William Pan ID: 922867228
Github: WilliamPanTW CSC415-01 Operating Systems

Assignment 6 – Device Driver

Description:

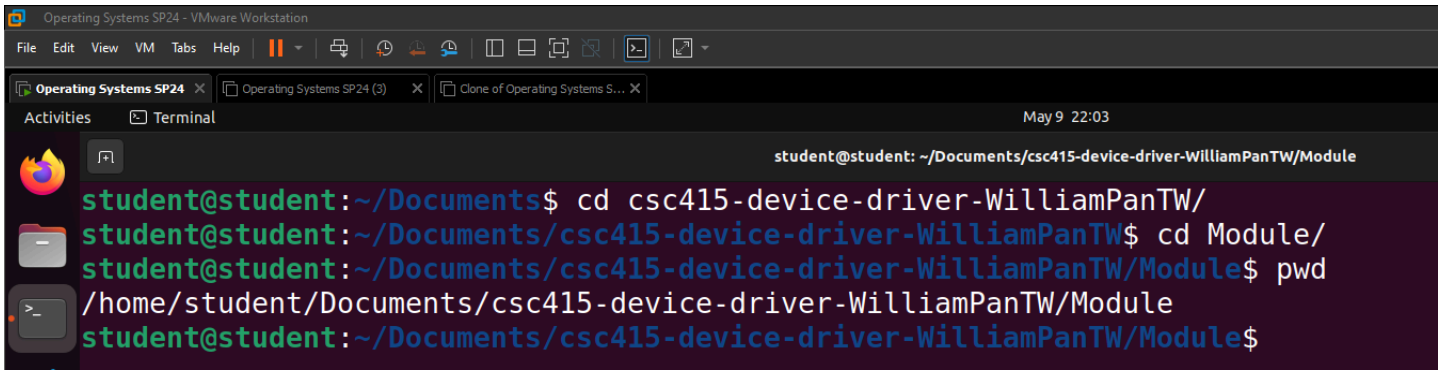
This assignment I have written a simple Caesar cipher device driver named "Cat" that manages a software-based device module in linux. It supports basic file operations like open, release, read, and write, as well as custom IOCTL commands for encryption and decryption. The driver implements functions for encrypting and decrypting messages by shifting characters 2 spots. And it allocates and deallocates memory for device data during open and close operations. Finally, it registers the device with the kernel during initialization and unregisters it during cleanup. With the "Pan_William_HW6_main.c" file, the user has the interface to interact with the Caesar cipher device driver named "Cat". Which prompts users with options of either encrypting or decrypting a message, or to exit the program. After selecting an option and input message, it will then be passed to the kernel module for encryption or decryption.

William Pan
Github: WilliamPanTW

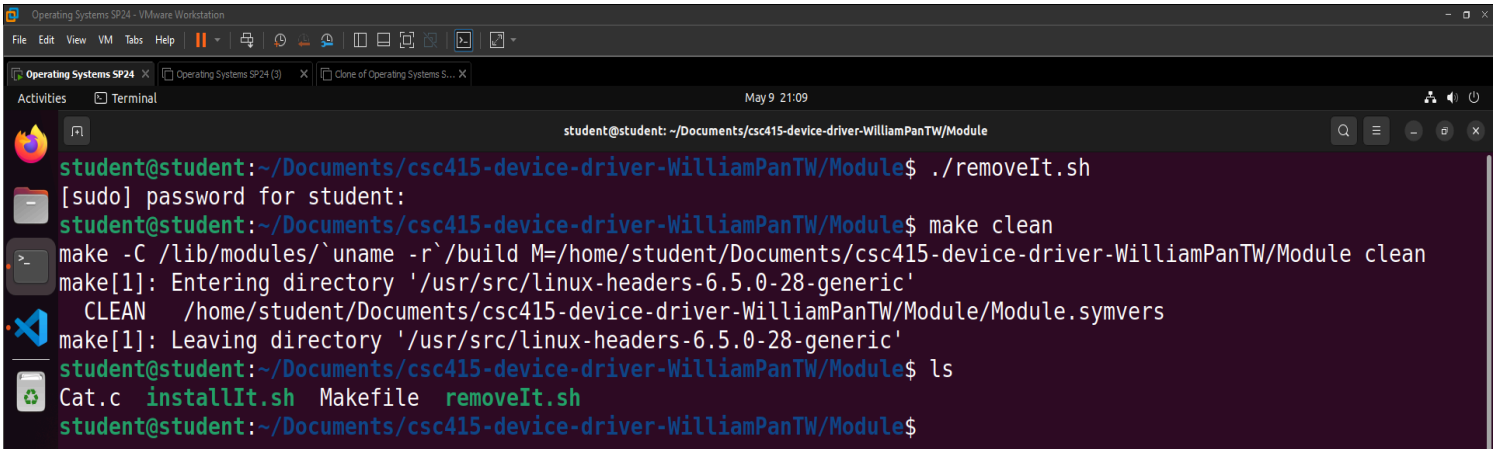
ID: 922867228
CSC415-01 Operating Systems

How to load and build the device driver:

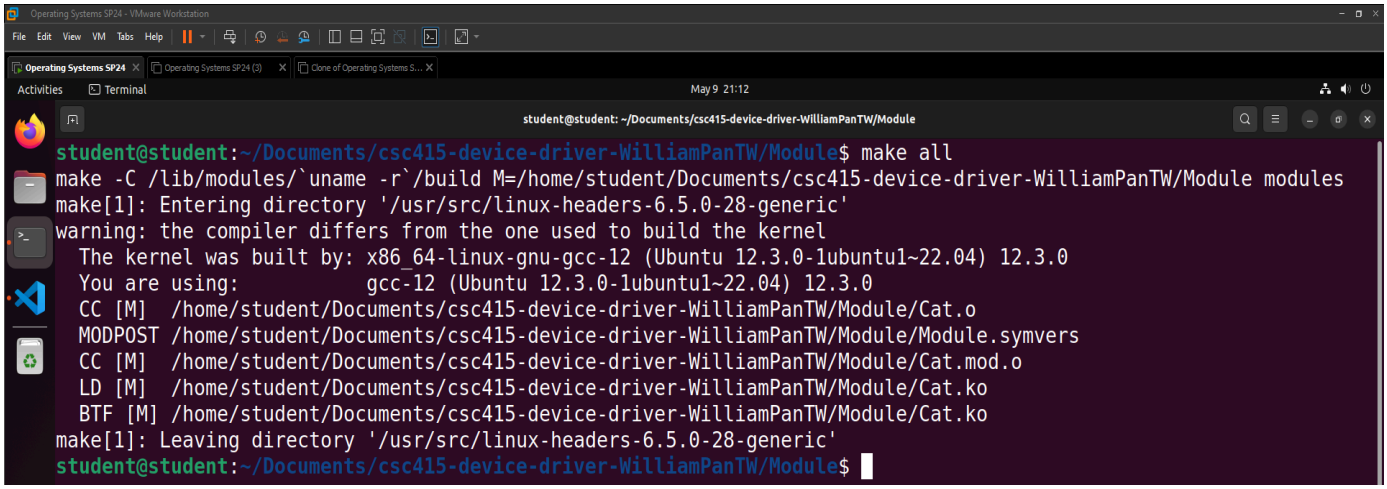
Step 1: Navigate to the "csc415-device-driver-WilliamPanTW/Module" directory.



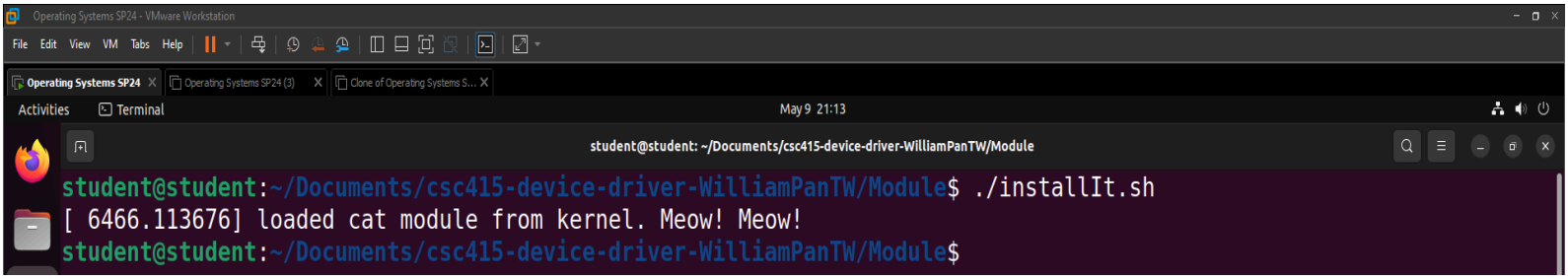
Step 2: Run removeIt script if necessary enter password, then run "$ make clean".



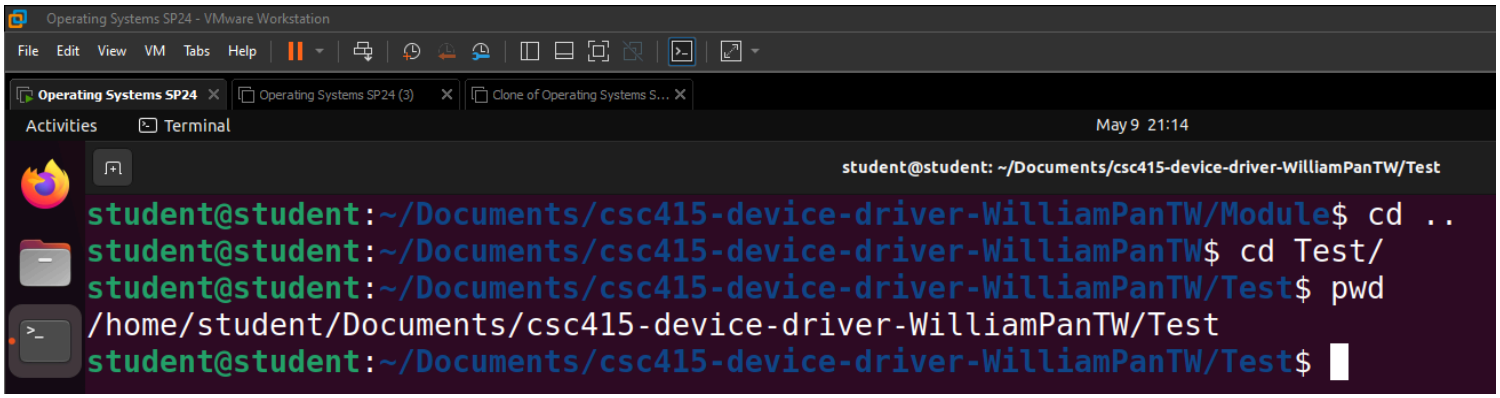
Step 3. Making sure “Cat.c , installIt.sh , Makefile and removeIt.sh” exist, then run “$ make all”



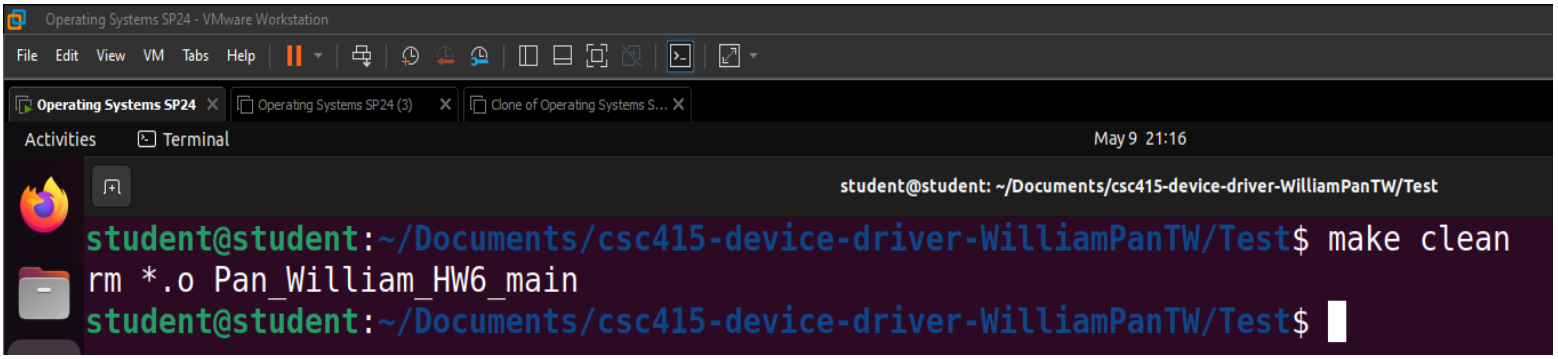
Step 4. Run installIt script with “ $ ./installIt.sh”, if necessary enter password



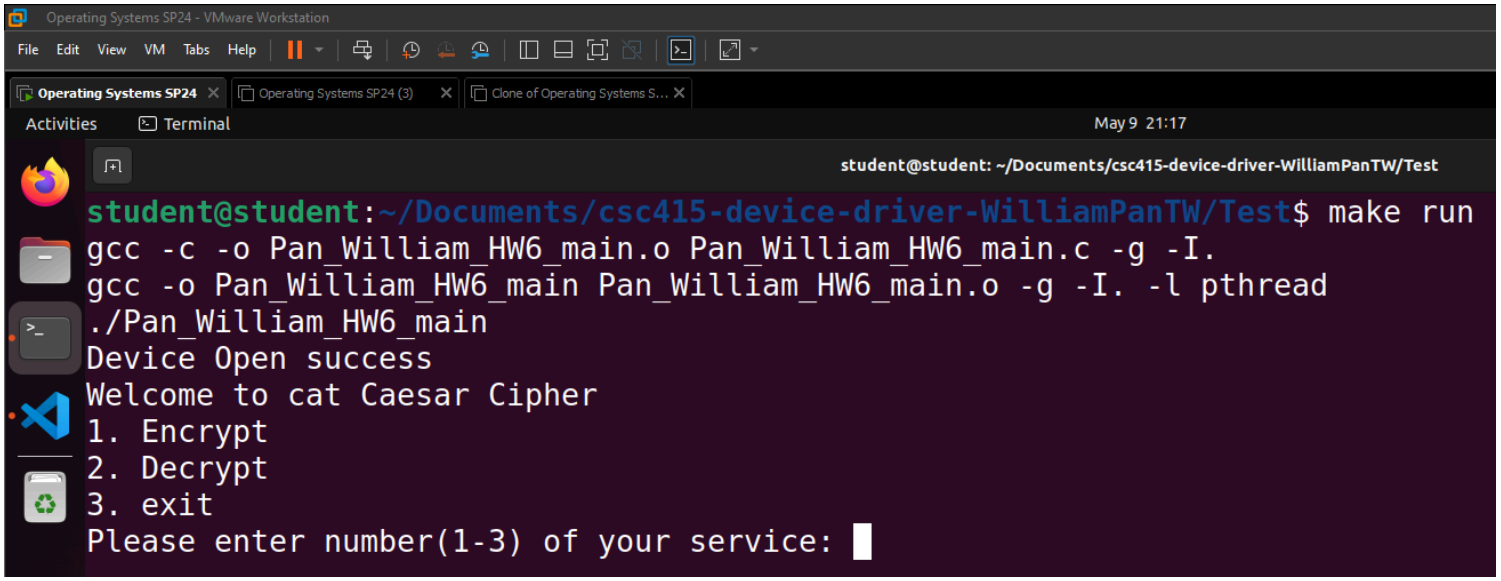
Step 5. Navigate to the "csc415-device-driver-WilliamPanTW/Test" directory.



Step 6. In the Test directory Run " $ make clean ".



Step 7. Run " $make run " in Test directory and you can start interact with the device driver



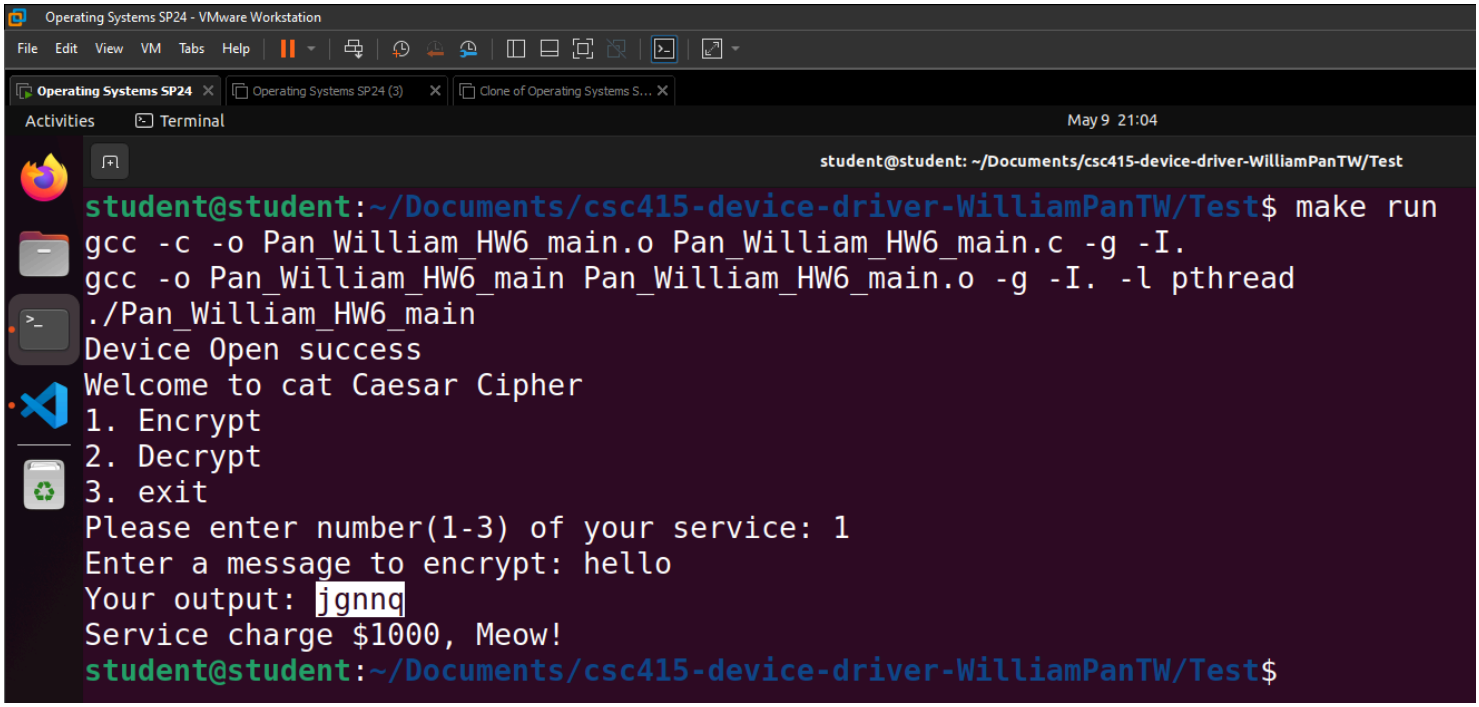
William Pan
Github: WilliamPanTW

ID: 922867228
CSC415-01 Operating Systems

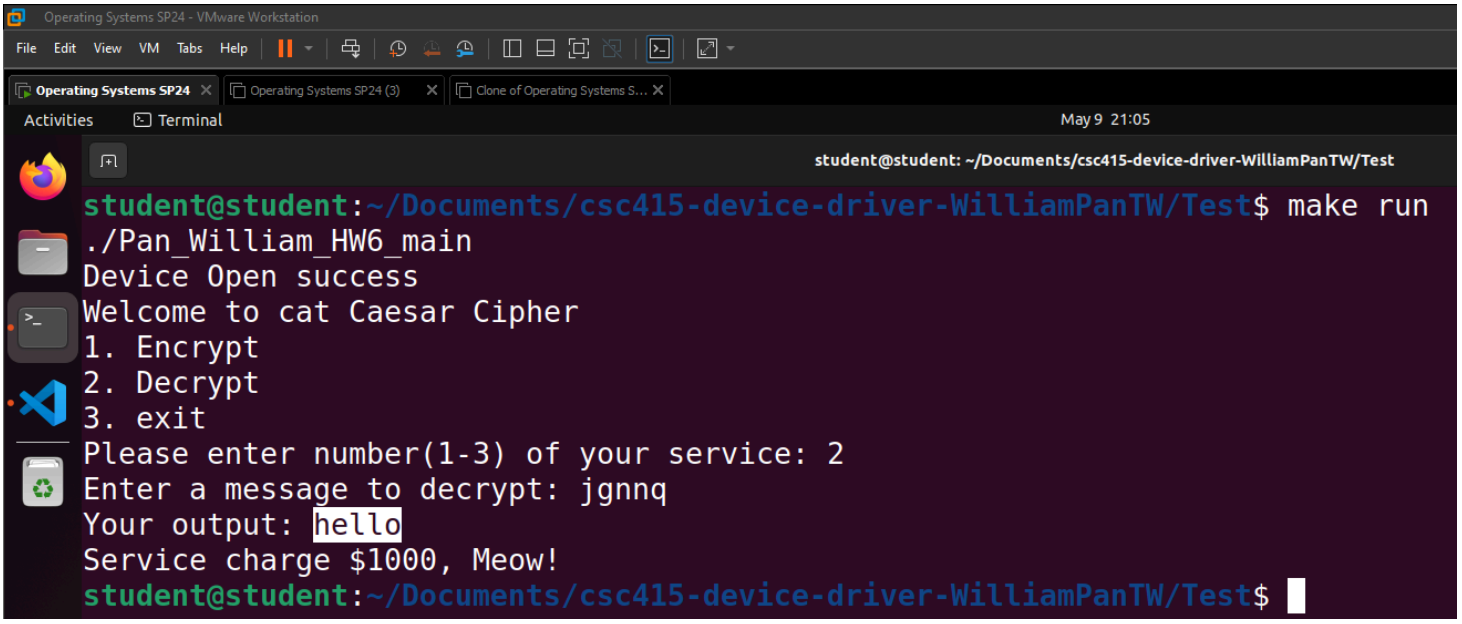
How to interact with the device driver

Step 1. Enter the function you want in number, “1” for Encrypt , “2” for decrypt , "3” for exit

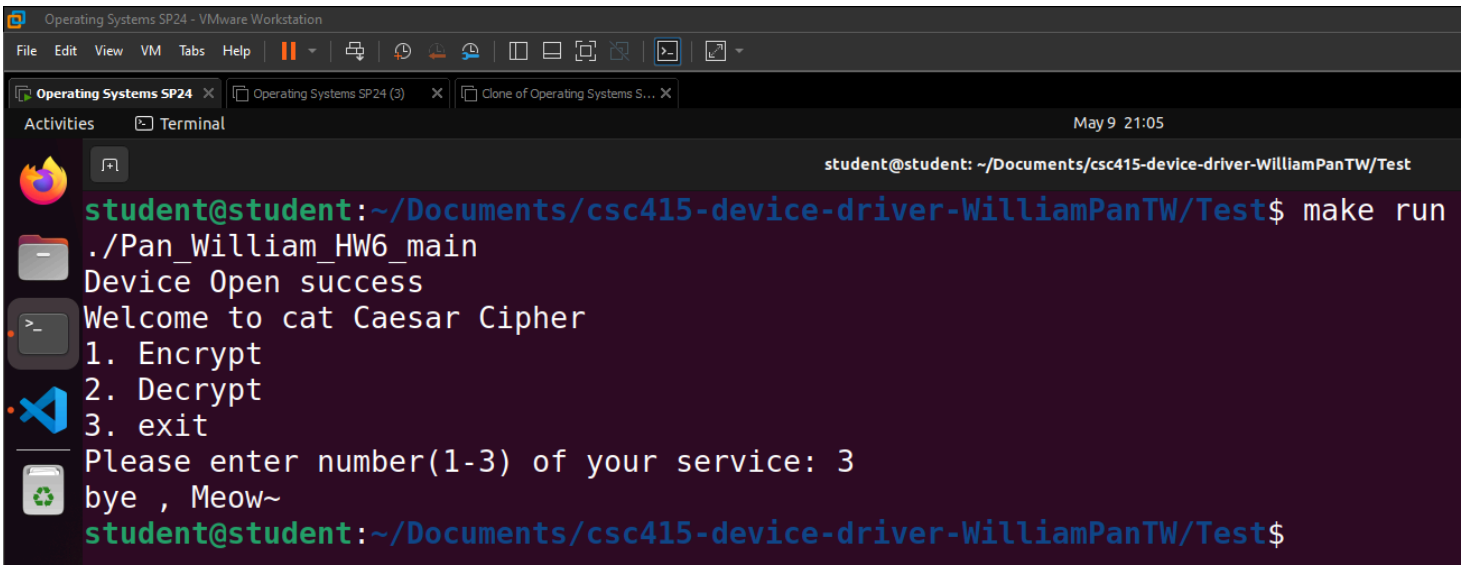
Step 2. If enter “1” you need to type in the message you want to encrypt then it will output the result



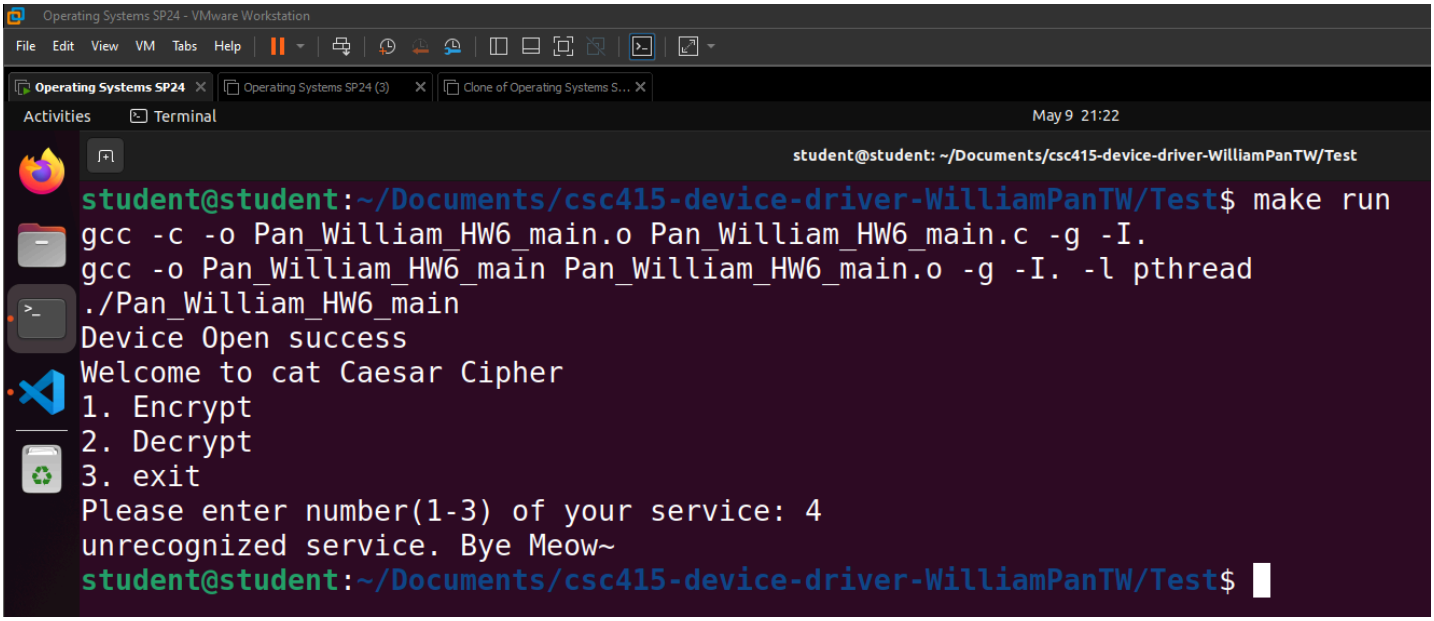
Step 3. If enter “2” you need to type in the message you want to decrypt then it will output the result



Step 4. If enter “3” you will exit the program



Step 5. If you enter instead of 1 ,2 ,3, it will output unrecognized service and exit the program



William Pan ID: 922867228
Github: WilliamPanTW CSC415-01 Operating Systems

Approach:

01. Understanding what is the task

Read the github background and task to plan what device driver I am going to write

02. Setup environment

Remember to update the version of Operating system and clone the virtual environment in case it crash

03. Read module Makefile

By understand the make file from module I know I have to set up the basename with my own module, and it will call the kernel directory with the correct version

04. Write init_module

By watching the lecture I knew I needed to initialize the module then register with the unique development device number(major and minor), afterward I also needed to initialize the cdev structure for writing driver code.

05. Write cleanup_module

The lecture also mentions the clean up function to unregister and remove devices, once the module is being removed from the kernel.

06. Understand file operation

Declare a function designator for the device driver to work in Test. Thus , it is important to know how this file operation structure works for the rest of the project. By command out function that is not yet finished, to be able to test out the error.

07. Understand how to install the module

After doing the init and clean up module we were able to test if our kernel object could compile. Making sure it is inserted(insmod) and listed it out(lsmod) to double check. Then make the directory(mknod) in the “dev” directory and change the permission for it to execute.

08. Make s shell script

After step 7 is completed make the previous command to script for better work efficiency. With both install and remove shells script .

09. Understand how to display kernel command

Use dmseg to displays kernel-related messages retrieved from the kernel ring buffer

10. Write a simple test

By understanding the make file in the test folder, simply create a last_first name c program. And try to open the kernel module we just created with open from file operation.

11. Understand how to write from module file operation

After doing some research I know that kernel space and user space memory is not sharedable, thus we have to copy_from_user meaning copy data from user space to kernel space and assign it to our structure in kernel space.

12. Understand how to read from module file operation

Then we have to use copy_to_user meaning copy data to user space from kernel space with the function we have modified back to the user to display.

13. Handle user input

Write a simple switch statement to hand user input for I/O control form file operation, to pass in the encrypt case or decrypt case in a unique number.

14. Test write from module file operation

Test write in our module of device driver by passing in the file descriptor from open and the text to be written.

15. Understand Caesar cipher

I want to do a device driver based on Caesar cipher that shifts each letter of the plaintext message by 3 letters. So, create two functions of encrypt and decrypt in a module for I/O control to decide what command the user asks and call the corresponding function to modify the data.

16. Understand how to control input and output from module file operation

Setup the same unique number from the test in the switch statement, and assign the corresponding case with the corresponding function(encrypt and decrypt) from step 15 to modify the data.

17. Switch encrypt and decrypt mod in test

If the I/O control does not fail then we read out our modified data either decrypted or encrypted from the kernel module and display it in the user space .

18. Setup release from file operations structure

Deallocated the associate data in kernel using "Vfree" to indicated to free from kernel heap

19. closes the file descriptor

This function triggers the kernel to call the .release function specified in the file operations structure and closes the file descriptor associated with the device file (/dev/Cat).

William Pan
Github: WilliamPanTW

ID: 922867228
CSC415-01 Operating Systems

Issues and Resolutions:

Issue 01: Keep typing printf in kernel and make run in module, encounter error to compile

Resolution 01: Correct it to printk because we are know writing in kernel and look into the make file found out it is “make all”

Issue 02: Cannot insert module and displays kernel-related messages r

```
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Module$ insmod Cat.ko
insmod: ERROR: could not insert module Cat.ko: Operation not permitted
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Module$ sudo insmod Cat.ko
[sudo] password for student:
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Module$ dmesg | tail
dmesg: read kernel buffer failed: Operation not permitted
```

Resolution 02: Use sudo to insert module and displays kernel-related messages r

```
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Module$ sudo dmesg | tail
[10263.010571] loaded cat module rom kernel meow meow
```

ISSUE 03: Cannot use install shells script

```
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Module$ ./installIt.sh
bash: ./installIt.sh: Permission denied
```

Resolution 03: Change the permission of the script

```
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Module$ sudo chmod 777 installIt.sh
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Module$ ./installIt.sh
[ 3807.271853] workqueue: hub_event hogged CPU for >10000us 16 times, consider switching to WQ_UNBOUND
[ 4716.947708] workqueue: pm_runtime_work hogged CPU for >10000us 8 times, consider switching to WQ_UNBOUND
[ 5405.476784] workqueue: hub_event hogged CPU for >10000us 32 times, consider switching to WQ_UNBOUND
[ 5533.945861] workqueue: netstamp_clear hogged CPU for >10000us 128 times, consider switching to WQ_UNBOUND
[ 8260.239775] workqueue: pm_runtime_work hogged CPU for >10000us 16 times, consider switching to WQ_UNBOUND
[ 9544.790979] workqueue: hub_event hogged CPU for >10000us 64 times, consider switching to WQ_UNBOUND
[13428.526598] Cat: loading out-of-tree module taints kernel.
[13428.526605] Cat: module verification failed: signature and/or required key missing - tainting kernel
[13428.528686] Register chardev succeeded :0
[13428.528692] loaded cat module rom kernel meow meow
```

ISSUE 04:Cannot make run before make directory in device directory

```
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Test$ make run
./Pan_William_HW6_main
return from open file -1
Device open error
Device file open error: No such file or directory
make: *** [Makefile:59: run] Error 255
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Test$
```

Resolution 04:
makes a character base directory entry with corresponding i-node(sudo mknod /dev/Cat c 415 0) to the device (/dev) for a module that just installs and changes the permission using chmod. Lastly verify using list in device directory(/dev) to see if it exists.

ISSUE 05: Origin I set my Input/output control switch switch in numerical of 1 is encrypt and 2 is decrypt , but It cannot be process and output Bad address

```
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Test$ make run
gcc -c -o Pan_William_HW6_main.o Pan_William_HW6_main.c -g -I.
gcc -o Pan_William_HW6_main Pan_William_HW6_main.o -g -I. -l pthread
./Pan_William_HW6_main
Welcome to cat Caesar Cipher
return from open file 3
Device Open success
1. Encrypt
2. decrypt
3. exit
Please enter number of your service: 2
Enter a message to decrypt: ww
write 2 to device success
control reuturn : -1
Failed to set encrypt mode: Bad address
make: *** [Makefile:59: run] Error 255
student@student:~/Documents/csc415-device-driver-WilliamPanTW/Test$ ./testIt.sh
```

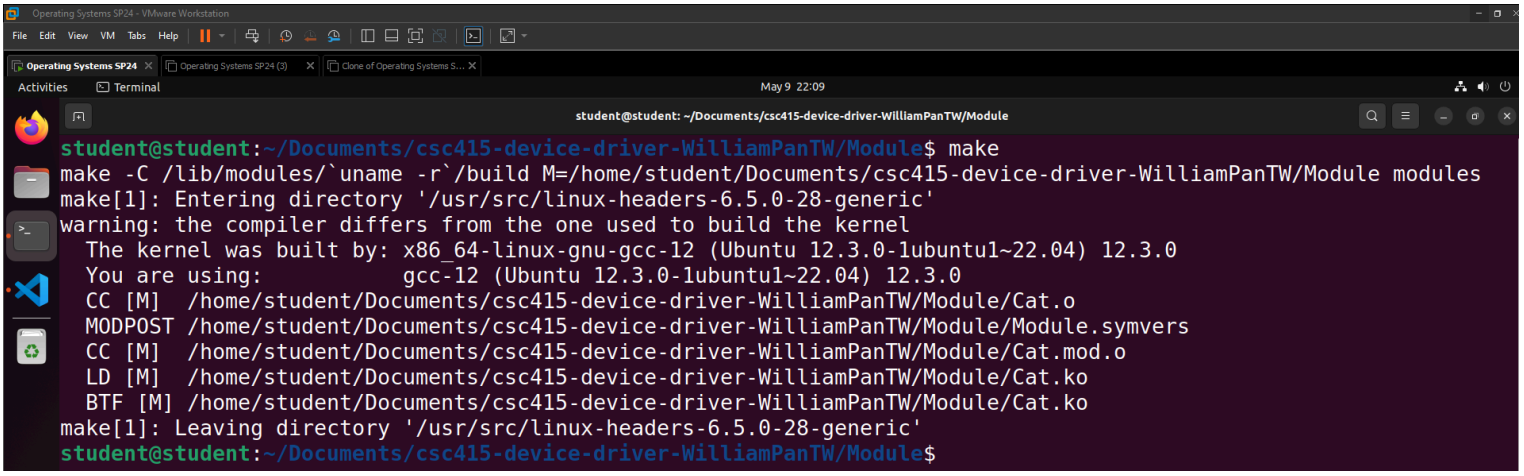
Resolution 05: After some research I found out ioctl should be unique, as explained in the Linux Device Drivers book(INtime SDK Help),thus I added the macro (_IOR) to create a unique ioctl identifier and it fixed it .

William Pan
Github: WilliamPanTW

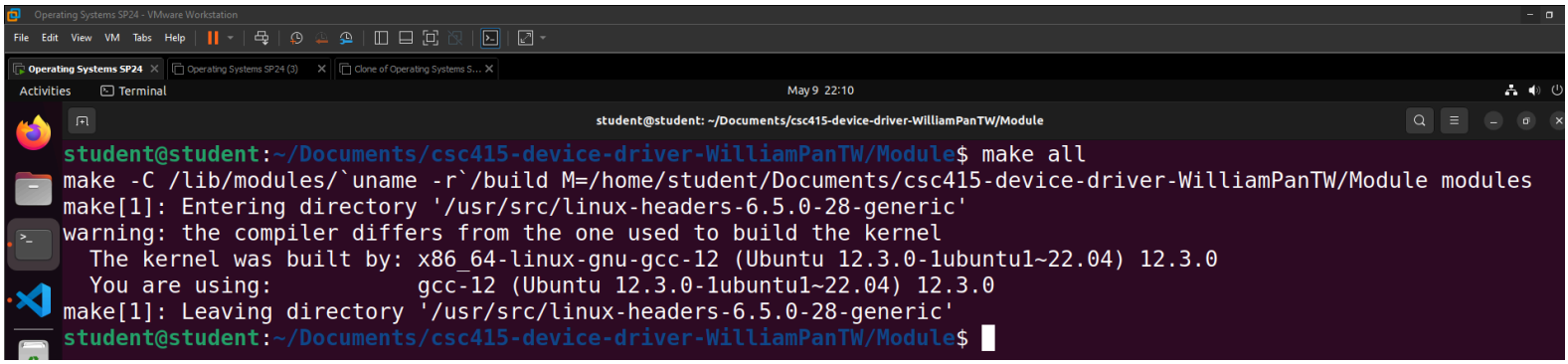
ID: 922867228
CSC415-01 Operating Systems

Analysis: (no Analysis section needed for this assignment)

Screenshot of compilation of Module :



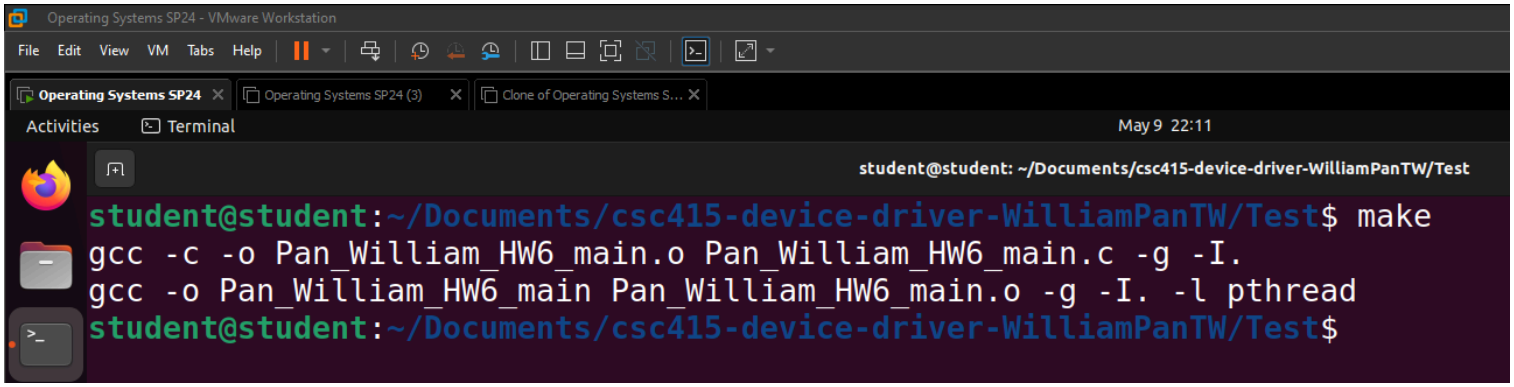
Screen shot(s) of the execution of the program in Module:



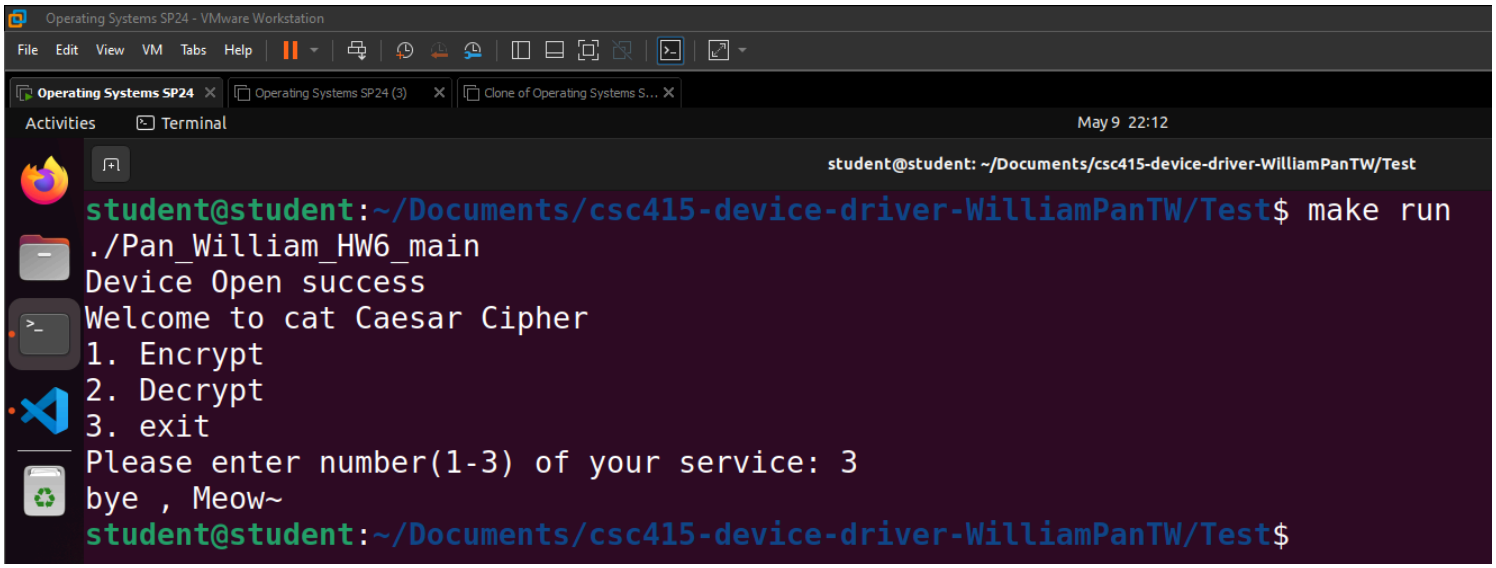
William Pan
Github: WilliamPanTW

ID: 922867228
CSC415-01 Operating Systems

Screenshot of compilation of Test:



Screen shot(s) of the execution of the program in Test:



William Pan
Github: WilliamPanTW

ID: 922867228
CSC415-01 Operating Systems

Screenshots of valid loadable and unloadable of device driver: