paramsModule - clean_exit] Student Id = [109550073]
[  +0.000002] [paramsModule - clean_exit] String inside module = [Hello world! Project 03 - Example 02]
[  +0.000001] [paramsModule - clean_exit] Secret value = [7777]
```

SCREENSHOT #14 load module with invalid parameter

The screenshot shows the result of load process with invalid parameter on both terminal 1 and terminal 2.

```
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$ sudo insmod paramsModule.ko dummystudentId=109550073
usertest109550073@usertest-vm:~/Desktop/Modules/paramsModule$
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ sudo dmesg --clear
usertest109550073@usertest-vm:~/Desktop/Modules/helloModule$ dmesg -wH
[十二14 16:45] paramsModule: unknown parameter 'dummystudentId' ignored
[  +0.000056]
                 [paramsModule - initialize] =============
[  +0.000001] [paramsModule - initialize] Hello!
[  +0.000001] [paramsModule - initialize] Student Id = [109550073]
[  +0.000002] [paramsModule - initialize] String inside module = [Hello world! Project 03 - Example 02]
[  +0.000001] [paramsModule - initialize] Secret value = [987654321]
```

Section 1.3: Dynamically-Loadable Kernel Modules – Example 3: Dynamically loading and unloading a module by user-space application

SCREENSHOT #15 loadUnloadModule

The screenshot shows the file of the folder loadUnloadModule, loaderUnloader.c, paramsModule02.c, and Makefile

```
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ ls -a
.  ..  loaderUnloader.c  Makefile  paramsModule02.c
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$
```

SCREENSHOT #16 loadUnloadModule

The screenshot shows the content of loaderUnloader.c, paramsModule02.c, and Makefile.

```
  GNU nano 4.8                                    loaderUnloader.c                                    Mo
#include <sys/syscall.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#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, param_values, flags)
#define delete_module(name, flags) syscall(__NR_delete_module, name, flags)
// Change your data accordingly
// Author: CHEN, Yu-An
// StudentID: 109550073
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=109550073"; // Use your StudentID without leading 0
        int fd, use_finit;
        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 (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 extension>/parameters
        // WARNING: IF YOU MODIFY THE VARIABLES WITHOUT FOLLOWING THE CORRECT DATATYPE
        // YOUR MODULE WILL GET LOCKED AND YOU MUST 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;
}
```

```
  GNU nano 4.8                                   paramsModule02.c                                   M
#include <linux/init.h>
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/string.h>
#define DRIVER_AUTHOR "Chen, Yu-An - 109550073" // Replace with your name and student ID
#define DRIVER_DESC "Example of how to dynamically load and unload a module from user space - OS Project 03"
static char *kernelModuleName = "paramsModule02"; //Change module's name when needed

static int studentId = 109550073; // real studentId = 012345, 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 03";
module_param(charparameter, charp, 0644);
MODULE_PARM_DESC(charparameter,"states - Hello world");

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!";
        }
        printk(KERN_INFO "\n[%s - %s] =============\n",kernelModuleName,__func__);
        printk(KERN_INFO "[%s - %s] Hello!\n",kernelModuleName,__func__);
        printk(KERN_INFO "[%s - %s] Student Id = [%d]\n",kernelModuleName, __func__, studentId);
        printk(KERN_INFO "[%s - %s] String inside module = [%s]\n", kernelModuleName, __func__, charparameter);
        printk(KERN_INFO "[%s - %s] Secret value = [%ld]\n", kernelModuleName, __func__, secretValue);
        return 0;
}
static void clean_exit(void){
        printk(KERN_INFO "\n[%s - %s] =============\n",kernelModuleName,__func__);
        printk(KERN_INFO "[%s - %s] Goodbye!\n",kernelModuleName,__func__);
        printk(KERN_INFO "[%s - %s] Student Id = [%d]\n",kernelModuleName, __func__, studentId);
        printk(KERN_INFO "[%s - %s] String inside module = [%s]\n", kernelModuleName, __func__, charparameter);
        printk(KERN_INFO "[%s - %s] Secret value = [%ld]\n", kernelModuleName, __func__, secretValue);
}
module_init(initialize);
module_exit(clean_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESC);
```

```
  GNU nano 4.8                                   Makefile
obj-m = paramsModule02.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 #17 load module with loaderUnloader.c

The screenshot shows the process of building and running loaderUnloader.c on terminal 1, terminal 2, and terminal 3.

```
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ gcc -o loaderUnloader loaderUnloader.c
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ make clean
make -C /lib/modules/5.13.19/build M=/home/usertest109550073/Desktop/Modules/loadUnloadModule clean
make[1]: Entering directory '/usr/src/linux-5.13.19'
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ make
make -C /lib/modules/5.13.19/build M=/home/usertest109550073/Desktop/Modules/loadUnloadModule modules
make[1]: Entering directory '/usr/src/linux-5.13.19'
  CC [M]  /home/usertest109550073/Desktop/Modules/loadUnloadModule/paramsModule02.o
  MODPOST /home/usertest109550073/Desktop/Modules/loadUnloadModule/Module.symvers
  CC [M]  /home/usertest109550073/Desktop/Modules/loadUnloadModule/paramsModule02.mod.o
  LD [M]  /home/usertest109550073/Desktop/Modules/loadUnloadModule/paramsModule02.ko
make[1]: Leaving directory '/usr/src/linux-5.13.19'
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ sudo ./loaderUnloader

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=109550073]...
Module is mounted!

[Press ENTER to continue]
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ sudo dmesg --clear
[sudo] password for usertest109550073:
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ dmesg -wH
[十二14 17:03]
                 [paramsModule02 - initialize] =============
[  +0.000007] [paramsModule02 - initialize] Hello!
[  +0.000001] [paramsModule02 - initialize] Student Id = [109550073]
[  +0.000002] [paramsModule02 - initialize] String inside module = [Hello world! Project 02 - Example 03]
[  +0.000001] [paramsModule02 - initialize] Secret value = [987654321]
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ ls /sys/module/paramsModule02/parameters/
charparameter  modifyValues  secretValue  studentId
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ lsmod | grep paramsModule02
paramsModule02         16384  0
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$
```

SCREENSHOT #18 unload module with loaderUnloader.c

The screenshot shows the process of unloading module with loaderUnloader.c on terminal 1, terminal 2 and terminal 3.

```
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ sudo ./loaderUnloader

This is a dynamic loader and unloader for a kernel module!
Loading module [paramsModule02] with parameters [studentId=109550073]...
Module is mounted!

[Press ENTER to continue]

Unmounting module...
Module is unmounted!
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ dmesg -wH
[十二14 17:03]
                 [paramsModule02 - initialize] =============
[  +0.000007] [paramsModule02 - initialize] Hello!
[  +0.000001] [paramsModule02 - initialize] Student Id = [109550073]
[  +0.000002] [paramsModule02 - initialize] String inside module = [Hello world! Project 02 - Example 03]
[  +0.000001] [paramsModule02 - initialize] Secret value = [987654321]
[十二14 17:06]
                 [paramsModule02 - clean_exit] =============
[  +0.000007] [paramsModule02 - clean_exit] Goodbye!
[  +0.000001] [paramsModule02 - clean_exit] Student Id = [109550073]
[  +0.000003] [paramsModule02 - clean_exit] String inside module = [Hello world! Project 02 - Example 03]
[  +0.000001] [paramsModule02 - clean_exit] Secret value = [987654321]
```

```
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ ls /sys/module/paramsModule02/parameters/
ls: cannot access '/sys/module/paramsModule02/parameters/': No such file or directory
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$ lsmod | grep paramsModule02
usertest109550073@usertest-vm:~/Desktop/Modules/loadUnloadModule$
```

Section 1.4: Dynamically-Loadable Kernel Modules – Final Exercise: Simple calculator

SCREENSHOT #19 calculatorModule

The screenshot shows the content of the folder calculatorModule, calculator.c, calculatorModule.c, and Makefile.

For calculator.c, it judges the input argv[] and do the operations. Addition(), subtraction(), and multiplication() first allocate space for paramsNew. Then they call SetParamString(), LoadModule(), GetResult(), and UnloadModule().

SetParamString() make three input parameters into one single string for passing it to the module. LoadModule() mounts the module with the string paramsNew that we just set. GetResult() get the result by checking the file where it stores the value of resultParam. Finally, UnloadModule() unmounts the module and free the module space.

```
  GNU nano 4.8                                    calculator.c                                    M
#include <stdio.h>
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
//ADD ADDITIONAL DEFINES HERE
#define init_module(module_image, len, param_values) syscall(__NR_init_module, module_image, len, param_values)
#define delete_module(name, flags) syscall(__NR_delete_module, name, flags)

int addition(long* result, int input1, int input2);
int substraction(long* result, int input1, int input2);
int multiplication(long* result, int input1, int input2);
int LoadModule(const char* params);
int UnLoadModule();
void SetParamString(char* parm, int input1, int input2, const char* op);
long GetResult();
// StudentID = 109550073 // replace with your student ID
// ADD ADDITIONAL VARIABLES HERE
// You must load and unload the "calculatorModule.ko" module.
const char *moduleName = "calculatorModule.ko";
const char *moduleNameNoExtension = "calculatorModule";
int fd, use_finit;
size_t image_size;
struct stat st;
void *image;
int main(int argc, char** argv){
    long result = -9999999;
    long resultError = -9999999;
    if (argc != 4) {
        printf("%ld\n", resultError);
        return resultError;
    }
    int param01 = atoi(argv[1]);
    int param02 = atoi(argv[2]);
    int operationError = 1;
```

```c
    if (strcmp(argv[3], "add") == 0) {
        operationError = addition(&result, param01, param02);
    }
    else if (strcmp(argv[3], "sub") == 0) {
        operationError = substraction(&result, param01, param02);
    }
    else if (strcmp(argv[3], "mul") == 0) {
        operationError = multiplication(&result, param01, param02);
    }
    if (operationError == EXIT_SUCCESS) {
        printf("%ld\n", result);
    }
    else {
        printf("%ld\n", resultError);
    }
    return 0;
```

```c
int addition(long* result, int input1, int input2)
{
    *result = 0;
    int operationError = 0;
    //INSERT YOUR CODE HERE
    char *paramsNew = malloc(100);
    SetParamString(paramsNew, input1, input2, "add");
    LoadModule(paramsNew);
    *result = GetResult();
    UnLoadModule();
    return operationError;
}
int substraction(long* result, int input1, int input2)
{
    *result = 0;
    int operationError = 0;
    //INSERT YOUR CODE HERE
    char *paramsNew = malloc(100);
    SetParamString(paramsNew, input1, input2, "sub");
    LoadModule(paramsNew);
    *result = GetResult();
    UnLoadModule();
    return operationError;
}
int multiplication(long* result, int input1, int input2)
{
    *result = 0;
    int operationError = 0;
    //INSERT YOUR CODE HERE
    char *paramsNew = malloc(100);
    SetParamString(paramsNew, input1, input2, "mul");
    LoadModule(paramsNew);
    *result = GetResult();
    UnLoadModule();
    return operationError;
}
```

```c
void SetParamString(char* parm, int input1, int input2, const char* op) {
    //INSERT YOUR CODE HERE
    sprintf(parm, "firstParam=%d secondParam=%d operationParam=%s", input1, input2, op);
}
int LoadModule(const char* params) {
    //INSERT YOUR CODE HERE
    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");
        return EXIT_FAILURE;
    }
    return 0;
}
long GetResult() {
    long result = 0;
    //INSERT YOUR CODE HERE
    FILE *fptr = fopen("/sys/module/calculatorModule/parameters/resultParam", "r");
    if (fptr == NULL){
       printf("Error! opening file"); // Program exits if the file pointer returns NULL.
       exit(1);
   }
    fscanf(fptr,"%ld", &result);
    fclose(fptr);
    return result;
}
int UnLoadModule() {
    //INSERT YOUR CODE HERE
    if (delete_module(moduleNameNoExtension, O_NONBLOCK) != 0) {
        perror("delete_module");
        return EXIT_FAILURE;
    }
    close(fd);
    free(image);
    return 0;
}
```

For calculatorModule.c, its parameters are changed by the paramsNew sending by calculator.c. It judges the paramsNew and do the specific work and calculate the answer by operationParam.

```
  GNU nano 4.8                                calculatorModule.c                                Modified
#include <linux/init.h>
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/string.h>
#define DRIVER_AUTHOR "CHEN, Yu-An - 109550073" // Replace with your name and student ID
#define DRIVER_DESC "Dynamic calculator - OS Project 03"
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: 'add' - addition / 'sub' - substraction / 'mul' - multiplicatio>

static long resultParam = -1;
module_param(resultParam, long, 0644);
MODULE_PARM_DESC(resultParam, "Result parameter for operation.");
static int initialize(void){
    printk(KERN_INFO "\n[%s - %s] =============\n",kernelModuleName,__func__);
    printk(KERN_INFO "[%s - %s] Hello from calculatorModule!\n",kernelModuleName,__func__);
    // INSERT YOUR CODE HERE
    if(strcmp(operationParam, "add") == 0){
        operationParam = "add";
        resultParam = firstParam + secondParam;
    }
    else if(strcmp(operationParam, "sub") == 0){
        operationParam = "sub";
        resultParam = firstParam - secondParam;
    }
    else if(strcmp(operationParam, "mul") == 0){
        operationParam = "mul";
        resultParam = firstParam * secondParam;
    }
    else{
        resultParam = -9999999;
        return 0;
    }
    printk(KERN_INFO "[%s - %s] Operation = %s\n", kernelModuleName, __func__, operationParam);
    printk(KERN_INFO "[%s - %s] First parameter = %d\n", kernelModuleName, __func__, firstParam);
    printk(KERN_INFO "[%s - %s] Second parameter = %d\n", kernelModuleName, __func__, secondParam);
    printk(KERN_INFO "[%s - %s] Result = %ld\n", kernelModuleName, __func__, resultParam);
    return 0;
}
static void clean_exit(void){
    printk(KERN_INFO "\n[%s - %s] =============\n",kernelModuleName,__func__);
    printk(KERN_INFO "[%s - %s] Goodbye from calculatorModule!\n", kernelModuleName,__func__);
    printk(KERN_INFO "[%s - %s] Operation = %s\n", kernelModuleName,__func__,operationParam);
    printk(KERN_INFO "[%s - %s] First parameter = %d\n", kernelModuleName,__func__,firstParam);
    printk(KERN_INFO "[%s - %s] Second parameter = %d\n", kernelModuleName,__func__,secondParam);
    printk(KERN_INFO "[%s - %s] Result = %ld\n", kernelModuleName,__func__,resultParam);
}

module_init(initialize);
module_exit(clean_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESC);
```

SCREENSHOT #20 dynamically load and unload calculatorModule by calculator.c

The screenshot shows the result running calculator.c. We can see that it runs smoothly with correct operation parameters and returns -9999999 with invalid parameters.

```
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$ sudo ./calculator 20 35 add
55
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$ sudo ./calculator 15 7 sub
8
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$ sudo ./calculator 10 35 sub
-25
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$ sudo ./calculator 5 10 mul
50
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$ sudo ./calculator 5 10 test
-9999999
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$ sudo ./calculator
-9999999
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$ sudo ./calculator -5 7 mul
-35
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$ sudo ./calculator 2 3
-9999999
usertest109550073@usertest-vm:~/Desktop/Modules/calculatorModule$
```

```
                 [calculatorModule - initialize] =============
[  +0.000023] [calculatorModule - initialize] Hello from calculatorModule!
[  +0.000019] [calculatorModule - initialize] Operation = add
[  +0.000056] [calculatorModule - initialize] First parameter = 20
[  +0.000020] [calculatorModule - initialize] Second parameter = 35
[  +0.000019] [calculatorModule - initialize] Result = 55
[  +0.001232]
              [calculatorModule - clean_exit] =============
[  +0.000021] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[  +0.000019] [calculatorModule - clean_exit] Operation = add
[  +0.000019] [calculatorModule - clean_exit] First parameter = 20
[  +0.000239] [calculatorModule - clean_exit] Second parameter = 35
[  +0.000019] [calculatorModule - clean_exit] Result = 55
[  +8.437797]
              [calculatorModule - initialize] =============
[  +0.000025] [calculatorModule - initialize] Hello from calculatorModule!
[  +0.000020] [calculatorModule - initialize] Operation = sub
[  +0.000020] [calculatorModule - initialize] First parameter = 15
[  +0.000020] [calculatorModule - initialize] Second parameter = 7
[  +0.000019] [calculatorModule - initialize] Result = 8
[  +0.000459]
              [calculatorModule - clean_exit] =============
[  +0.000022] [calculatorModule - clean_exit] Goodbye from calculatorModule!
              [calculatorModule - initialize] =============
[  +0.000024] [calculatorModule - initialize] Hello from calculatorModule!
[  +0.000020] [calculatorModule - initialize] Operation = sub
[  +0.000019] [calculatorModule - initialize] First parameter = 10
[  +0.000020] [calculatorModule - initialize] Second parameter = 35
[  +0.000019] [calculatorModule - initialize] Result = -25
[  +0.000469]
              [calculatorModule - clean_exit] =============
[  +0.000022] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[  +0.000019] [calculatorModule - clean_exit] Operation = sub
[  +0.000019] [calculatorModule - clean_exit] First parameter = 10
[  +0.000020] [calculatorModule - clean_exit] Second parameter = 35
[  +0.000019] [calculatorModule - clean_exit] Result = -25
[十二25 16:35]
                 [calculatorModule - initialize] =============
[  +0.000014] [calculatorModule - initialize] Hello from calculatorModule!
[  +0.000012] [calculatorModule - initialize] Operation = mul
[  +0.000011] [calculatorModule - initialize] First parameter = 5
[  +0.000011] [calculatorModule - initialize] Second parameter = 10
[  +0.000012] [calculatorModule - initialize] Result = 50
[  +0.000756]
              [calculatorModule - clean_exit] =============
[  +0.000012] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[  +0.000011] [calculatorModule - clean_exit] Operation = mul
[  +0.000011] [calculatorModule - clean_exit] First parameter = 5
[  +0.000011] [calculatorModule - clean_exit] Second parameter = 10
[  +0.000012] [calculatorModule - clean_exit] Result = 50
[ +26.790047]
```

```
                [calculatorModule - initialize] =============
[  +0.000018] [calculatorModule - initialize] Hello from calculatorModule!
[  +0.000058] [calculatorModule - initialize] Operation = mul
[  +0.000013] [calculatorModule - initialize] First parameter = -5
[  +0.000013] [calculatorModule - initialize] Second parameter = 7
[  +0.000013] [calculatorModule - initialize] Result = -35
[  +0.001597]
                [calculatorModule - clean_exit] =============
[  +0.000020] [calculatorModule - clean_exit] Goodbye from calculatorModule!
[  +0.000030] [calculatorModule - clean_exit] Operation = mul
[  +0.000014] [calculatorModule - clean_exit] First parameter = -5
[  +0.000015] [calculatorModule - clean_exit] Second parameter = 7
[  +0.000014] [calculatorModule - clean_exit] Result = -35
```

2. Questions

2.1. 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)?

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.

System calls are requests to Kernel from user, executed sequentially, and cannot be interrupted, while kernel modules are a part of OS that control access to resources, not executed sequentially, and can be interrupted.

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

Because system call is like a program which needs **link to the kernel** for specific resources, it needs re-compilation. Kernel module is a part of OS. If we load a kernel, we just have to compile the kernel instead of recompiling the whole kernel space.

2.3. 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).

insmod load the module, **rmmod** remove the module, and **modinfo** acquire the module's information.

Load dummyModule: **sudo insmod dummyModule.ko**
Remove dummyModule: **sudo rmmod dummyModule.ko**
Acquire dummyModule information: **sudo modinfo dummyModule.ko**

2.4. 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

a. **module_init** (static int initialize(void)) -> macro -> module initialization entry point
b. **module_exit**(static void clean_exit(void)) -> macro -> module unmounting point
c. **MODULE_LICENSE**(program license) -> macro -> declare module license
d. **module_param**(parameter name, parameter type, access right) -> macro -> declare module's parameter
e. **MODULE_PARM_DESC**(parameter description) -> macro -> declare parameter description

2.5. 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

a. **cat** -> print the content of a file onto the standard output stream
   -x -> Option not available
b. **ls** -> lists the contents of current working directory
   -a -> do not ignore entries starting with .
   -x -> list entries by lines instead of by columns.
c. **dmesg** -> print or control the kernel ring buffer
   -wH -> Wait for new messages and Enable human-readable output
   -x -> Decode facility and level (priority) numbers to human-readable prefixes.
d. **lsmod** -> shows what kernel modules are currently loaded
   -x -> Option not available
e. **lsmod | grep** -> shows what kernel modules are currently loaded and print lines that match patterns
   -x -> Select only those matches that exactly match the whole line.

2.6. There is a 0644 in the line module_param(studentId, int, 0644); inside paramsModule.c (Section 1.2). What does 0644 mean?

0644 is the **access right** for the parameter. They will be readable by all the user groups, but writable by the user only.

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

When it returns -1, **an error is occurred** because of the following error type:
EBADMSG, EBUSY, EFAULT, ENOKEY, ENOMEM, EPERM, EEXIST, EINVAL, ENOEXEC.

2.8. <u>In Section 1.2 – step 6, modinfo shows the information of some variables inside the module but two of them are not displayed. Why is it?</u>

Because they **did not define the description with MODULE_PARM_DESC()**

2.9. <u>What is the /sys/module folder for?</u>

It is the folder for **collecting the entry about all loaded modules and their parameters** on the device.

2.10. <u>In Section 1.2 (paramsModule.c), the variable charparameter is of type charp. What is charp?</u>

It is **character pointer.**

2.11. <u>Which project (01 / 02 / 03) did you like the most? Why?</u>

I like project 03 the most since it does not evolve kernel recompilation, which really takes a lot of time ☹.

2.12. <u>Which project (01 / 02 / 03) did you like the least? Why?</u>

It seems to be project 01 since it contains so many steps and it almost has no relationship with follow-up projects.

2.13. <u>Did you learn anything new with these three projects? What did you learn?</u>

Sure. Not only get familiar with more linux commands but also have basic concept and experience with linux kernel, which is really a precious opportunity.

2.14. <u>Do you think these projects can help you in the future, if you look for a job in the industry?</u>

Yes, if I look for a job relating to OS/kernel development.