PA1 - Part B

Start Assignment

Due Sep 12 by 11:59pm **Points** 25 **Submitting** a file upload **File Types** c
Available until Sep 12 at 11:59pm

Programming Assignment One - Part B

Writing a second system call

Assuming you've gotten hello_world() to work from Part A, you now have to write a second system call. This system call will be given two numbers and an address of where to store the result.

Name this new system call **cs3753_add()** and define three arguments: **number1**, **number2**, and the pointer to **result**.

As with hello_world, you will need to write a C test program that calls this new system call and passes the correct types of arguments.

Your cs3753_add() must do the following:

- use printk() to log the numbers to be added
- add those two numbers
- store the result
- use printk() to log the result
- return an appropriate return value

Additionally, have your user space test program print out it's result as well.

Simple Loadable Kernel Module

As we saw in the previous parts, if you make a change to the kernel, you must recompile and then reboot before these changes become active. This approach is time-consuming and can be painstaking. In contrast, LKMs add functionality to the OS without the need to reboot or recompile the entire kernel. In short, LKMs are object files that can be used to extend a running kernel's functionality on the fly.

LKM Example

Create a directory, **/home/kernel/modules**, and edit a new file named **helloModule.c**. Populate this file with the following code:

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

MODULE_AUTHOR("Your Name");
MODULE_LICENSE("GPL");

int hello_init(void) {
    printk(KERN_ALERT "inside %s function\n",__FUNCTION__);
    return 0;
}

void hello_exit(void) {
    printk(KERN_ALERT "inside %s function\n",__FUNCTION__);
}

module_init(hello_init);
module_exit(hello_exit);
```

Notice a few things about the above:

- **<linux/init.h>** & **<linux/module.h>** contain the library headers for module initialization and other functions that support LKMs
- **module_init()** and **module_exit()** bind two functions, **hello_init()** and **hello_exit()**, to be executed when the module is installed and uninstalled
- As with PA1, you cannot use user space functions such as **printf()**, use **printk()** with **KERN_ALERT** to write messages to **/var/log/syslog**

We'll now use a makefile to build the module. Create a file named **Makefile** in **/home/kernel/modules** and add the following line to it:

```
obj-m:=helloModule.o
```

In this line, **obj-m** means module, and the line as a whole tells the compiler to create a module object named **helloModule.o**

To compile the module enter the following command:

```
make -C /lib/modules/$(uname -r)/build M=$PWD
```

and if the compilation is successful, your output should be something similar to:

```
make: Entering directory '/home/kernel/linux-hwe-4.15.0'
CC [M] /home/kernel/modules/helloModule.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/kernel/modules/helloModule.mod.o
LD [M] /home/kernel/modules/helloModule.ko
make: Leaving directory '/home/kernel/linux-hwe-4.15.0'
```

You can now list out the contents of the **/home/kernel/modules** directory with the **ls -l** command:

```
user@csci3753-vm:/home/kernel/modules$ ls -l
total 276
-rw-rw-r-- 1 user user    331 Sep 12 13:04 helloModule.c
-rw-rw-r-- 1 user user 127232 Sep 12 13:51 helloModule.ko
-rw-rw-r-- 1 user user    603 Sep 12 13:51 helloModule.mod.c
-rw-rw-r-- 1 user user  72584 Sep 12 13:51 helloModule.mod.o
-rw-rw-r-- 1 user user  57864 Sep 12 13:51 helloModule.o
-rw-rw-r-- 1 user user     21 Sep 12 12:56 Makefile
-rw-rw-r-- 1 user user     43 Sep 12 13:51 modules.order
-rw-rw-r-- 1 user user      0 Sep 12 13:51 Module.symvers
```

Notice the file named **helloModule.ko**. This is the kernel module (.ko) object you will be inserting into the running kernel.

Install the module by executing **sudo insmod helloModule.ko**. Now when you use **lsmod**, you should see that your module is installed:

```
user@csci3753-vm:/home/kernel/modules$ sudo insmod helloModule.ko
user@csci3753-vm:/home/kernel/modules$ lsmod | grep hello
helloModule 16384 0
```

Run **dmesg** or **sudo tail /var/log/syslog** to see the message emitted by your kernel module when it initialized:

```
[ 2643.230212] inside hello_init function
```

Now remove the module by typing **sudo rmmod helloModule**. If you again enter the **lsmod** command, you will see that your module is no longer listed. Type **dmesg** to see if the expected output from unloading the module is printed:

```
[ 2657.344579] inside hello_exit function
```

If the all the above checks out, you've successfully completed the first and most basic part of the assignment. Now we can move onto the next part, where you will write a user space test program.

Submission

You are required to upload the following to Canvas by the due date:

- ./arch/x86/kernel/cs3753_add.c
- We'll confirm the operation of your LKM and system call by looking at your Cloud VM.