Mark Kim ID: 918204214
Github: mkim797 CSC415 Operating Systems

Assignment 6 – Device Driver

Description:
This assignment is to write a device driver for Linux in C along with a test application that can interface with the device driver and control its operation. My particular device driver is an encryption character device. We can write text/characters into the device, which will then copy the text from the user space into the kernel space. From there the data can be encrypted and decrypted. Through the IoCtl, we can control the decryption, encryption, and change the key for the Caesar cipher being used. My test application allows the user to specify an encryption key, and text to be encrypted. It then shows the user the text that is to be encrypted, the encrypted output, and finally decrypts it to reveal the unencrypted contents again.

Approach / What I Did:
I began with just recreating a null driver to understand the workings of a device driver as well as the processes for installing/uninstalling it. Once I had a working null driver, I moved onto simply copying data onto and off of the device. Finally, I worked on the encryption algorithm and the IoCtl.

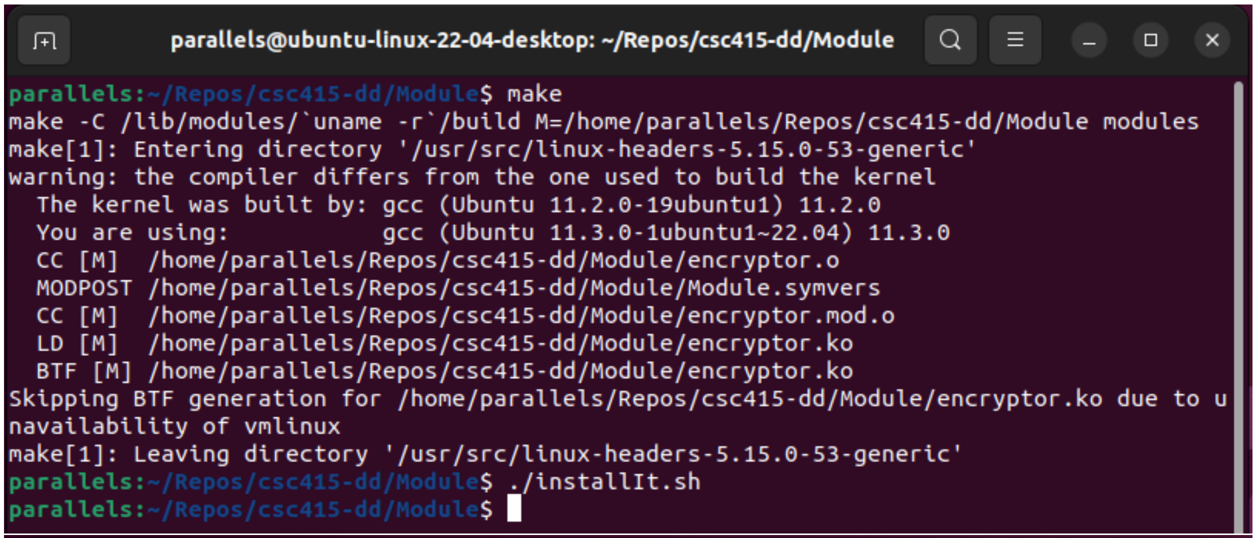
Issues and Resolutions:

Once again, I was faced with Segmentation Faults. But since the amount of data being handled was relatively small, I was able to hunt down those segmentation faults pretty easily. They were caused by small arithmetic mistakes with buffer lengths.

The second problem I ran into was the encryption algorithm itself. I was having difficulty with the conversion because of the null terminator. If my key was within a certain range, it would cause the encryption to abruptly stop since C strings are null terminated. So if a particular character was converted to null, the encryption would stop since the system would think that the string was at its end. This was fixed by doing a few small changes to the math of the cipher and also limiting the key range.

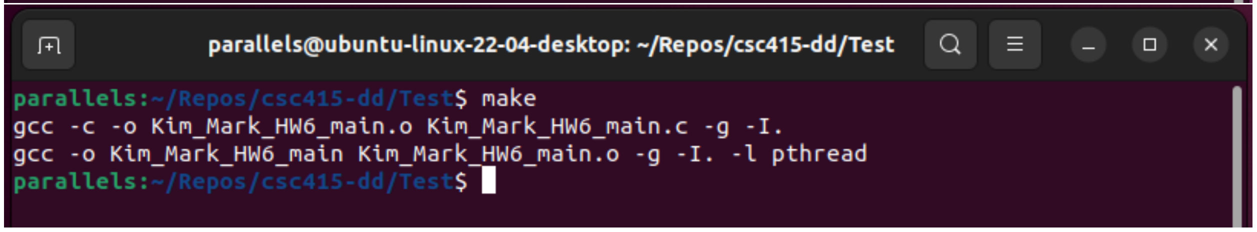
Analysis: (If required for the assignment). NOT REQUIRED

Screen shot of compilation:



parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-dd/Module

```
parallels:~/Repos/csc415-dd/Module$ make
make -C /lib/modules/`uname -r`/build M=/home/parallels/Repos/csc415-dd/Module modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-53-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0
  You are using:           gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
  CC [M]  /home/parallels/Repos/csc415-dd/Module/encryptor.o
  MODPOST /home/parallels/Repos/csc415-dd/Module/Module.symvers
  CC [M]  /home/parallels/Repos/csc415-dd/Module/encryptor.mod.o
  LD [M]  /home/parallels/Repos/csc415-dd/Module/encryptor.ko
  BTF [M] /home/parallels/Repos/csc415-dd/Module/encryptor.ko
Skipping BTF generation for /home/parallels/Repos/csc415-dd/Module/encryptor.ko due to u
navailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-53-generic'
parallels:~/Repos/csc415-dd/Module$ ./installIt.sh
parallels:~/Repos/csc415-dd/Module$ 
```



parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-dd/Test

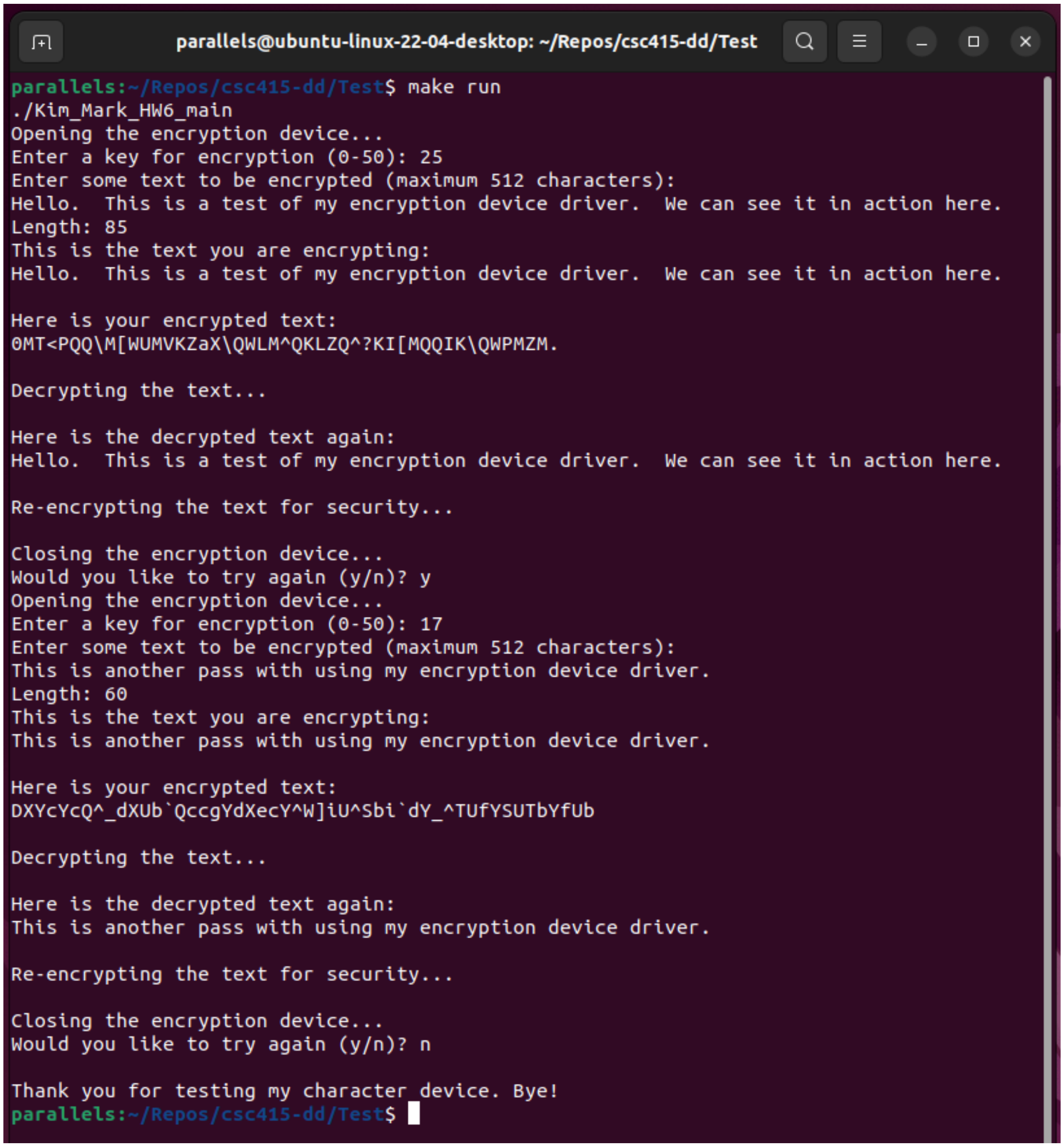
```
parallels:~/Repos/csc415-dd/Test$ make
gcc -c -o Kim_Mark_HW6_main.o Kim_Mark_HW6_main.c -g -I.
gcc -o Kim_Mark_HW6_main Kim_Mark_HW6_main.o -g -I. -l pthread
parallels:~/Repos/csc415-dd/Test$ 
```

SCREEN SHOTS OF EXECUTION ON NEXT PAGE

Mark Kim
Github: mkim797

ID: 918204214
CSC415 Operating Systems

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



parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-dd/Test

```
parallels:~/Repos/csc415-dd/Test$ make run
./Kim_Mark_HW6_main
Opening the encryption device...
Enter a key for encryption (0-50): 25
Enter some text to be encrypted (maximum 512 characters):
Hello.  This is a test of my encryption device driver.  We can see it in action here.
Length: 85
This is the text you are encrypting:
Hello.  This is a test of my encryption device driver.  We can see it in action here.

Here is your encrypted text:
0MT<PQQ\M[WUMVKZaX\QWLM^QKLZQ^?KI[MQQIK\QWPMZM.

Decrypting the text...

Here is the decrypted text again:
Hello.  This is a test of my encryption device driver.  We can see it in action here.

Re-encrypting the text for security...

Closing the encryption device...
Would you like to try again (y/n)? y
Opening the encryption device...
Enter a key for encryption (0-50): 17
Enter some text to be encrypted (maximum 512 characters):
This is another pass with using my encryption device driver.
Length: 60
This is the text you are encrypting:
This is another pass with using my encryption device driver.

Here is your encrypted text:
DXYcYcQ^_dXUb`QccgYdXecY^W]iU^Sbi`dY_^TUfYSUTbYfUb

Decrypting the text...

Here is the decrypted text again:
This is another pass with using my encryption device driver.

Re-encrypting the text for security...

Closing the encryption device...
Would you like to try again (y/n)? n

Thank you for testing my character device. Bye!
parallels:~/Repos/csc415-dd/Test$
```