Denean Le
Github: ledenean
ID: 921330745
CSC415 Operating Systems

Assignment 6 – Device Driver

Description:
This assignment is to develop a device driver that can be loaded and run in Linux. The program I implemented is an encryption device driver, where the user chooses between encrypting or decrypting a string. Then, the driver will return the encrypted or decrypted string of what was inputted.

Approach / What I Did:
First, I started by writing the init_module and cleanup_module functions from the class lecture on November 8, 2022. After that, I started writing my write, read, open, close, and ioctl functions. For myWrite, I copied the string from the user and assigned it to my kernel buffer. Opposite from myWrite, myRead copies the kernel buffer to the user. For myOpen, I opened the file by allocating space for the kernel buffer and checking if it was able to vmalloc. In myClose, I freed the kernel buffer. In myIoCtl, I implemented a switch case where 0 represented encryption, while 1 represented decryption. For my encryption algorithm, I used the Caesar Cipher algorithm, where it substitutes each letter of a string with a letter that is five spaces ahead in the alphabet. For example, if the letter is A, the encrypted letter would be F. Decryption is the opposite, where each letter of a string is replaced with a letter five spaces behind in the alphabet.

How to Build:
To build the kernel module, you must enter the following commands:
cd csc415-device-driver-ledenean/
cd Module/
make
./installIt.sh

How to Test:
After building the module, you must enter the following commands:
cd ..
cd Test/
make
make run

When testing, the program will ask for the user's choice in encrypting or decrypting. Since it is the first run, it would be wise to start by encrypting, so enter the number 1 to encrypt. The program will then ask the user to enter a string to encrypt. After entering a string, the program will output the encrypted string. From there, the user may decrypt the output to get the original string.

Issues and Resolutions:

My first issue was relying too heavily on the example device driver in the lecture, SampleNullDev. I was still confused on what each function did, so I was referring to the example and trying to replace things based on my idea of encryption. The issue with that was that I had a lot of useless code that I really didn't need for what I wanted my device driver to do. It also kept me confused on whether or not certain aspects of SampleNullDev was irreplaceable for a device driver. I resolved this by removing things that weren't necessary, such as the "myds" structure, and many lines of code that involved the structure.

My next issue was that my encryption algorithm is incorrect. Based on what I've learned from the Caesar Cipher algorithm, it should theoretically work, but it doesn't. I am unsure if it is an issue with the algorithm or if it's an issue with how I am reading the changed kernel buffer. This issue still remains because I was not able to finish the device driver by the due date.

Analysis: (If required for the assignment)

Screen shot of compilation:
Compilation for Module main file

```
student@student-VirtualBox:~/csc415-device-driver-ledenean/Module$ make
make -C /lib/modules/`uname -r`/build M=/home/student/csc415-device-driver-leden
ean/Module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-135-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-135-generic'
student@student-VirtualBox:~/csc415-device-driver-ledenean/Module$
```

Compilation for Test main file

```
student@student-VirtualBox:~/csc415-device-driver-ledenean/Test$ make
gcc -c -o Le_Denean_HW6_main.o Le_Denean_HW6_main.c -g -I.
Le_Denean_HW6_main.c: In function ‘main’:
Le_Denean_HW6_main.c:39:13: warning: format ‘%s’ expects argument of type ‘char
*’, but argument 2 has type ‘char (*)[512]’ [-Wformat=]
     scanf("%s", &buffer);
             ~^   ~~~~~~~
Le_Denean_HW6_main.c:43:15: warning: passing argument 2 of ‘ioctl’ makes integer
 from pointer without a cast [-Wint-conversion]
     ioctl(fd, &option);
               ^
In file included from Le_Denean_HW6_main.c:15:0:
/usr/include/x86_64-linux-gnu/sys/ioctl.h:41:12: note: expected ‘long unsigned i
nt’ but argument is of type ‘int *’
 extern int ioctl (int __fd, unsigned long int __request, ...) __THROW;
            ^~~~~
gcc -o Le_Denean_HW6_main Le_Denean_HW6_main.o -g -I. -l pthread
student@student-VirtualBox:~/csc415-device-driver-ledenean/Test$
```

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