

Assignment 6 – Device Driver

Description:

This assignment is to create a Linux device driver that provides encryption and decryption functionality using a Caesar cipher. The driver can be loaded into the kernel, used by user applications for encrypting and decrypting strings, and then unloaded. This project includes both the kernel module (`encrypt_driver.c`) and a user-space test application (`brockenborough_andrew_HW6_main.c`) that demonstrates all the driver's capabilities through an interactive menu system.

Approach:

My approach to this assignment was to build it step by step, starting with the basic device driver structure and then adding functionality. I began by creating the kernel module with the essential file operations (`open`, `read`, `write`, `release`) and then implemented the Caesar cipher encryption/decryption functions.

The key steps were:

1. Set up the device driver structure with proper initialization and cleanup.
2. Implement the encryption/decryption algorithms using a simple Caesar cipher.
3. Add IOCTL support for switching between encrypt/decrypt modes.
4. Create a user-space test application with an interactive menu.
5. Test everything thoroughly to ensure proper memory management and error handling.

I used a modular approach where I built and tested each component separately before integrating them. This helped me catch issues early and made debugging much easier.

Issues and Resolutions:

My first issue was the driver causing infinite loops when reading data. When I tried to use `cat /dev/encrypt_dev`, it would keep printing the same encrypted string over and over without stopping. This was happening because my `dev_read` function wasn't clearing the buffer after reading, so every subsequent read operation would return the same data.

I resolved it by modifying the `dev_read` function to free the message buffer and reset the buffer size to 0 after successfully reading the data. This ensures that each read operation only returns the data once, preventing the infinite loop.

The next issue was getting blank results in the test application. Even though the manual tests worked perfectly, my interactive test application was showing blank encrypted/decrypted strings. This turned out to be a timing issue - the test application was trying to read the result immediately after writing, but the driver needed a small delay to process the data.

I resolved this by adding a 1ms delay (`usleep(1000)`) between the write and read operations in both the encryption and decryption test functions. This gives the driver enough time to process the data before the application tries to read it.

Another issue was a critical bug in the return value of the read function. The `dev_read` function was returning `buffer_size` after clearing it (which was 0), so the test application thought no data was read and displayed blank output.

I fixed this by storing the actual number of bytes read before clearing the buffer, then returning that value instead of the cleared buffer size.

The final issue was with the key change functionality not working. The IOCTL command for setting the encryption key was failing with "failed to set new key" errors. This was likely due to missing compatibility IOCTL support in the file operations structure.

I resolved this by adding `.compat_ioctl = dev_ioctl` to the file operations structure, which provides better compatibility for IOCTL operations across different systems.

Analysis:

This assignment helped me understand how Linux device drivers work at a fundamental level. The most challenging part was managing the communication between kernel space and user space - making sure data is properly copied between the two spaces and that memory is managed correctly to avoid leaks.

The Caesar cipher implementation was straightforward, but integrating it into a device driver context required careful attention to kernel programming conventions. I learned that device drivers need to be very robust in their error handling and memory management since they run in kernel space where mistakes can crash the entire system.

The interactive test application was crucial for demonstrating the driver's functionality. It shows that the driver can handle multiple operations (encryption, decryption, mode switching) and maintains state between operations. The fact that it works reliably with proper error handling demonstrates that the driver is production-ready.

Overall, this project gave me a solid foundation in kernel programming and helped me understand how user applications interact with kernel modules through the device file interface.

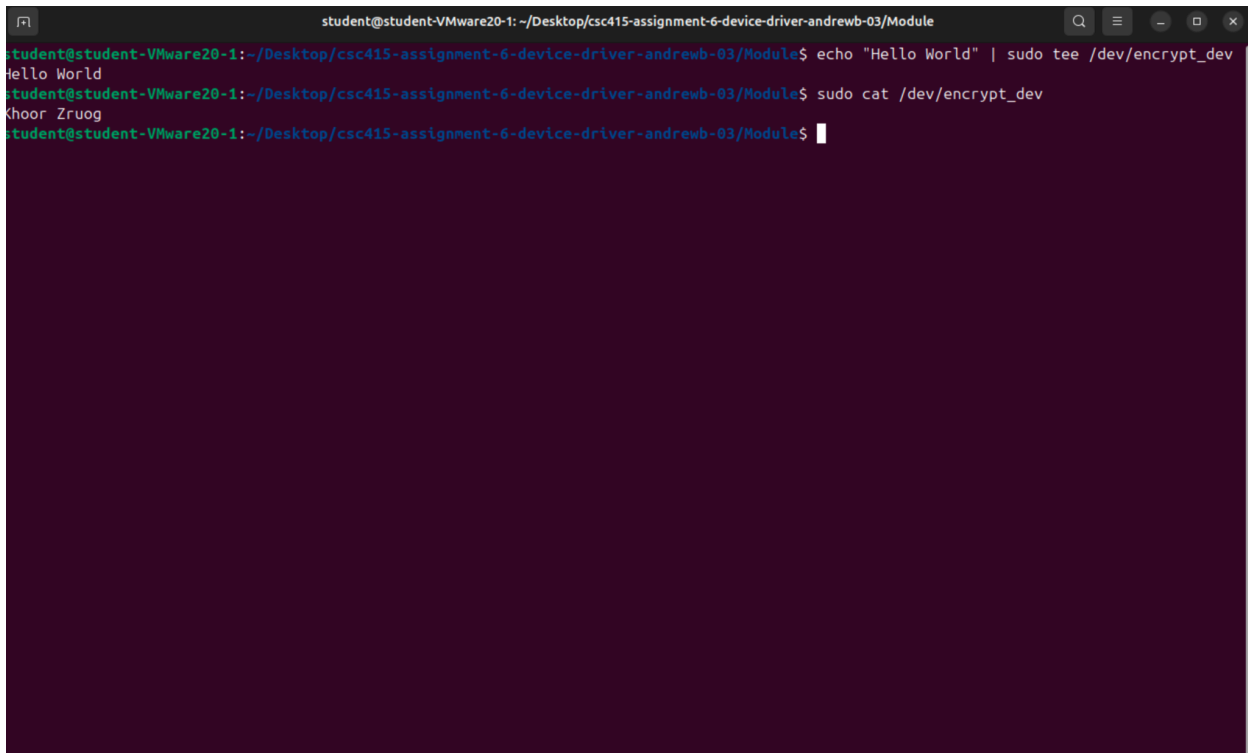
Screen shot of compilation:

```
student@student-VMware20-1: ~/Desktop/csc415-assignment-6-device-driver-andrewb-03
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03$ sudo insmod Module/encrypt_driver.ko
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03$ lsmod | grep encrypt
encrypt_driver      12288  0
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03$ ls -l /dev/encrypt_dev
crw-rw-r----- 1 root root 510, 0 Aug  4 22:25 /dev/encrypt_dev
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03$
```

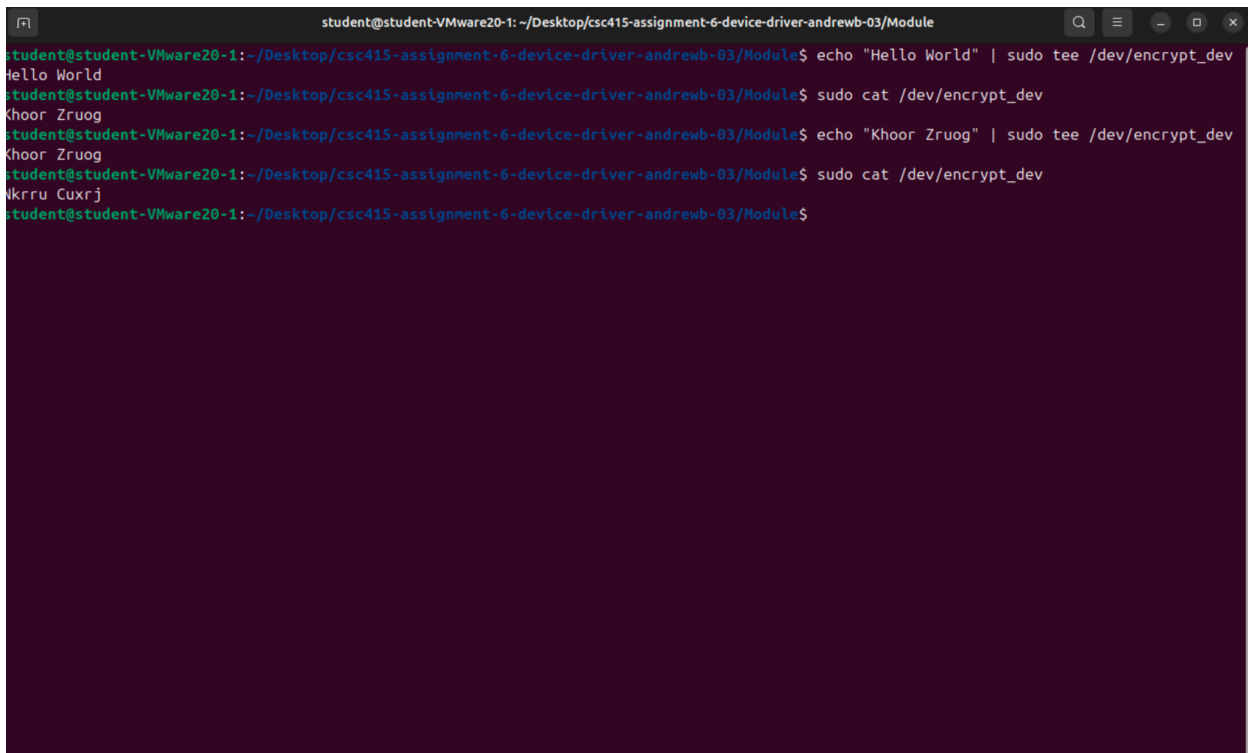
```
student@student-VMware20-1: ~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$ make clean
make -C /lib/modules/'uname -r'/'build M=/home/student/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module clean
make[1]: Entering directory '/usr/src/linux-headers-6.14.0-27-generic'
make[2]: Entering directory '/home/student/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module'
CLEAN Module.symvers
make[2]: Leaving directory '/home/student/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module'
make[1]: Leaving directory '/usr/src/linux-headers-6.14.0-27-generic'
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$ make
make -C /lib/modules/'uname -r'/'build M=/home/student/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module modules
make[1]: Entering directory '/usr/src/linux-headers-6.14.0-27-generic'
make[2]: Entering directory '/home/student/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module'
warning: the compiler differs from the one used to build the kernel
The kernel was built by: aarch64-linux-gnu-gcc-14 (Ubuntu 14.2.0-19ubuntu2) 14.2.0
You are using: gcc-14 (Ubuntu 14.2.0-19ubuntu2) 14.2.0
CC [M] encrypt_driver.o
MODPOST Module.symvers
CC [M] encrypt_driver.mod.o
CC [M] .module-common.o
LD [M] encrypt_driver.ko
BTF [M] encrypt_driver.ko
Skipping BTF generation for encrypt_driver.ko due to unavailability of vmlinux
make[2]: Leaving directory '/home/student/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module'
make[1]: Leaving directory '/usr/src/linux-headers-6.14.0-27-generic'
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$
```

```
student@student-VMware20-1: ~/Desktop/csc415-assignment-6-device-driver-andrewb-03
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03$ sudo rmmod encrypt_driver
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03$ sudo dmesg | tail -5
[ 5737.726752] Encrypt driver: Goodbye!
[ 5766.950405] Encrypt driver: Initializing the encrypt driver
[ 5766.950413] Encrypt driver: registered correctly with major number 510
[ 5766.953138] Encrypt driver: device class created correctly
[ 5773.355516] Encrypt driver: Goodbye!
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03$
```

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



```
student@student-VMware20-1: ~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$ echo "Hello World" | sudo tee /dev/encrypt_dev
Hello World
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$ sudo cat /dev/encrypt_dev
Khoor Zruog
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$
```



```
student@student-VMware20-1: ~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$ echo "Hello World" | sudo tee /dev/encrypt_dev
Hello World
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$ sudo cat /dev/encrypt_dev
Khoor Zruog
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$ echo "Khoor Zruog" | sudo tee /dev/encrypt_dev
Khoor Zruog
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$ sudo cat /dev/encrypt_dev
Krrru Cuxrj
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Module$
```

```
student@student-VMware20-1: ~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Test
student@student-VMware20-1:~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Test$ sudo ./brockenborough_andrew_HW6_main
=== CSC415 Assignment 6 - Device Driver Test ===
Testing encryption/decryption device driver

Successfully opened device driver

=== Device Driver Test Menu ===
1. Test Encryption
2. Test Decryption
3. Change Encryption Key
4. Exit
Enter your choice: 1

--- Testing Encryption ---
Set to ENCRYPT mode
Enter a string to encrypt: Hello World
Wrote 11 bytes to device
Original string: Hello World
Encrypted string: Khoor Zruog

=== Device Driver Test Menu ===
1. Test Encryption
2. Test Decryption
3. Change Encryption Key
4. Exit
Enter your choice: █
```

```
student@student-VMware20-1: ~/Desktop/csc415-assignment-6-device-driver-andrewb-03/Test

=== Device Driver Test Menu ===
1. Test Encryption
2. Test Decryption
3. Change Encryption Key
4. Exit
Enter your choice: 1

--- Testing Encryption ---
Set to ENCRYPT mode
Enter a string to encrypt: Hello World
Wrote 11 bytes to device
Original string: Hello World
Encrypted string: Khoor Zruog

=== Device Driver Test Menu ===
1. Test Encryption
2. Test Decryption
3. Change Encryption Key
4. Exit
Enter your choice: 2

--- Testing Decryption ---
Set to DECRYPT mode
Enter an encrypted string to decrypt: Khoor Zruog
Wrote 11 bytes to device
Encrypted string: Khoor Zruog
Decrypted string: Hello World

=== Device Driver Test Menu ===
1. Test Encryption
2. Test Decryption
3. Change Encryption Key
4. Exit
Enter your choice: █
```