**Operating Systems Project Report**

|  |  |
| --- | --- |
| **Project Number (01 / 02 / 03):** | 03 |
| **Name:** | 蕭望緯 |
| **Student ID:** | 0811521 |
| **YouTube link (Format youtube.com/watch?v=[key]):** | <https://youtu.be/GY-x0657k4I> |
| **Date (YYYY-MM-DD):** | 2021-12-11 |
| **Names of the files**  **uploaded to E3:** | OS\_Project03\_0811521.pdf |
| **Physical Machine Total RAM (Example: 8.0 GB):** | 16GB |
| **Physical Machine CPU (Example: Intel i7-2600K):** | 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz |

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

# Individual questions

1. What is a static kernel module?

What is a dynamic kernel module?

What is the other name of a dynamic kernel module?

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

Ans:

a. compiled as part of the base kernel and available anytime

b. compiled separately and dynamically loaded when needed

c. Loadable Kernel Modules (LKM)

d. using LKM does not require recompiling the entire kernel, while adding system calls is otherwise.

LKMs can be loaded and unloaded based on the demand, so the memory is efficiently utilized, whereas system calls occupy the memory once they are installed into the kernel.

LKMs run slower than system calls.

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

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

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

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

Ans:

**insmod**: load a module into the kernel

**rmmod**: remove a module from the kernel

**modinfo**: display information(attributes) of a module

**insmod dummyModule.ko**

**rmmod dummyModule**

**modinfo dummyModule.ko**

* 1. 4. Write the usage (parameters, what data type they are and what do they do) of the following commands:
  2. **a. module\_init**
  3. **b. module\_exit**
  4. **c. MODULE\_LICENSE**
  5. **d. module\_param**
  6. **e. MODULE\_PARM\_DESC**

1. Ans:
2. a. **module\_init(*x*)**
3. x(function): function to be called at module insertion time
4. b. **module\_exit(*x*)**
5. x(function): function to be called at module insertion time

c. **MODULE\_LICENSE(x)**

1. x(string): license name (e.g., “GPL”, “Dual BSD/GPL”)
2. d. **module\_param(name, type, perm)**
3. name: variable name
4. type: its type (e.g., int, bool, etc.)
5. perm(int): permissions for the corresponding file in sysfs
6. e. **MODULE\_PARM\_DESC(name, desc)**
7. name: variable name

desc(string): description of the variable

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

**a. cat**

* 1. **b. ls -l**
  2. **c. dmesg -wH**
  3. **d. lsmod**
  4. **e. lsmod | grep**

Ans:

a. read or write content to files

b. list file in a directory. **-l:** show file or directory, size, modified date and time, file or folder name and owner of the file, and its permission.

c. read messages stored in the ring buffer. **-H:** enable human-readable output. **-w**: Wait for new messages.

d. show what kernel modules are currently loaded.

e. search with keywords and show whether related kernel modules are currently loaded

* 1. 6. There is a 0644 in the line

**module\_param(studentId, int, 0644);**

inside **paramsModule.c** (Section 1.2). What does 0644 mean?

Ans: owner can read and write, group can read, and everyone else can read.

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

What type of error do you get?

Ans: show loading error messages; operation not permitted.

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

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

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

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

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

Ans: string

**Additional questions (also answer in the report):**

11. Which project (01 / 02 / 03) did you like the most? Why?

Project 3. I do not need to rebuild the kernel, which takes a lot of time.

12. Which project (01 / 02 / 03) did you like the least? Why?

Project 1. I must wait for a while when the kernel is rebuilt. However, Project 1 lays the foundation of knowledge, so I think it is worth going through the process.

13. Did you learn anything new with these three projects? What did you learn?

Literally everything in the projects is new for me. I learn about how to work on kernels and also gain experience with Linux environment.

14. Do you think these projects can help you in the future, if you look for a job in the industry?

Even though I might not be researching and programming on operating systems in the near future, it is good to be knowledgeable about the wheels under them as a programmer.

# Screenshots

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

Graphical user interface, application

Description automatically generated

#2 compile the module

Text

Description automatically generated

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

Text

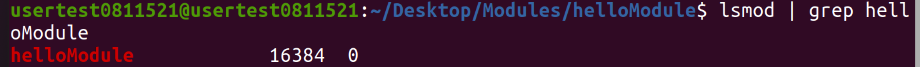
Description automatically generated

#4 the list of loaded modules

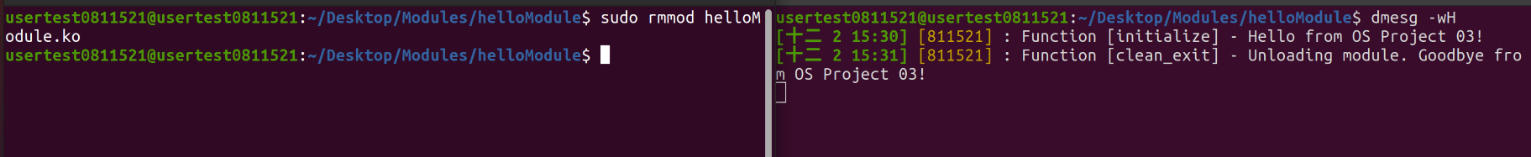
Text

Description automatically generated

search the list with keywords



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



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

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application, chat or text message

Description automatically generated

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

Graphical user interface, text

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

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

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

Text

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application, chat or text message

Description automatically generated

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

Text

Description automatically generated

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

Graphical user interface, text

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

#15

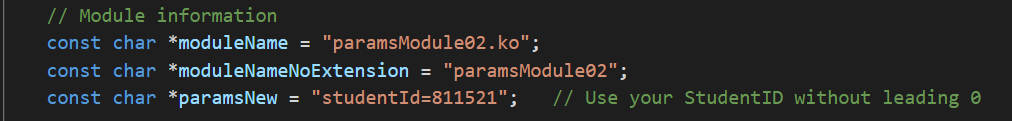
**loaderUnloader.c** is the program that loads and unloads a module. **paramsModule02.c** is the source code of the module.

Graphical user interface, application

Description automatically generated

#16

**paramsNew** contains the parameters to be passed into the module.



load the module.

Text

Description automatically generated

wait for input and the module is loaded.

Text

Description automatically generated

unload the module.

Text

Description automatically generated

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

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

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

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

Text

Description automatically generated

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

Text

Description automatically generated

#19

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

* call **setParamString()** to make a string argument of parameters
* call **LoadModule()** to load, initialize and pass the parameters to the module
* call **GetResult()** to read the result value from **/sys/module/calculatorModule/parameters/resultParam**
* call **UnloadModule()** to unload the module

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

Text

Description automatically generated

#20

Execution Results

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

Text

Description automatically generated

Results of normal operations and kernel messages in terminal T2.

A screenshot of a computer

Description automatically generated with medium confidence